UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

RAPHAEL DE AQUINO GOMES

# Implantação Eficiente de Múltiplas Coreografias de Serviços em Nuvens Híbridas

Goiânia
2017

**TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR AS TESES E DISSERTAÇÕES ELETRÔNICAS NA BIBLIOTECA DIGITAL DA UFG**

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei nº 9610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou *download*, a título de divulgação da produção científica brasileira, a partir desta data.

**1. Identificação do material bibliográfico**: **[ ] Dissertação** **[ X ] Tese**

**2. Identificação da Tese ou Dissertação**

Nome completo do autor: Raphael de Aquino Gomes

Título do trabalho: Implantação Eficiente de Múltiplas Coreografias de Serviços em Nuvens Híbridas

**3. Informações de acesso ao documento:**

Concorda com a liberação total do documento [ X ] SIM [ ] NÃO[1]

Havendo concordância com a disponibilização eletrônica, torna-se imprescindível o envio do(s) arquivo(s) em formato digital PDF da tese ou dissertação.

Raphael de Aquino Gomes

Data: 19 / 05 / 2017

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. A extensão deste prazo suscita justificativa junto à coordenação do curso. Os dados do documento não serão disponibilizados durante o período de embargo.

RAPHAEL DE AQUINO GOMES

# Implantação Eficiente de Múltiplas Coreografias de Serviços em Nuvens Híbridas

Tese apresentada ao Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Doutor em Ciência da Computação.

**Área de concentração:** Ciência da Computação.

**Orientador:** Prof. Fábio Moreira Costa

**Co-Orientador:** Prof. Ricardo Couto Antunes da Rocha

Goiânia
2017

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

de Aquino Gomes, Raphael
Implantação Eficiente de Múltiplas Coreografias de Serviços em Nuvens Híbridas [manuscrito] / Raphael de Aquino Gomes. - 2017.
cclvi, 256 f.: il.

Orientador: Prof. Dr. Fábio Moreira Costa; co-orientador Dr. Ricardo Couto Antunes da Rocha.
Tese (Doutorado) - Universidade Federal de Goiás, Instituto de Informática (INF), Programa de Pós-Graduação em Ciência da Computação em rede (UFG/UFMS), Goiânia, 2017.
Bibliografia. Apêndice.
Inclui siglas, gráfico, tabelas, algoritmos, lista de figuras, lista de tabelas.

1. computação em nuvem. 2. coreografia de serviços. 3. implantação de serviços. 4. restrição não-funcional. I. Moreira Costa, Fábio, orient. II. Título.

CDU 004

UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA
DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

## Ata de Defesa de Tese de Doutorado

Aos seis dias do mês de abril de dois mil e dezessete, no horário das oito horas e trinta minutos, foi realizada, nas dependências do Instituto de Informática da UFG, a defesa pública da Tese de Doutorado do aluno Raphael de Aquino Gomes, matrícula no. 2012 0630, intitulada **"Implantação Eficiente de Múltiplas Coreografias de Serviços em Nuvens Híbridas"**.

A Banca Examinadora, constituída pelos professores:

Prof. Dr. Fábio Moreira Costa – INF/UFG - orientador

Prof. Dr. Ricardo Couto Antunes da Rocha – DCC Catalão/UFG - coorientador

Prof. Dr. Bruno Richard Schulze – LNCC

Prof. Dr. Daniel de Angelis Cordeiro – EACH/USP

Prof. Dr. Edson Norberto Cáceres – FACOM/UFMS

Prof. Dr. Humberto José Longo - INF/UFG

emitiu o resultado:

(X) Aprovado

( ) Aprovado com revisão

(A Banca Examinadora deve definir as exigências a serem cumpridas pelo aluno na revisão, ficando o orientador responsável pela verificação do cumprimento das mesmas.)

( ) Reprovado

com o seguinte parecer: _____

Prof. Dr. Fábio Moreira Costa

Prof. Dr. Ricardo Couto Antunes da Rocha

Prof. Dr. Bruno Richard Schulze

Prof. Dr. Daniel de Angelis Cordeiro

Prof. Dr. Edson Norberto Cáceres

Prof. Dr. Humberto José Longo

**Raphael de Aquino Gomes**

Possui graduação (2006) e mestrado (2009) em Ciência da Computação pelo Instituto de Informática (INF) da Universidade Federal de Goiás (UFG). Realizou parte da pesquisa de doutorado em um estágio no *Institut National de Recherche en Informatique et en Automatique* (INRIA), França. Foi bolsista da Fundação de Amparo à Pesquisa do Estado de Goiás (FAPEG) e do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) durante o doutorado. Desde 2011 é professor efetivo do Instituto Federal de Educação, Ciência e Tecnologia de Goiás – Câmpus Goiânia. Tem experiência na área de Ciência da Computação, com ênfase em Sistemas Distribuídos, atuando principalmente nos seguintes temas: *Middleware* para Computação nas Nuvens, Modelos em Tempo de Execução e Sistemas Adaptativos. *Sun Certified Java Programmer*.

À minha avó Bárbara, que em sua maneira foi a primeira a acreditar que eu me tornaria um Doutor.

# Agradecimentos

Acima de tudo, agradeço a Deus por ter me proporcionado a oportunidade, a sabedoria e a força para vencer mais este desafio.

À minha família, pelo apoio incondicional em todos os momentos. Em especial, agradeço meus pais, José e Isabel, por terem aberto mão em vários momentos da minha companhia para que eu pudesse me dedicar aos meus estudos. Certamente só cheguei até aqui porque vocês foram os primeiros a me incentivar e a acreditarem em mim. Agradeço também a minhas irmãs, Thayza e Thayana, com quem eu sempre pude contar para resolver vários problemas, e a meus sobrinhos, José Neto e Ana Laura, que com certeza representam uma das grandes motivações de eu querer sempre melhorar. E também não poderia deixar de agradecer ao Roberson e à Darci que, apesar de não serem oficialmente parte da família, para mim são tão importante quanto. Obrigado "Dona" Darci por estar sempre me incentivando e por cuidar de mim como filho. Obrigado Roberson, sobretudo pelo apoio, pelo companheirismo, pela paciência, pelas correções sensacionais no texto e por ser essa pessoa especial que espero sempre ter ao meu lado.

A meus amigos, Lídia do Carmo, Mayara Carvalho, Alex Oliveira, Valdilaine Vieira, Luciana Nishi, Carmem Centeno, que também sempre me apoiaram e que me ajudaram a lembrar que existe vida além do doutorado. De maneira particular, aos amigos e verdadeiros irmãos que serão certamente um dos mais valiosos legados que conquistei no meu período na França, e com quem vivenciei alguns dos momentos mais enriquecedores da minha vida: Cynara Kern, Guilherme de Melo, Giselle Utida, Mariana Pombo, Gabriela Mitidieri e Diego da Silva. *Vous êtes formidables!*

Aos meus orientadores, Dr. Fábio Costa e Dr. Ricardo da Rocha, pelos ensinamentos e direção nesta trajetória. Sobretudo, obrigado Fábio pelos quase 15 anos, assumindo não somente o papel de orientador mas de alguém que me espelho. Agradeço profundamente por ter compartilhado comigo toda sua experiência e sabedoria. Certamente, tudo o que sou hoje como pesquisador devo a você.

Ao Dr. Nikolaos Georgantas pela dedicada orientação durante meu estágio sanduíche, pelas discussões científicas e pelos diversos momentos de descontração durante os nossos cafés. A todos os colegas do time Mimove, em especial ao Georgios

Bouloukakis, com quem tive a oportunidade de discutir grande parte do meu trabalho e aprender muitos aspectos da cultura grega. $\epsilon\upsilon\chi\alpha\rho\iota\sigma\tau\omega$.

A todos que colaboraram para que este projeto fosse desenvolvido. Ao Leonardo Leite pela consultoria e sugestões com relação à implementação. A todos os colegas do Instituto de Informática da UFG, em especial Júnio César e Leandro Alexandre, que vivenciaram as mesmas dificuldades que eu e com quem compartilho os bônus e ônus desta experiência.

Ao Instituto Federal de Goiás pela concessão do afastamento que, embora não tenha sido por todo o período do meu curso, permitiu que eu me dedicasse integralmente ao doutorado.

À Fundação de de Amparo à Pesquisa do Estado de Goiás (FAPEG) pela concessão de bolsa (Edital 003/2013, Processo 201310267000281) e pelo financiamento do projeto (Edital 012/2012, Processo 201310267001106), o que foi fundamental para que o trabalho fosse desenvolvido de maneira efetiva.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pela concessão da bolsa que permitiu a realização do estágio sanduíche (Edital Doutorado Sanduíche - SWE, Processo 249809/2013-3). As colaborações e experiências desenvolvidas durante este período foram de fundamental importância para o sucesso em sua conclusão.

Ao *Institut National de Recherche en Informatique et en Automatique* (INRIA), por ter me acolhido durante o estágio sanduíche e pelos infindáveis conhecimentos, não somente em computação mas especialmente em cultura francesa, obtidas neste ambiente.

A todos que direta ou indiretamente contribuíram para a conclusão desta tese.

*Peu à peu, miette à miette, goutte à goutte et cœur à cœur*.
(Pouco a pouco, migalha a migalha, gota a gota e coração a coração.)

**Jean-Jacques Goldman**,
*1951-?*.

# Resumo

Gomes, Raphael de Aquino. **Implantação Eficiente de Múltiplas Coreografias de Serviços em Nuvens Híbridas**. Goiânia, 2017. 256p. Tese de Doutorado . Instituto de Informática, Universidade Federal de Goiás.

Esta tese apresenta uma abordagem baseada em modelos para abstrair, simplificar e automatizar as decisões de gerenciamento de recursos em nuvem ao implantar um conjunto de coreografias de serviços sujeitas a restrições não-funcionais. Dada uma descrição em alto nível das coreografias e das restrições relacionadas, a abordagem realiza de maneira autônoma a estimativa, a seleção e a alocação dos recursos em um ambiente de nuvem híbrida com múltiplos provedores, enquanto reduz os custos associados à utilização dos recursos e o atraso de comunicação entre os serviços. A principal motivação para o seu desenvolvimento se deve ao fato das coreografias de serviço serem amplamente utilizadas para o desenvolvimento de soluções com necessidades complexas, que geralmente compartilham serviços entre si. Isso faz com que o gerenciamento de recursos esteja sujeito a uma série de desafios, principalmente relacionados aos diferentes papéis que um serviço assume, à interferência que uma restrição causa em outra e à grande quantidade de tipos de recurso a serem considerados. A tese também propõe uma arquitetura que agrega à abordagem uma proposta para a automação das atividades relacionadas ao gerenciamento dinâmico de recursos como forma de reparar violações detectadas sobre as restrições. Esta arquitetura foi parcialmente implementada em um protótipo que foi utilizado na avaliação da abordagem.

**Palavras–chave**

Computação em Nuvem, Coreografia de Serviços, Implantação de Serviços, Restrição Não-Funcional

# Abstract

Gomes, Raphael de Aquino. **Efficient Deployment of Multiple Service Choreographies in Hybrid Clouds**. Goiânia, 2017. 256p. Ph.D. Thesis . Instituto de Informática, Universidade Federal de Goiás.

This thesis proposes a model-based approach to abstracting, simplifying, and automating cloud resource management decisions to deploy a set of service choreographies subject to non-functional constraints. Given a high-level description of service choreographies and related constraints, the approach autonomously performs resource estimation, selection, and allocation in a hybrid cloud environment with multiple cloud providers whilst decreases resource utilization costs and inter-services communication overhead. The main motivation for this work is because service choreographies are widely used for the development of solutions with complex needs, with service sharing among them. This scenario turns resource management a challenging task, mainly due to the different roles that a service assumes, the interference among constraints, and a large number of available resource types. This thesis also proposes an architecture that extends the approach with strategies to dynamic resource management to face constraint violations. This architecture was partially implemented in a prototype that was used in the proposed approach evaluation.

**Keywords**

Cloud Computing, Service Choreography, Service Deployment, Non-Functional Constraint

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Lista de Algoritmos

# Lista de Códigos de Programas

# Lista de Siglas

| | |
|---|---|
| ACU | *Azure Compute Unit* |
| AHP | *Analytic Hierarchy Process* |
| AWS | *Amazon Web Services* |
| BFS | *Breadth-First Search* |
| BPD | *Business Process Diagram* |
| BPEL | *Business Process Execution Language* |
| BPMN | *Business Process Modeling Notation* |
| BPSS | *Business Process Specification Schema* |
| CEP | *Complex Event Processing* |
| CEx | *Cloud Exchange* |
| CloudMF | *Cloud Modelling Framework* |
| CPU | *Central Processing Unit* |
| CRM | *Customer Relationship Management* |
| CSDP | *Constraint-aware resource Synthesis for multiple service Choreography Problem* |
| CSMIC | *Cloud Service Measurement Index Consortium* |
| CSP | *Cloud Service Provider* |
| CTL | *Computation Tree Logic* |
| CVM | *Communication Virtual Machine* |
| DSML | *Domain-Specific Modeling Language* |
| ECU | *Elastic Compute Unit* |
| EE | *Enactment Engine* |
| EPPSL | *Extended Process Pattern Specification Language* |
| GCE | *Google Compute Engine* |

| | |
|---|---|
| IaaS | *Infrastructure as a Service* |
| IME | Instituto de Matemática e Estatística |
| INRIA | *Institut National de Recherche en Informatique et en Automatique* |
| KWH | Kilowatt-hora |
| MARTE | *Modeling and Analysis of Real-Time Embedded systems* |
| MCDM | *Multiple Criteria Decision Making* |
| MDE | *Model Driven Engineering* |
| MI | Milhões de Instruções |
| MMKP | *Multiple-choice Multi-dimension Knapsack Problem* |
| MODAClouds | *MOdel-Driven Approach for design and execution of applications on multiple Clouds* |
| mOSAIC | *Open source API and Platform for multiple Clouds* |
| MSC | *Message Sequence Chart* |
| OMG | *Object Management Group* |
| PaaS | *Platform as a Service* |
| PMM | *Property Meta-Model* |
| PT | *Parallel Tree* |
| PyBPMN | *Performability-enabled BPMN* |
| Q4BPMN | *Quality for BPMN* |
| QoS | *Quality of Service* |
| REST | *Representational State Transfer* |
| RNF | Requisito Não-Funcional |
| RO | *Resource Orchestration* |
| SaaS | *Software as a Service* |
| SCG | *Service Concurrence Graph* |
| SDG | *Service Dependency Graph* |
| SLA | *Service Level Agreement* |
| SLA | *Service Level Agreement* |
| SMI | *Service Measurement Index* |

| | |
|---|---|
| SOAP | *Simple Object Access Protocol* |
| SRG | *Service Relationship Graph* |
| TCO | *Total Cost of Ownership* |
| TI | Tecnologia da Informação |
| TOSCA | *OASIS Topology and Orchestration Specification for Cloud Applications* |
| URI | *Uniform Resource Identifier* |
| USP | Universidade de São Paulo |
| VM | *Virtual Machine* |
| VSBPP | *Variable Size Bin Packing Problem* |
| WS-CDL | *Web Services Choreography Description Language* |
| WSFL | *Web Services Flow Language* |
| XaaS | *Anything as a Service* |

# Parte I

# Motivação e contexto

CAPÍTULO 1

# Introdução

Recentes avanços no desenvolvimento de sistemas de *software* evidenciaram uma mudança de paradigma, partindo de soluções convencionais (que se baseiam no uso de bibliotecas ligadas ao programa, com chamadas a funções através do compartilhamento de memória) em direção à adoção de aplicações baseadas em serviços, que são criadas usando implementações sob medida ou utilizando componentes previamente desenvolvidos. Nesta nova realidade, a possível limitada funcionalidade que um serviço isolado oferece nem sempre satisfaz as necessidades complexas das aplicações ou reflete de maneira apropriada seus emaranhados processos de negócio [7].

Neste cenário, composições de serviços representam uma abordagem mais apropriada. Contudo, devido à extensa quantidade de serviços interconectados e à existência de muitos pontos de interação, manter um único ponto de coordenação não é uma boa estratégia. Em virtude disso, uma solução promissora é o uso de serviços coordenados e descentralizados, divididos através de coreografias.

Coreografias são composições de serviços que implantam processos de negócio distribuídos, visando reduzir as trocas de mensagens de controle e distribuir a lógica do negócio. Neste modelo de composição não há a necessidade de controladores centralizados, uma vez que cada serviço "sabe" quando executar suas operações e com quais outros serviços interagir [27]. Quando os serviços participantes executam seus papéis, diz-se que a coreografia foi encenada [93].

A criação de coreografias de serviços é guiada não somente pelas propriedades funcionais dos serviços e pelas dependências entre eles, mas também por restrições não-funcionais, como qualidade de serviço (*Quality of Service* – QoS), regulamentações ou obrigações legais impostas sobre a execução dos serviços, propriedades associadas ao ambiente de execução, *etc.* Essas restrições podem ser especificadas visando à execução ou encenação da coreografia como um todo (fim-a-fim) ou a execução de serviços específicos.

As decisões tomadas antes dos serviços serem executados, mais precisamente durante sua implantação, afetarão como as restrições serão avaliadas, uma vez que essas decisões indicarão o ambiente onde esses serviços serão implantados e, portanto,

estabelecerão como a execução do serviço ocorrerá. Isso gera a necessidade de uma estratégia eficiente de provisionamento de recursos. Por estratégia eficiente, neste caso, entende-se a seleção de um conjunto de recursos que satisfaça todas as restrições especificadas e favoreça algum critério que seja relevante do ponto de vista do implantador da coreografia, por exemplo, o custo de sua encenação.

O aperfeiçoamento da Tecnologia da Informação (TI) possibilitou que a implantação de componentes de *software* seja realizada extrapolando os recursos disponíveis na infraestrutura privada da organização, através do modelo de negócio conhecido como Computação em Nuvem (*Cloud Computing*) [79, 282]. Este modelo faz com que a computação passe a ser utilizada da mesma forma que serviços como eletricidade, água e telefonia [50], permitindo acesso sob demanda, através de rede, a um conjunto de recursos computacionais configuráveis, que podem ser rapidamente oferecidos e liberados com um mínimo de esforço, gerenciamento e interação por parte de seu provedor [178]. Mais especificamente, se destaca a categoria de infraestrutura como serviço (*Infrastructure as a Service* – IaaS) que oferece acesso ubíquo e sob demanda a um ilimitado conjunto de recursos com propriedades distintas.

Neste cenário, estratégias de gerenciamento de recursos devem permitir a tradução das restrições não-funcionais para parâmetros de infraestrutura e o seu mapeamento para tipos pré-definidos de máquinas virtuais (*Virtual Machines* – VMs), o que inclui características como capacidade de CPU, memória, armazenamento e largura de banda. Além dessa tradução é preciso decidir onde alocar o tipo de VM selecionado, de forma a reduzir o atraso de comunicação entre os serviços, uma vez que esse atributo pode ter grande impacto no desempenho e em outros critérios de qualidade da composição [127].

No cenário considerado para implantação de coreografias de serviços, ilustrado na Figura 1.1, o usuário é responsável por todas as atividades relacionadas ao gerenciamento de recursos. A primeira atividade por ele realizada é a especificação das coreografias de serviços, que inclui a indicação dos serviços que serão compostos e como a interação entre eles ocorre. Em complemento a isso, consideramos também a especificação de um conjunto de restrições não-funcionais que devem ser satisfeitas na execução dos serviços, sendo assim, influenciam sua implantação.

Com base na especificação estabelecida, e nas restrições que devem ser satisfeitas é realizada a estimativa de recursos que consiste em determinar a capacidade necessária de recurso para implantar cada serviço. Em seguida, a capacidade é então utilizada para guiar a seleção dos tipos de VM mais apropriados para atender essa demanda. Essa seleção ocorre considerando um ambiente de nuvem híbrido, sobre o qual os recursos selecionados são então alocados.

Durante a execução dos serviços, o usuário monitora atributos relacionados

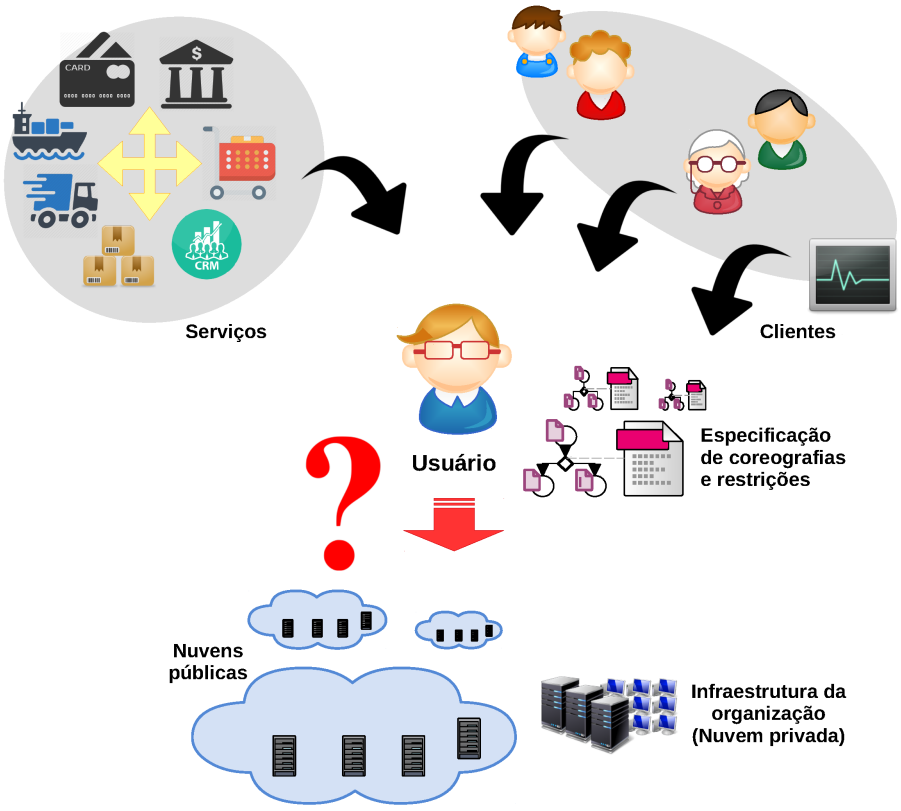



**Figura 1.1:** *Cenário considerado na tese.*

aos recursos, como nível de utilização; assim como obtém *feedback* dos clientes que utilizam os serviços, como nível de satisfação. Com base nos dados de monitoramento, adaptações são realizadas no provisionamento de recursos, visando que as restrições continuem a ser satisfeitas.

Nesta tese, propomos uma solução para automação das atividades relacionadas ao gerenciamento de recursos, após a definição dos serviços que compõem as coreografias e das restrições não-funcionais associadas. Esta automação tem como objetivo permitir que estas atividades possam ser realizadas de maneira eficiente, sem interação direta do usuário.

## 1.1 Motivação

Uma estratégia para o provisionamento eficiente de recursos é levar em consideração que aplicações orientadas a serviços podem apresentar um conjunto comum de funcionalidades, o que torna desejável a criação de composições com compartilhamento de serviços. Esta necessidade torna a satisfação de restrições ainda mais desafiadora, principalmente devido à degradação de QoS em serviços compartilhados e à possível existência de múltiplas restrições sobre um mesmo serviço. Como exemplo, é possível utilizar uma mesma métrica de QoS para restringir um mesmo serviço que

participa de duas ou mais coreografias usando diferentes valores-alvo em cada uma delas. Por esse motivo, a estimativa de recursos deve ser realizada de forma a decidir a capacidade mais adequada para o recurso que será usado para implantar cada serviço, satisfazendo restrições sobre serviços específicos e balanceando, de maneira eficaz, a contribuição do serviço na satisfação de restrições fim-a-fim especificadas para cada coreografia da qual este participa.

A possibilidade de expressar requisitos de recursos com base nas necessidades dos serviços que esses recursos deverão hospedar favorece um gerenciamento mais preciso da capacidade computacional disponível. Como consequência, torna-se possível alocar de forma mais precisa a capacidade necessária à coreografia, além de facilitar o redimensionamento dinâmico da capacidade alocada em face de mudanças.

Nesse cenário, o provisionamento de recursos não pode ser baseado em um único provedor de nuvem, uma vez que a habilidade de executar e gerenciar sistemas que usam múltiplos ambientes de nuvem permite explorar as peculiaridades de cada um destes ambientes e consequentemente otimizar o desempenho, disponibilidade e custo das aplicações [90]. Em adição a isso, o provisionamento deve considerar os recursos existentes na infraestrutura da própria organização e o redirecionamento de requisições para nuvens públicas apenas quando a infraestrutura privada estiver completamente alocada (geralmente em períodos de grande demanda), o que é conhecido como *cloud bursting* [231].

Para efetivamente tirar vantagem da elasticidade disponível ao utilizar ambientes de nuvem, o mapeamento de serviços para recursos deve considerar o modelo de cobrança baseado em funções de utilidade empregado por provedores de nuvem [282] e encontrar um equilíbrio entre as restrições não-funcionais e o custo. Essa tarefa constitui um grande desafio porque, geralmente, um mesmo tipo de VM pode ser instanciado em diferentes localidades com custo variável, criando um vasto conjunto de opções. Delegar esta decisão (e as demais relacionadas ao gerenciamento de recursos) para a pessoa responsável por implantar as coreografias não é uma tarefa simples, pois esta é uma atividade suscetível a erros dado o número de opções e porque gerenciamento manual de recursos é uma tarefa demorada e complexa [82]. Apesar disso, muitas organizações ainda realizam essas tarefas de forma manual, tornando o processo moroso, propenso a erros e não reprodutível [80].

## 1.2 Definição do problema

O problema investigado nesta tese refere-se ao gerenciamento de recursos para a implantação de coreografias de serviços com restrições não-funcionais associ-

adas. Mais precisamente, o principal problema abordado é definido a seguir:

> *Dado um conjunto de coreografias de serviços, com possíveis compartilhamentos de operações entre elas e restrições não-funcionais associadas, deve-se decidir a configuração ideal de recursos para implantar cada serviço e mapear esses recursos para ambientes de nuvem apropriados. A estimativa, o provisionamento de recursos e a implantação de serviços devem ser realizados de maneira autônoma e eficiente.*

Apesar da existência de muitos trabalhos que visam ao gerenciamento de recursos em nuvem [129, 173, 243], a implantação de composições de serviços em ambientes de nuvem com o mínimo de intervenção humana e levando em consideração aspectos não-funcionais associados aos serviços ainda é um tema pouco explorado. Porém, a estimativa e o provisionamento autônomo de recursos é uma necessidade cada vez mais eminente, pois essa funcionalidade está alinhada à filosofia que rege o modelo de IaaS, que é facilitar e tornar transparente a forma como esses recursos são gerenciados. Ao desenvolver esse tema de pesquisa, objetivamos propor uma solução que, em nível semelhante à abstração oferecida na alocação de recursos em ambientes de nuvem, abstrai também a estimativa e as demais etapas do provisionamento de recursos. Esta solução considera desafios adicionais relacionados à implantação de múltiplas composições de serviços, como a necessidade de satisfazer simultaneamente uma quantidade maior de restrições e a degradação de QoS causada pelo compartilhamento de serviços entre elas. Para tal, as principais questões de pesquisa que buscamos responder nessa tese são listadas abaixo:

- **QP1:** *Como automatizar a implantação de múltiplas coreografias de serviços visando à satisfação de restrições com uso eficiente de recursos?*
- **QP2:** *Quais fatores e métodos devem ser considerados para decidir (de maneira autônoma) a configuração de recursos adequada para implantar um conjunto de coreografias de serviços, dada sua especificação e restrições não-funcionais associadas?*
- **QP3:** *Como abstrair a estimativa e a seleção de recursos em um ambiente de nuvem híbrido com infraestrutura privada e múltiplos provedores de nuvem pública?*

A investigação dessas questões de pesquisa é motivada por uma série de dificuldades encontradas na satisfação de restrições ao implantar composições de serviços. Como exemplo, uma característica importante da estratégia de gerenciamento de

recursos é que ela deve garantir que haja capacidade computacional suficiente para atender a demanda sobre o serviço, satisfazendo as restrições requisitadas, ao mesmo tempo em que favorece o critério adotado para guiar o provisionamento (como redução do custo). Sem utilizar uma estratégia como a que propomos, o mapeamento das restrições para recursos de infraestrutura é geralmente realizado com base em experiências prévias dos desenvolvedores, dificultando esta tarefa quando esta experiência não existe (como quando a aplicação é um produto inédito).

A seguir detalhamos os objetivos do estudo apresentado nessa tese.

## 1.3 Objetivos

O objetivo geral deste trabalho consiste em propor, implementar e avaliar uma estratégia para a implantação automatizada e eficiente de múltiplas coreografias de serviços, sujeitas a restrições não-funcionais associadas aos serviços e à sua encenação. Essa estratégia considera o compartilhamento de serviços entre as coreografias e se beneficia da multiplicidade de tipos de recursos disponíveis em um ambiente híbrido formado por uma nuvem privada e múltiplas nuvens públicas. A estratégia visa automatizar as atividades relacionadas à estimativa, descoberta, seleção e alocação de recursos. A consecução deste objetivo geral se desdobra nos seguintes objetivos específicos:

- investigar e propor formas de modelagem de um conjunto de coreografias de serviços levando em consideração o compartilhamento de operações entre elas e restrições não-funcionais associadas à implantação e ao funcionamento dos serviços e das coreografias;
- prover mecanismos para a representação dos tipos disponíveis de recursos, de forma que seja possível tratar a heterogeneidade na representação de recursos e modelar, de maneira consistente, múltiplos provedores de nuvem. Também deve ser viável o uso dessa representação para seleção de recursos sem comprometer o tempo necessário para realizar a implantação;
- investigar e propor mecanismos para estimar a capacidade adequada de recursos necessários para implantar um conjunto de coreografias, dadas suas estruturas e restrições;
- investigar e propor mecanismos para seleção de recursos em uma nuvem híbrida, tendo como base a capacidade de recursos estimada e as restrições previamente estabelecidas;
- definir uma arquitetura que permita a implantação eficiente e autônoma de múltiplas coreografias de serviços usando os mecanismos propostos;

- construir um protótipo funcional da arquitetura definida, o qual servirá de base para avaliação dos mecanismos implantados; e
- avaliar, com base no protótipo, propriedades de efetividade e desempenho dos mecanismos propostos.

## 1.4 Contribuições

Esta tese propõe uma abordagem para abstrair, simplificar e automatizar decisões sobre gerenciamento de recursos virtualizados em ambientes de nuvem, e que satisfaça as restrições não-funcionais na implantação de múltiplas coreografias de serviços. Dada a descrição em alto nível de uma ou mais coreografias de serviços e restrições associadas, a abordagem proposta estima e provisiona de maneira autônoma a configuração de recursos mais apropriada para satisfazer as restrições. Para tal, modelamos o problema apresentado como um problema de otimização e propomos uma solução que busca minimizar o custo financeiro da implantação das coreografias e a sobrecarga de comunicação entre os serviços. Essa solução se destaca por ser a primeira abordagem (até onde é de nosso conhecimento) para implantação de múltiplas composições de serviços, considerando todas as atividades relacionadas ao gerenciamento de recursos, satisfazendo diferentes categorias de restrições não-funcionais.

A tese apresenta uma solução para provisionamento de recursos levando em consideração o compartilhamento de serviços e recursos. Espera-se que seu uso possibilite aos usuários implantar composições de serviços aproveitando, de forma otimizada, o vasto potencial de provedores de nuvem e tipos de recursos, permitindo que eles foquem nos aspectos do negócio enquanto delegam à solução os tecnicismos relacionados à alocação de recursos e implantação de serviços. Outras contribuições desta tese são descritas a seguir.

*i)* Propomos um modelo de coreografia de serviços que ajuda no processo de identificação, categorização e especificação de integração de recursos em nuvem. Através de um modelo de recursos gerado a partir do modelo da coreografia, a abordagem nos permite encontrar os recursos de *hardware* necessários em um ambiente híbrido formado por uma nuvem privada, e múltiplos provedores de nuvem pública.

*ii)* Propomos um arcabouço para especificação de restrições sobre coreografias de serviço que facilita a inclusão de métricas de QoS, oferece uma linguagem para descrição de restrições e implementa a geração automática de filtros a serem aplicados na seleção de recursos.

*iii)* Apresentamos uma solução eficiente e coordenada para a seleção de recursos que contempla necessidades comerciais emergentes com o aumento no número

de provedores de nuvem. Essa solução agrega um conjunto de estratégias para mapeamento de especificações de coreografias, determinando os recursos necessários para sua implantação e levando em consideração restrições impostas sobre seu funcionamento.

## 1.5 Estrutura da tese

A seguir é apresentada uma visão geral dos capítulos que compõem as partes desta tese.

**Parte II: Estado da arte**

**Capítulo 2: Conceitos fundamentais na implantação de coreografias de serviços.** Neste capítulo, discutimos alguns tópicos que fundamentam o trabalho desenvolvido e são essenciais para o seu entendimento, além de apresentarmos a terminologia usada nos capítulos seguintes. Expomos os principais conceitos sobre computação em nuvem e coreografias de serviços e buscamos identificar as principais necessidades na implantação de aplicações desenvolvidas usando essas tecnologias. Além disso, ilustramos os principais desafios na implantação de coreografias de serviços através de um exemplo do cenário trabalhado na tese.

**Capítulo 3: Trabalhos relacionados.** Neste capítulo descrevemos como a modelagem de restrições sobre coreografias e o provisionamento de recursos são tratados em outras abordagens. Listamos e descrevemos alguns dos trabalhos mais relevantes sobre estes tópicos, comparando-os com nossa proposta. Buscamos destacar os benefícios de usar nossa solução para o provisionamento eficiente de recursos na implantação de múltiplas coreografias de serviços.

**Parte III: Contribuição**

**Capítulo 4: Representação de múltiplas coreografias de serviços com restrições não-funcionais.** Este capítulo apresenta a formalização dos principais elementos relacionados ao problema discutido na tese. Além disso, propomos uma notação para modelagem de coreografias de serviços com restrições e comparamos essa notação com uma linguagem amplamente adotada na academia e na indústria. Através da notação proposta, mostramos como uma representação conjunta de múltiplas coreografias pode ser gerada para guiar o provisionamento de recursos.

**Capítulo 5: Modelagem de recursos.** Neste capítulo discutimos a multiplicidade de tipos de recursos e como essa propriedade pode ser usada para satisfazer res-

trições na implantação de coreografias de serviços. Mostramos como os recursos são modelados em nossa abordagem e apresentamos uma técnica utilizada para agregar a representação de recursos semelhantes.

**Capítulo 6: Síntese de recursos.** Neste capítulo apresentamos nossa solução para o provisionamento de recursos. Discutimos o tratamento do problema de estimativa e seleção de recursos. Além disso, apresentamos estratégias para contornar os casos onde não é possível implantar atender a demanda de requisições usando apenas uma instância de cada serviço. Também discutimos possíveis mudanças em tempo de execução e apresentamos uma estratégia para lidar com a variabilidade na carga de requisições, permitindo a encenação da coreografia da maneira requisitada.

**Capítulo 7: Arquitetura e implementação.** Este capítulo propõe uma arquitetura para implementação das estratégias propostas como nossa contribuição. Discutimos os cenários em que essa arquitetura pode ser utilizada e como seus componentes foram implementados em um protótipo usado para avaliar as estratégias desenvolvidas.

**Parte IV: Validação e conclusão**

**Capítulo 8: Avaliação experimental.** Neste capítulo apresentamos alguns experimentos realizados com o objetivo de avaliar a eficácia e eficiência das técnicas apresentadas. Inicialmente, discutimos alguns experimentos que comprovam a factibilidade de uso da proposta durante a implantação e encenação de coreografias de serviços. Em seguida, descrevemos a avaliação da proposição comparando-a com outras estratégias.

**Capítulo 9: Trabalhos futuros.** Neste capítulo discutimos perspectivas futuras para o aprimoramento da abordagem proposta nessa tese, buscando identificar suas principais limitações.

**Capítulo 10: Considerações finais.** Este capítulo conclui o trabalho apresentado nessa tese, resumindo suas contribuições.

# Parte II

# Estado da arte

CAPÍTULO 2

# Conceitos fundamentais na implantação de coreografias de serviços

Este capítulo versa sobre alguns conceitos relevantes para o desenvolvimento da abordagem proposta. Como forma de facilitar o entendimento do trabalho e contextualizar os termos utilizados, inicialmente apresentamos a terminologia adotada neste trabalho. Esta terminologia foi apresentada em [109], e estendida aqui.

Dentre os conceitos discutidos, inicialmente são apresentadas as principais características e desafios de sistemas baseados em nuvem. Em especial, são descritas as principais atividades relacionadas ao gerenciamento de recursos neste tipo de ambiente, destacando quais estão relacionadas à contribuição. Em seguida, são abordados os tópicos de modelagem e implantação de coreografias de serviços e serão discutidas algumas definições sobre restrições não-funcionais na implantação de serviços, destacando as particularidades dessas definições no contexto de composições de serviços.

Finalizando o capítulo, é descrito um exemplo de cenário que visa destacar a necessidade de expressar restrições não-funcionais na implantação de coreografias de serviços, e de considerar o compartilhamento de serviços entre elas. Com base neste exemplo, discutimos alguns dos principais desafios encontrados no problema tratado nesta tese.

O objetivo deste capítulo não é apresentar uma descrição profunda de todas as soluções existentes nas áreas discutidas, mas propiciar uma breve introdução aos conceitos usados na tese.

## 2.1 Terminologia

A terminologia adotada nesta tese usa a definição padrão de ***aplicação***, que consiste em um programa de computador desenvolvido com um propósito específico, como por exemplo, uma aplicação para agendamento de consultas médicas no sistema público de saúde. Uma aplicação pode ser implementada como um único ser-

viço ou ser uma composição de dois ou mais serviços. Para ***serviço*** adotamos a mesma definição de serviço *Web* [22], segundo a qual um serviço é uma entidade de *software* autocontida, cujas interfaces e ligações são definidas, descritas e localizadas por artefatos que seguem especificações como *Simple Object Access Protocol* (SOAP) [119] e *Representational State Transfer* (REST) [91].

Neste trabalho assumimos que todos os serviços são *stateless*, seguindo o padrão arquitetural REST. Esta decisão foi tomada visando vantagens como *i)* maior confiabilidade, uma vez que falhas são mais facilmente recuperadas; *ii)* maior escalabilidade, dado que não há necessidade de manter o estado das solicitações, permitindo que os serviços não necessitem de recursos além daqueles relacionados ao processamento das requisições; *iii)* requisições auto-contidas, pois a mensagem da chamada a uma operação contém todas as informações necessárias para processar a solicitação; e *iv)* simplificar o desenvolvimento da abordagem.

Ignoramos o custo associado à utilização dos serviços, como pagamento de licenças ou aluguel de software, de forma que a natureza dos serviços utilizados e a quantidade de instâncias de serviço envolvidas não tem impacto sobre o custo de execução de uma composição. Outra simplificação adotada foi abstrair a arquitetura de um serviço como sendo uma única camada cujos elementos não são explorados individualmente, apesar da possibilidade da implantação de serviços ser realizada de maneira mais precisa, caso o gerenciamento de recursos seja realizado separadamente para cada camada da arquitetura [38, 118, 248]. Consideramos esse tratamento como trabalho futuro.

Apesar do termo “serviço” ser amplamente usado em sistemas baseados em nuvem para se referir a qualquer componente virtualizado (inclusive componentes de infraestrutura), abandonamos esse significado para evitar qualquer ambiguidade entre serviços de nuvem e serviços (*Web*) que são elementos que compõem coreografias. Uma vez que o escopo desta tese se limita ao modelo de infraestrutura como serviço (*Infrastructure as a Service* – IaaS), usamos o termo ***recurso*** para referir aos serviços de nuvem que representam recursos virtualizados (VMs) em provedores de IaaS, e usamos o termo ***serviço*** para referir aos elementos que compõem coreografias; salvo algumas exceções em que o uso do termo ***serviço de nuvem*** é explicitamente adotado para evitar mal entendimento.

Um serviço (*Web*) implementa uma ou mais operações, devendo ser capaz de interagir com outros serviços ou aplicações através da troca de mensagens XML utilizando os protocolos de comunicação padrão atualmente disponíveis na *Internet*. Uma ***operação***, por sua vez, define alguma tarefa executada pelo serviço, que requer uma quantidade de poder computacional para ser processada. Cada operação se refere a um ***papel*** usado na definição de coreografias, o qual constitui uma função cuja

responsabilidade é assumida pelo serviço que a executa, que define as ações e a interação deste serviço com os demais serviços na composição.

Geralmente, os serviços que compõem uma coreografia podem ser descritos de maneira abstrata, devido ao uso do papel desempenhado pelo serviço na interação. Estes papéis podem então ser atribuídos a responsáveis usando entidades concretas, isto é, identificando uma implementação compatível para cada serviço. A abstração de serviços é particularmente importante em ambientes com múltiplas nuvens, uma vez que a especificação, usando serviços concretos, pode restringir de maneira excessiva a definição da composição e porque certas implementações podem ser específicas para um provedor de nuvem ou tecnologia. Apesar disso, devido à alta complexidade de propor uma abordagem que também contemple a seleção de serviços, consideramos que este aspecto constitui um trabalho futuro e, no contexto desta tese, assumimos que a especificação de coreografias é realizada usando entidades concretas.

Assumimos também que informações sobre serviços são gerenciadas por um ou mais ***catálogos de serviços***, comumente conhecidos como *brokers* de serviços [168]. Cada catálogo classifica serviços em duas categorias:

1. ***Serviço implantável***: descreve a implementação de um serviço. Para cada serviço nessa categoria há um conjunto de informações necessárias à implantação de uma instância deste serviço (por exemplo seu URI, protocolo de comunicação usado, *etc.*) e atributos das operações por ele implementadas (por exemplo, papéis executados, número de instruções do programa, *etc.*).
2. ***Serviço legado***: descreve serviços de terceiros que já foram previamente implantados, e para os quais não há possibilidade de interferência. Neste caso assumimos que o catálogo disponibiliza informações sobre as propriedades não-funcionais garantidas pelo serviço, assim como dados sobre seu ambiente de execução.

***Usuário*** refere-se às pessoas responsáveis pela composição da aplicação e sua manutenção, o que inclui o gerenciamento de recursos. Essas atividades devem ser realizadas de forma a fazer com que os requisitos funcionais e não-funcionais para uma classe de clientes sejam satisfeitos durante a execução da aplicação. Outra função realizada pelo usuário é decidir quais alternativas devem ser adotadas para adaptação da alocação de recursos, com base em mudanças nas condições do sistema e nas expectativas dos clientes. O ***cliente***, ou usuário final, é uma entidade que interage com essa aplicação.

Por último, temos um ***sistema*** que é um conjunto de componentes interdependentes que interagem entre si, formando um conjunto integrado. Usamos este termo para referir ao conjunto das aplicações gerenciadas, juntamente com os componentes que constituem nossa solução.

## 2.2 Computação em nuvem

Computação em nuvem (*Cloud Computing*) é um modelo que permite acesso a recursos computacionais sob medida e sob demanda, de maneira similar a recursos como água e eletricidade, o que leva esse modelo a ser chamado também de computação utilitária (*Utility Computing*). Este paradigma possibilita acesso, através de rede, a um conjunto de recursos computacionais (rede, armazenamento, processamento, plataformas de desenvolvimento e aplicações), que podem ser rapidamente aprovisionados e liberados com um mínimo esforço de gerenciamento e interação por parte do provedor [178].

Uma nuvem também pode ser definida como um sistema paralelo e distribuído que consiste em uma coleção de componentes virtualizados e interconectados, que são provisionados dinamicamente e apresentados como um ou mais recursos computacionais unificados [50].

Novas organizações sem o alto capital necessário para criar sua própria infraestrutura, utilizam este modelo para ter acesso a um vasto conjunto de recursos. Além disso, empresas como Amazon [11] logo perceberam a vantagem de oferecer o seu poder de computação excedente na forma de serviços em nuvem, pois com isso poderiam continuar a superdimensionar sua infraestrutura para atender aos picos de demanda, enquanto proveem sua capacidade excedente para os usuários na forma de recursos virtualizados. Além destes dois cenários, uma alternativa bastante promissora é a utilização de uma abordagem híbrida, na qual a organização possui sua própria infraestrutura, mas utiliza recursos de terceiros em períodos de grande demanda, o que é conhecido como *cloud bursting* [151, 137]. Neste caso surge a necessidade de maximizar o uso dos recursos privados por questões de segurança e economia.

Entre os principais fatores que contribuíram para a popularização de computação em nuvem está a diminuição de custo de *hardware*, aliada a um aumento no poder de computação e capacidade de armazenamento. Além disso, paralelamente tem havido um crescimento exponencial no volume de dados e uma ampla adoção de aplicações e serviços na *Web*. Nesta nova realidade, as corporações se esforçam para disponibilizar plataformas de nuvem mais poderosas, confiáveis e financeiramente atrativas devido à crescente ascensão de novos serviços que apresentam requisitos rigorosos. Dessa forma, há um crescente interesse na oferta de propriedades não-funcionais. Essa necessidade existe porque a utilização de um *software* é determinada não só pelas suas características funcionais, mas também por suas propriedades não-funcionais, sendo que algumas funcionalidades podem não ser úteis se certos atributos não-funcionais não forem oferecidos [61].

Apesar da importância dos aspectos não-funcionais, atualmente há garantias

bem reduzidas sobre atributos dessa categoria em serviços em nuvem, sendo que o foco maior ocorre em desempenho e disponibilidade [32]. A oferta de qualidade de serviço (*Quality of Service* – QoS) em computação em nuvem é um dos principais desafios desta área [40]. Essa necessidade se dá principalmente porque serviços providos em nuvem devem ser altamente confiáveis, escaláveis e autônomos. O próprio modelo de computação em nuvem já prevê que a alocação de recursos oferecida neste ambiente possa ser dinamicamente redimensionada para se ajustar a cargas variáveis de utilização, o que permite obter uma utilização ótima dos recursos físicos [256].

A seguir são apresentadas as principais características de ambientes de computação em nuvem.

### 2.2.1 Características

As principais características que definem computação em nuvem são apresentadas abaixo:

**Auto-atendimento sob demanda:** o usuário pode obter acesso ao conjunto de recursos no momento que necessitar, sem a necessidade de interação humana com o provedor. Isto é feito através de interfaces disponíveis na *Web* ou através de APIs que podem ser utilizadas programaticamente.

**Independência de localização:** os recursos são acessados por meio da *Internet*, utilizando diferentes plataformas (como *desktop*, *tablet* e *smartphone*) de forma ubíqua.

**Abstração de recursos:** os recursos providos são utilizados simultaneamente por múltiplos usuários através de virtualização, com diferentes recursos físicos e virtuais atribuídos e reatribuídos de acordo com a demanda dos usuários. O resultado é que recursos físicos tornam-se "invisíveis" para os usuários, que não possuem controle ou conhecimento da localização, formação e origem desses recursos. Isso diminui os riscos de negócio para os usuários deste paradigma, além de eliminar seus gastos com manutenção de recursos.

**Elasticidade:** o usuário tem acesso à quantidade de recursos que precisar, sem a necessidade que contratos rigorosos sejam preestabelecidos, podendo liberá-los assim que terminar o uso. A elasticidade garante a escalabilidade, o que significa que a alocação de recursos pode ser redimensionada ascendentemente para demandas de pico, e descendentemente para demandas mais leves.

**Serviço medido:** a infraestrutura de nuvem é capaz de usar mecanismos adequados para medir o uso dos recursos para cada usuário, que paga somente aquilo que consome.

Computação em nuvem é considerada como uma generalização de computação em grade (*Grid Computing*) [71, 94, 95]. Uma grade computacional pode ser

definida como uma infra-estrutura de *software* capaz de interligar e gerenciar diversos recursos computacionais (capacidade de processamento, dispositivos de armazenamento, instrumentos científicos, *etc.*), possivelmente distribuídos por uma grande área geográfica, de maneira a oferecer ao usuário acesso transparente a tais recursos, independente da localização dos mesmos [106]. Os paradigmas de grade e nuvem são similares no sentido de utilizarem recursos distribuídos para oferecer serviços. Contudo, computação em nuvem pode ser considerada um avanço por facilitar o compartilhamento e provisionamento dinâmico de recursos [282].

Além de computação em grade, computação em nuvem tem como base outras tecnologias:

- **Web 2.0:** Consiste no segundo estágio de desenvolvimento da *World Wide Web*, caracterizado principalmente pelo aumento na quantidade de páginas *Web* dinâmicas ao invés de páginas estáticas, maior presença de conteúdo gerado pelos clientes e crescimento de mídias sociais. Essa evolução permite o oferecimento de serviços pela *Internet*, disponibilizando via *Web* aplicações que antes existiam somente como programas para *desktop* [164].
- **Virtualização:** É uma técnica que possibilita a representação baseada em *software* (ou virtual) de um elemento físico como, por exemplo, servidores, armazenamento e rede. Em computação em nuvem este conceito possibilita que recursos virtuais sejam usados da mesma forma que os recursos reais equivalentes, permitindo o particionamento de um recurso físico para vários usuários simultaneamente através do uso de máquinas virtuais.
  representação baseada em *software* (ou virtual) de algo, em vez de um processo físico
- **Conteinerização:** Recentemente, contêineres têm surgido como uma alternativa mais leve à virtualização usando máquinas virtuais. Conteinerização é uma técnica de virtualização no nível do sistema operacional, usada para implantar e executar aplicações distribuídas sem a necessidade de criar uma máquina virtual inteira para cada aplicação. Em vez disso, múltiplos sistemas isolados, chamados contêineres, são executados em uma máquina, compartilhando um único *kernel* [203]. Com isso, contêineres são mais eficientes que máquinas virtuais.
- **Computação autonômica:** Um sistema computacional autonômico controla as funcionalidades de sistemas e aplicações com o mínimo de intervenção humana, da mesma forma que o sistema nervoso autônomo regula o sistema corporal, sem intervenção do indivíduo. O objetivo da computação autonômica é criar sistemas com maior grau de independência, capazes de implementar funcionalidades complexas enquanto mantém esta complexidade transparente ao

usuário. Como resultado, profissionais de TI podem focar em tarefas que agregam mais valor ao negócio [67].

A Figura 2.1 apresenta a relação de computação em nuvem com estes domínios. A *Web* 2.0 cobre quase todo o espectro de aplicações orientadas a serviços. Se baseando nela, ou seja, sobrepondo esta tecnologia está a computação em nuvem que geralmente oferece larga escala. Essas tecnologias, por sua vez, são baseadas em computação autonômica e virtualização, que podem ser utilizadas para desenvolver soluções orientadas a aplicações ou orientadas a serviços, com escala que dependerá da solução desenvolvida. Por outro lado, a conteinerização também representa uma tecnologia baseada em virtualização mas constitui uma alternativa de maior escala. Assim como computação em nuvem, computação em grade sobrepõe, ou seja, se baseia em todas as demais tecnologias mas é voltada para aplicações convencionais. Subpondo quase totalmente as tecnologias descritas estão as tecnologias de sistemas distribuídos em geral.

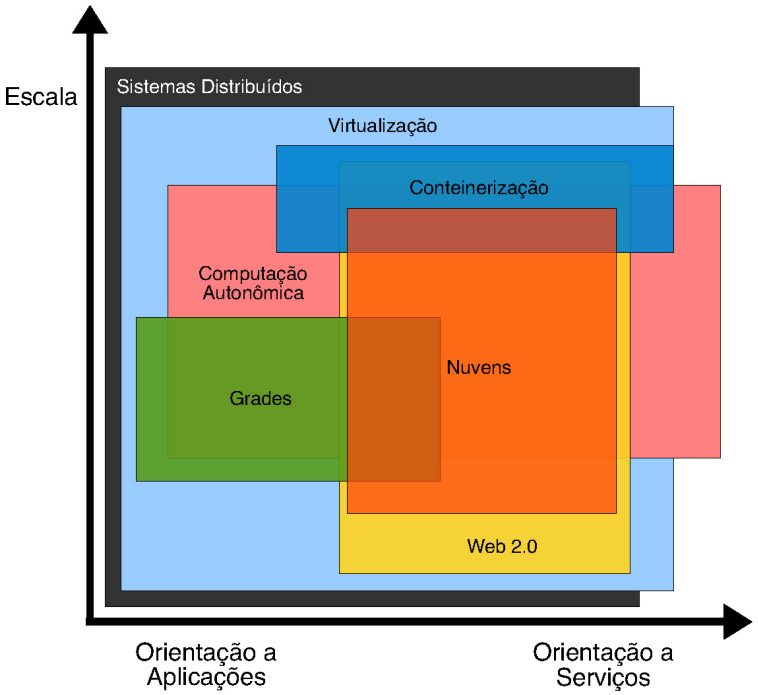



**Figura 2.1:** *Relacionamento de nuvens com outras tecnologias (Adaptada de [96]).*

### 2.2.2 Categorias de serviços

Mais comumente, um provedor de nuvem oferece recursos de *hardware* (processamento, armazenamento, rede, *etc.*), mas é possível encontrar outros tipos de recurso, como serviços de *software*, aplicações, APIs e ferramentas de desenvolvimento.

De acordo com a categoria de serviço que oferece, os provedores de nuvem são mais comumente classificados em três categorias principais:

**Infraestrutura como serviço (*Infrastructure as a Service* – IaaS):** usuários usam diretamente a infraestrutura provida. Virtualização é extensivamente utilizada para integrar/decompor recursos físicos para atender a demanda dos usuários. Ex.: *Amazon Web Services* (AWS) [11] e Nimbus [147].

**Plataforma como serviço (*Platform as a Service* – PaaS):** permite aos usuários desenvolverem serviços e aplicações que irão executar na nuvem, utilizando linguagens de programação, bibliotecas, serviços e ferramentas oferecidas pelo provedor. O usuário não tem que gerenciar elementos da infraestrutura de nuvem, como rede, servidores, sistemas operacionais, ou armazenamento, mas tem controle sobre as aplicações implantadas e a configuração do ambiente de hospedagem [178]. Ex.: Google App Engine [110] e Microsoft Azure[1] [182].

***Software* como serviço (*Software as a Service* – SaaS):** o fornecedor do serviço hospeda as aplicações, de modo que não é preciso instalá-las, gerenciá-las, ou comprar *hardware* para sua execução. Com isso, as ações que devem ser realizadas pelo usuário se limitam a acessar o serviço e proceder com sua utilização. Ex.: Google Drive [112] e SAP Business ByDesign [237].

Esta categorização é geralmente construída seguindo uma arquitetura de camadas, conforme indicado na Figura 2.2. Diferentes categorias podem ser oferecidas por um mesmo provedor de nuvem e, como pode ser visto, é possível ao usuário acessar qualquer uma das camadas. Porém, é importante ressaltar que o oferecimento conjunto de múltiplas camadas dessa arquitetura por um mesmo provedor de nuvem não representa um padrão, uma vez que, conforme exemplificado acima, há provedores que são específicos de uma certa categoria.

Com a popularização de computação em nuvem, outras categorias de serviço passaram a ser oferecidas na *Web* utilizando este modelo de negócio. Com isso, atualmente, o termo comumente utilizado para representar de forma genérica as categorias de serviços em nuvem é: tudo como serviço (*Anything as a Service* – XaaS). Além dos serviços já citados, este termo engloba também a provisão de comunicação, infraestrutura de rede, monitoramento e objetos como serviços [5, 207].

Ao mover uma aplicação para ambientes de nuvem há muitas questões a considerar. Como exemplo, algumas organizações estão mais interessadas em reduzir o custo de execução da aplicação, enquanto outras podem preferir alta confiabilidade e segurança. Consequentemente, existem diferentes tipos de nuvem, cada um com seus

[1] Apesar de inicialmente o Microsoft Azure (Windows Azure) ter sido desenvolvido visando PaaS, atualmente IaaS é um dos principais focos deste provedor.

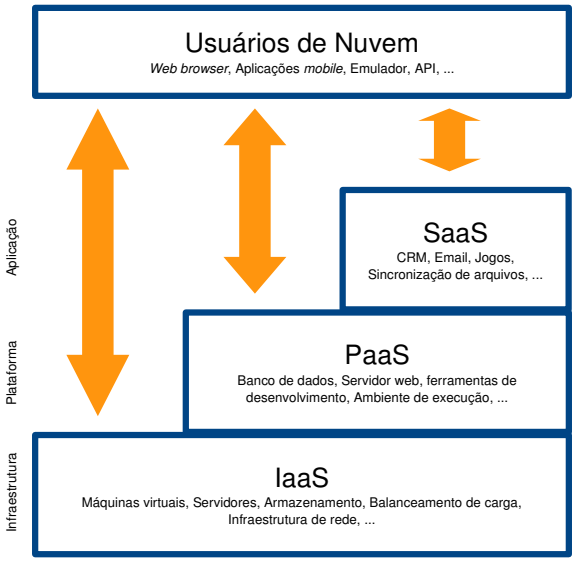



**Figura 2.2:** *Categorização e relacionamento dos serviços em nuvem.*

próprios benefícios e desvantagens:

**Nuvem privada (*private cloud*):** a infraestrutura é mantida e gerenciada pela própria organização, visando maximizar o uso de seus recursos e evitar problemas como transferência de dados e segurança.

**Nuvem pública (*public cloud*):** este é o modelo dominante, no qual a infraestrutura de uma organização é utilizada pelo público em geral, sendo que as políticas de uso e cobrança são definidas pelo provedor de recursos.

**Nuvem híbrida (*hybrid cloud*):** é a combinação dos dois modelos anteriores, onde uma organização usa sua própria infraestrutura, geralmente para armazenamento de dados e execução de serviços essenciais, e usa os recursos de uma nuvem pública para aumentar sua produtividade ou obter um serviço que não possui.

O trabalho apresentado nesta tese tem como escopo nuvens híbridas, com múltiplos provedores de nuvens públicas, focando na camada de IaaS.

### 2.2.3 Gerenciamento de recursos em nuvem

O objetivo desta seção é prover uma análise das atividades desempenhadas pelo usuário do ambiente de nuvem no controle dos recursos virtualizados. Investigamos também as principais ferramentas disponíveis para realizar essas atividades. Embora as mesmas atividades façam sentido do ponto de vista do provedor de nuvem, ao gerenciar os recursos físicos, não levamos em consideração os aspectos que são relevantes somente nesse contexto (como por exemplo, a economia de energia [48]), uma vez que nosso objetivo é isolar o ponto de vista do usuário de serviços em nuvem.

Conforme ilustrado na Figura 2.3, o gerenciamento de recursos envolve atividades desde a estimativa até o monitoramento de recursos, sendo que as principais atividades nessa sequência geralmente são tratadas como sendo uma única atividade, identificada como provisionamento de recursos.

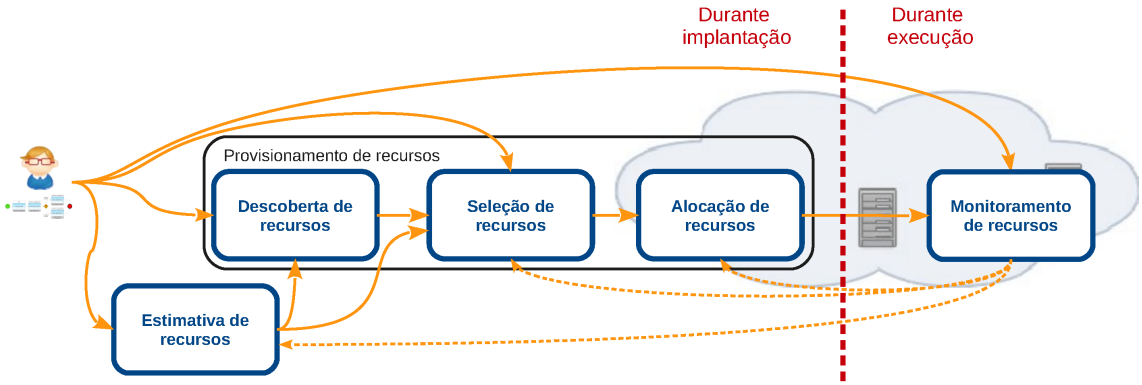



**Figura 2.3:** *Atividades envolvidas no gerenciamento de recursos em nuvem e como elas são acionadas.*

A primeira atividade, estimativa de recursos, visa identificar a capacidade ideal de recursos e o custo esperado ao realizar a implantação da aplicação. Essa atividade deve levar em consideração as características funcionais da aplicação, assim como as restrições não-funcionais impostas sobre ela. As ferramentas atualmente existentes só auxiliam a calcular o custo estimado para um provedor específico, como por exemplo *Microsoft Azure Princing Calculator* [25] ou *AWS Total Cost of Ownership (TCO) Calculator* [14], mas a capacidade e quantidade de recursos necessários devem ser fornecidos como parâmetros. Mesmo as soluções que permitem análise semelhante considerando múltiplos provedores, como PlanForCloud [226] e Aotearoa [17], limitam-se a estimar o custo em provedores públicos comparando-o com o gasto necessário para uma infraestrutura privada, requerendo que o usuário defina possíveis alternativas, objetivos e critérios de escolha. Além disso, nenhuma dessas soluções auxilia na tarefa de decidir a capacidade dos recursos para implantar a aplicação.

Após estimar os recursos necessários, o provisionamento de recursos se inicia com a atividade de descoberta de recursos, que consiste em encontrar uma lista de recursos disponíveis nos ambientes considerados. A atividade seguinte, chamada de seleção de recursos, é o processo de escolher o melhor recurso da lista gerada pela descoberta de recursos, tendo como base as restrições impostas pelo usuário [243]. Neste caso, assumimos que o resultado da seleção é um conjunto de tipos de VM. Em nossa abordagem, o custo e a localização da VM são importantes aspectos na solução. Em virtude disso, incluímos estes atributos como parte da descrição do tipo de VM, diferente dos provedores de nuvem pública, que consideram apenas atributos de capacidade de *hardware* para classificar um tipo. Ao adotar essa estratégia, o custo e a

localização são usados para distinguir tipos. Em virtude disso, VMs com custos e localizações diferentes são consideradas como de tipos distintos, mesmo sendo do mesmo provedor e tendo capacidade idêntica de *hardware*. Dado os tipos de recurso selecionados, a última atividade no provisionamento de recursos é a alocação, que consiste na instanciação do tipo selecionado e instalação dos componentes de *software* que serão executados nele. Por simplicidade, ignoramos nessa atividade a seleção das imagens (*Virtual Appliances*) que serão usadas na implantação da aplicação, pois assumimos que elas são fornecidas como entrada ao fazer a alocação de recursos.

É importante ressaltar que na análise realizada nesta seção consideramos que os recursos são instanciados sob demanda, ou seja, não há um conjunto de instâncias de VM pré-criadas, o que justifica a inclusão da alocação de recursos como uma das etapas que necessariamente ocorrem no provisionamento de recursos. Uma alternativa a esta suposição seria considerar que há um conjunto de VMs pré-instanciadas, que é também considerado na atividade de seleção de recursos. Dessa forma, se o recurso selecionado já possuir uma instância alocada, a atividade de alocação de recursos é substituída pela atividade de escalonamento de recursos. Contudo, não incluímos o escalonamento de recursos em nossa análise porque esta atividade está fora do escopo do nosso trabalho.

Realizar a descoberta, a seleção e a alocação de recursos, de maneira completamente automatizada é um desafio devido à variedade de tecnologias envolvidas. Soluções da indústria que são específicas de provedor ou solução de nuvem, como *AWS Cloud Formation* [12], não são suficientes para solucionar completamente este problema, uma vez que são limitadas a um ambiente específico e são incapazes de integrar múltiplos provedores, o que é essencial para satisfazer um conjunto de restrições muito restrito. Em contrapartida, soluções da indústria para o gerenciamento de recursos em ambientes com múltiplas nuvens, como RightScale [227], se propõem a oferecer provisionamento e gerenciamento de infraestrutura sobre múltiplos provedores de IaaS usando *scripts* de automação. Contudo, a adaptação desses *scripts* para cenários específicos não é uma tarefa simples. APIs de abstração de ambientes de nuvem, como OpenStack [199], proveem uma maneira de desacoplar as dependências do provedor de nuvem, facilitando o desenvolvimento de aplicações multi-nuvem. Contudo, elas não resolvem o problema de integrar diferentes APIs de gerenciamento e tecnologias, como propomos em nossa abordagem.

Ferramentas para alocação de recursos como Puppet [217] e Chef [57] e arcabouços como Marionette Collective [216] e Spiceweasel [222] permitem o gerenciamento de configuração de forma mais complexa. Além disso, há ferramentas como Juju [141], que permitem a orquestração de *scripts* de gerenciamento de configuração. Contudo, essas abordagens baseadas em *scripts* são limitadas a tarefas de instalação e

configuração de componentes de *software* em VMs preexistentes. O provisionamento completamente automatizado, usando essas estratégias no desenvolvimento de aplicações complexas (como composição de serviços com dependências entre si), não é trivial, uma vez que requer a escrita e atualização dos *scripts* de integração.

A última atividade considerada é o monitoramento de recursos, que visa identificar, durante a execução das aplicações, falhas nos recursos alocados bem como coletar medições sobre o uso de recursos, como, por exemplo, percentual médio de uso da CPU. Essa atividade está diretamente relacionada à atividade de monitoramento dos serviços, que consiste em verificar dinamicamente os parâmetros de QoS relacionados às aplicações [6], como, por exemplo, tempo de resposta. Apesar dessa diferenciação, é preciso destacar que, além dos parâmetros que não são explicitamente relacionados a QoS, o monitoramento de recursos também pode ser usado para identificar parâmetros de QoS, como por exemplo nível de ociosidade dos recursos. Contudo, na Figura 2.3, e no texto que a segue, nos referimos somente ao monitoramento de recursos, que não considera a priori aspectos de QoS. Atualmente, muitos provedores de nuvem pública oferecem aos seus usuários a capacidade de monitorar seus recursos usando ferramentas de monitoramento disponíveis para CPU, armazenamento e rede. Não obstante, estas ferramentas geralmente são fortemente integradas com o provedor, como AWS CloudWatch [13], não permitindo o monitoramento de componentes implantados em outro provedor de nuvem. Para tal, existem soluções que são multiplataforma, como CloudHarmony [65] e RevealCloud [130], mas que oferecem apenas métricas preestabelecidas.

Conforme ilustrado na Figura 2.3, as atividades relacionadas ao gerenciamento de recursos são comumente executadas em sequência (com exceção de estimativa e descoberta de recursos, que podem ser realizadas em paralelo), sendo que cada atividade depende dos resultados obtidos nas atividades precedentes. Ao implantar uma aplicação, o usuário pode executar cada uma delas ou ignorar algumas atividades. Nesse caso, assume-se que os resultados esperados já foram previamente processados ou são conhecidos. Apenas as atividades de alocação e monitoramento de recursos devem necessariamente ser executadas na infraestrutura do provedor de nuvem, ao passo que as demais podem ser executadas completamente fora desse ambiente, apesar de algumas requererem interação com o provedor.

Nossa solução considera todas as atividades relacionadas ao gerenciamento de recursos em nuvem. Embora nossa contribuição mais significativa seja com relação à seleção de recursos, também incorporamos na solução aspectos relacionadas às outras atividades.

### 2.2.4 Principais desafios

Apesar da adoção cada vez mais ampla de computação em nuvem e da existência de diversos trabalhos que exploram este paradigma, há necessidades emergentes que constituem desafios a serem solucionados. Os principais são:

**(Auto-)Gerenciamento:** a elasticidade e abstração de uso dos recursos encontradas em um ambiente de nuvem requer gerenciamento eficiente. É necessário que as tarefas de manutenção sejam realizadas de forma autônoma [221], fazendo com que os usuários não precisem conhecer detalhes da infraestrutura e haja o mínimo de esforço por parte do provedor dos recursos.

**Gerenciamento de energia:** enquanto infraestruturas físicas, nuvens são essencialmente *datacenters* que requerem um grande volume de energia para se manter em operação. Reduzir o custo de energia pode ser uma alternativa para aumentar o lucro [102]. Contudo, manter a disponibilidade de recursos e, ao mesmo tempo, construir um modelo sustentável representa outro desafio.

**Privacidade e segurança:** migrar os dados de uma organização para recursos gerenciados por terceiros representa o principal obstáculo ao uso de nuvens. Há um elevado grau de preocupação com a privacidade e segurança das informações compartilhadas, sendo que o oferecimento de um ambiente seguro e confiável é um tema frequentemente discutido.

**Gerenciamento de dados:** com a geração contínua de um volume cada vez maior de dados, o gerenciamento se torna extremamente complexo, sobretudo ao se considerar recursos heterogêneos, como geralmente existe em nuvens.

**Interoperabilidade:** a comunicação e migração de aplicações e dados entre provedores de nuvem ainda não é possível na maioria dos cenários. Mecanismos que permitam esse intercâmbio constituem uma necessidade prioritária, principalmente em nuvens híbridas [137].

**Qualidade de serviço (QoS):** o oferecimento e controle de QoS em ambientes de nuvem representa um dos maiores desafios atuais, uma vez que há a necessidade de acesso amplo por diferentes aplicações e categorias de clientes. A falta de padronização e a alta heterogeneidade encontrada nesses ambientes constituem os principais problemas.

Apesar de nossa proposta estar mais fortemente relacionada ao oferecimento de qualidade de serviço (aliada à satisfação de demais restrições), também buscamos contribuir, mesmo que indiretamente, com outros desafios aqui listados. A modelagem de recursos abstrai detalhes da infraestrutura e guia as atividades do sistema, provendo um nível mais autonômico de gerenciamento. Os critérios de segurança relacionados aos atributos dos recursos podem ser tratados em nossa proposta como

restrições em uma aplicação. Além disso, a possibilidade de utilização de diferentes provedores de nuvem na implantação de aplicações favorece a interoperabilidade.

## 2.3 Coreografia de serviços

O desenvolvimento de sistemas orientados a serviços é complexo, principalmente devido à existência de serviços fortemente pervasivos e com níveis cada vez mais elevados de heterogeneidade em termos de paradigmas de interação, protocolos de comunicação e modelos de representação de dados [104]. Além disso, a rapidez com a qual as expectativas dos usuários e do ambiente evoluem exigem soluções cada vez mais autônomas e dinâmicas. Como consequência, novos paradigmas de composição que permitem adaptação dinâmica são constantemente investigados. Coreografia de serviços é uma forma de composição promissora e flexível, que provê uma metodologia para especificação adaptável do comportamento esperado dos serviços, sua colaboração e as mensagens trocadas [31].

A visão integrada de serviços passou a ser o modelo de referência para a *Internet* [205] e se tornará ainda mais evidente na chamada *Internet* do Futuro, resultante da evolução da *Internet* atual com a união e cooperação da *Internet* de Conteúdos [70], *Internet* de serviços [204] e *Internet* das coisas [21]. Coreografias de serviços são uma alternativa para satisfazer as necessidades emergentes nesta nova realidade. A popularização de soluções desenvolvidas por meio de cooperação entre serviços, no entanto, destacou alguns problemas que não eram facilmente perceptíveis nos esforços de integração anteriores, pois dificilmente alcançavam a escala que os sistemas possuem agora [258].

Coreografias constituem um paradigma de composição na qual o protocolo de interação entre os serviços participantes é definido em uma perspectiva global. Cada papel na coreografia especifica o comportamento esperado dos participantes, que irão executá-lo através da sequência e periodicidade das mensagens que eles irão produzir e consumir [218]. Em tempo de execução cada participante na coreografia executa seu papel de acordo com o comportamento de outros participantes [211], o que é conhecido como encenação da coreografia.

Esta forma de criar processos de negócios é comumente confundida com orquestração de serviços, uma estratégia complementar na qual um único agente é responsável por controlar e coordenar as interações. Como ilustrado na metáfora da Figura 2.4, diferente de orquestrações, que tem um fluxo de controle centralizado, coreografias envolvem um arcabouço de composição de serviços compartilhado, em que somente as funcionalidades dos participantes e as mensagens associadas passadas são descritas [27]. Mais especificamente, coreografias se baseiam em um estilo de comu-

nicação par-a-par, ou seja, cada serviço envolvido na composição sabe exatamente quando executar suas operações e com quem interagir, sem um controle centralizado [27]. Neste sentido, uma coreografia pode ser vista como um contrato entre um conjunto de serviços que rege como uma dada colaboração deve ocorrer.

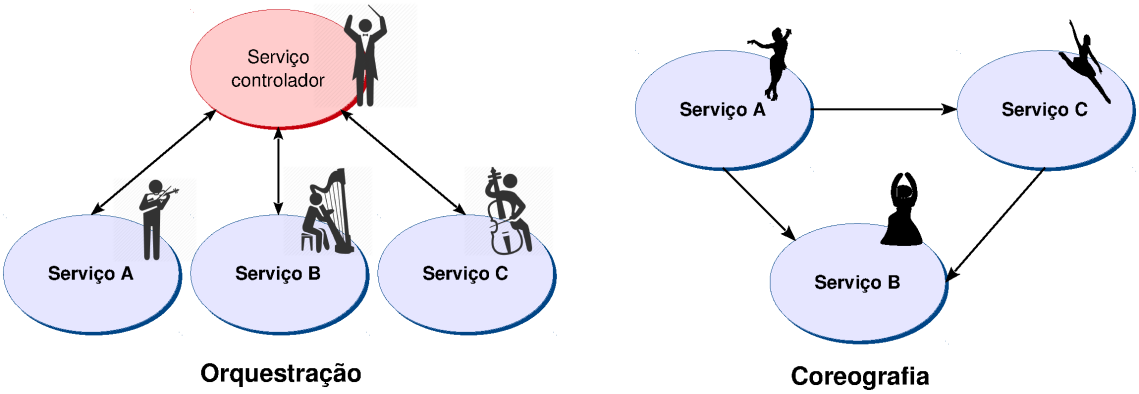



**Figura 2.4:** *Diferentes formas de composição de serviços.*

Embora o paradigma de coreografia pressuponha um modelo global de interação entre participantes, não é necessário que a implementação de cada serviço participante tenha conhecimento do fluxo de negócio completo da coreografia. É suficiente que cada serviço tenha conhecimento de sua parte no fluxo. Assim, cada participante da coreografia pode ter o seu comportamento modelado usando uma linguagem de orquestração. Assim sendo, uma coreografia pode também ser modelada como um conjunto de orquestrações distribuídas que interagem entre si, de forma que apenas os orquestradores precisam estar cientes de condições impostas pela coreografia [215].

Cada papel em uma composição pode ser executado por um conjunto de serviços alternativos que podem servir como substitutos um para o outro. Dessa forma, coreografias têm emergido como uma alternativa promissora no contexto da internet do futuro, na qual milhões de serviços, coisas e recursos participam em cenários complexos e de grande escala [135]. Este interesse também é aumentado pelo fato de que esta é uma alternativa promissora para formar sistemas complexos, dada a ampla aceitação do paradigma de arquitetura orientada a serviços.

### 2.3.1 Modelagem de coreografias

As linguagens para modelagem de coreografia de serviços podem ser categorizadas utilizando dois critérios [73]: linguagens independentes de implementação e linguagens específicas de implementação. As linguagens independentes de implementação são utilizadas principalmente para descrever processos da perspectiva de negócios, sem necessariamente expressar aspectos de como isso é implementado. Por

outro lado, as linguagens específicas de implementação definem os formatos de mensagens concretas ou protocolos de comunicação utilizados.

Existem duas abordagens de modelagem para linguagens de coreografia de serviços [27]: modelos de interação e modelos de interconexão. Os modelos de interação usam interações atômicas como blocos de construção básicos e os fluxos de dados e de controle estão baseados nas dependências entre essas interações de uma perspectiva global. Os modelos de interconexão, por sua vez, definem o fluxo de controle por participante ou por papel do participante, e as respectivas atividades de envio e recepção são conectadas usando fluxos de mensagens, representando deste modo as interações.

A expressividade em cada uma dessas duas abordagens de modelagem é diferente [73]. Por um lado, modelos de interconexão permitem a modelagem de processos incompatíveis ou mesmo processos para os quais nenhum parceiro existe [266]. Como exemplo, o serviço $A$ aguarda uma mensagem mas o serviço $B$ nunca envia esta mensagem. Por outro lado, modelos de interação podem representar restrições que precisam ser satisfeitas adicionando comunicação adicional que não corresponde ao modelo de interação original [280]. Por exemplo, se o serviço $A$ envia uma mensagem ao serviço $B$ e o modelo de interação exige que o serviço $C$ subsequentemente envie uma mensagem para o serviço $D$, o serviço $C$ tem que saber quando o serviço $B$ recebeu a mensagem.

A Figura 2.5 mostra a categorização das principais linguagens para modelagem de coreografia segundo os critérios e as abordagens citados. A seguir, essas linguagens são descritas resumidamente.

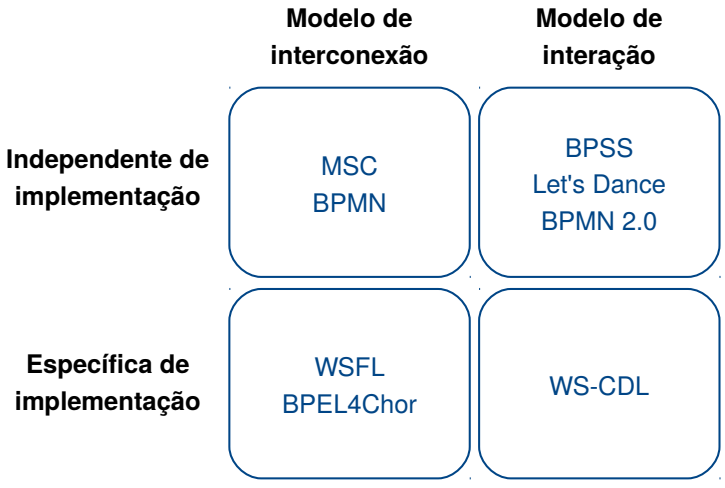



**Figura 2.5:** *Categorização de linguagens para modelagem de coreografias de serviços (Adaptada de [86]).*

*Message Sequence Chart* (MSC)[136] é uma notação gráfica para representar cenários de execução, modelando exemplos de fluxos convencionais ou alternativos

da execução do sistema. Essa linguagem é mais adequada para modelar sequências de interações simples ao invés de coreografias complexas, já que não suporta fluxos condicionais, fluxos paralelos e iterações [73].

*Business Process Modeling Notation* (BPMN) [198] é uma linguagem gráfica proposta pelo *Object Management Group* (OMG) para modelar processos. Ela oferece uma notação em alto nível que permite o usuário focar no papel dos serviços na composição, sem ter que se preocupar em como esse papel é implementado. Esta linguagem realiza a distinção explícita entre o fluxo de controle e o fluxo de mensagens. Todas as atividades conectadas por meio de fluxos de controle pertencem ao mesmo processo e o fluxo de mensagens é utilizado para interconectar processos diferentes. Uma desvantagem de BPMN 1.x é que ela permite apenas a especificação de coreografias através da interconexão de interfaces (usando diagramas de colaboração), sem descrever como a interação de fato ocorre. Essa deficiência foi corrigida na versão 2.0 da linguagem através de diagramas que modelam coreografias como entidades de primeira classe. Essa linguagem passou a ser o padrão *de facto* para modelar graficamente processos de negócio [73]. Por esse motivo, discutimos mais detalhes de BPMN 2.0 adiante.

A linguagem *Web Services Flow Language* (WSFL) [165] foi proposta pela IBM e teve como objetivo contribuir para um futuro padrão nesta área. Esta linguagem possui dois tipos de descrições. O primeiro, modelos de fluxo (*flow Model*) visa descrever fluxos de trabalho que interagem com outros fluxos, sendo as interações modeladas como serviços *Web*, cuja implementação é especificada usando uma descrição WSDL. O outro tipo, modelo global (*global model*) permite indicar ligações entre as operações de processos que sejam definidos por WSDL. Dessa forma, WSFL pode ser usada apenas para definir a estrutura da coreografia.

*Business Process Execution Language* (BPEL) [140] é a linguagem mais comumente utilizada para implementar processos de negócios baseados em serviços *Web* [74]. BPEL permite especificar a ordenação das mensagens trocadas entre os serviços, mas seu foco é descrever o comportamento da comunicação para um serviço individual. Como consequência, essa linguagem não pode ser usada para representar coreografias. Para contornar essa limitação, em [75] os autores propuseram extensões em BPEL para representação de coreografias, criando a linguagem BPEL4Chor. Em BPEL4Chor os processos BPEL abstratos são utilizados para descrições de comportamento entre participantes, as quais são unidas por meio de mensagens formando uma topologia de participantes.

A linguagem ebXML *Business Process Specification Schema* (BPSS) [64] visa descrever a colaboração entre dois participantes, proporcionando um conhecimento preciso acerca das mensagens trocadas, do seu sequenciamento e do estado final

das interações. Sua desvantagem é não permitir a interação de um número maior de participantes.

A linguagem Let's Dance [279] foi criada visando capturar a interação entre serviços sob uma perspectiva comportamental nas fases de análise e projeto de sistemas. Seu objetivo principal é abstrair completamente detalhes de implementação. Let's Dance suporta mais cenários de coreografias do que BPMN2, mas não é amplamente adotada na indústria.

A linguagem *Web Services Choreography Description Language* (WS-CDL) [146] é baseada em XML e descreve colaborações fim-a-fim modeladas como serviços *Web*. Esta linguagem define a composição de um ponto de vista global às várias entidades participantes, através do qual pode-se extrair as definições dos processos que cada participante deve implementar. Essa visão global tem a vantagem de permitir a separação do processo a ser seguido por uma organização individual, da definição da sequência segundo a qual ela trocará mensagens com as outras organizações. Contudo, WS-CDL é criticada por não separar de maneira apropriada o metamodelo da sintaxe, por oferecer suporte insuficiente a algumas categorias de cenários e por não ser facilmente compreensível [28].

Estas linguagens têm os seguintes elementos em comum [144]: *i)* mensagens: tipos e conteúdo das mensagens trocadas entre os participantes; *ii)* ordenação das mensagens: restrições de tempo para mensagens síncronas e assíncronas com pré/pós-condições associadas; *iii) endpoints*: participantes da coreografia que devem ser acessados com protocolos específicos. Porém, nenhuma delas possui sintaxe para modelagem de restrições não-funcionais sobre a coreografia. Em nossa solução, propomos uma representação para contornar essa limitação, que será apresentada no Capítulo 4. Antes, porém, discutimos maiores detalhes de BPMN 2.0 em virtude de sua ampla adoção na academia e na indústria.

#### 2.3.1.1 Modelagem de Coreografias com BPMN 2.0

Em BPMN 2.0, um processo de negócios é representado usando um diagrama de processo de negócio (*Business Process Diagram* – BPD), que é composto por um conjunto de atividades parcialmente ordenadas sendo executadas pelos participantes do processo.

Um BPD é essencialmente uma coleção de divisões identificadas como raias, de objetos e de fluxos de sequência e de mensagens. Raias representam os participantes do processo de negócios. Objetos podem ser eventos, tarefas ou conectores. Fluxos de sequência determinam a ordem de execução entre dois objetos em uma mesma raia, enquanto fluxos de mensagens representam mensagens trocadas entre dois objetos de raias diferentes.

A Figura 2.6 apresenta um subconjunto dos principais elementos em BPMN 2.0 para modelagem de coreografias de serviços.

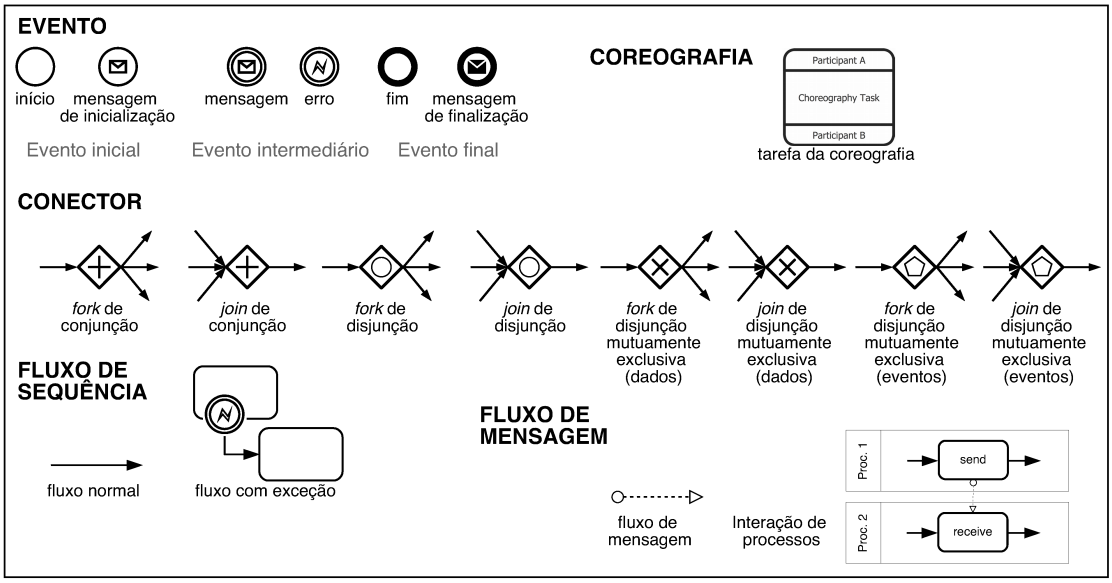



**Figura 2.6:** *Subconjunto do principais elementos BPMN 2.0 para modelagem de coreografias de serviços.*

Eventos podem representar o início de um processo (evento de inicialização), o fim do processo (evento de finalização) ou algo que deve acontecer durante o processo (evento intermediário). Há diferentes tipos de evento. Na figura, apresentamos eventos sem especial conotação (usados quando o modelador não especifica o tipo), eventos de mensagem (usados quando mensagens são usadas como gatilho do processo, para indicar envio ou recebimento de mensagens durante o processo, ou para indicar o fim do processo) e eventos de erro (podendo indicar que o fluxo dispara ou captura um erro).

Uma tarefa é uma atividade atômica que faz parte de uma coreografia. Cada tarefa é representada como um componente que apresenta três divisões. A divisão superior representa o participante que inicia a interação, a divisão inferior representa o participante que é o dono da tarefa e a divisão intermediária é usada para representar a tarefa propriamente dita.

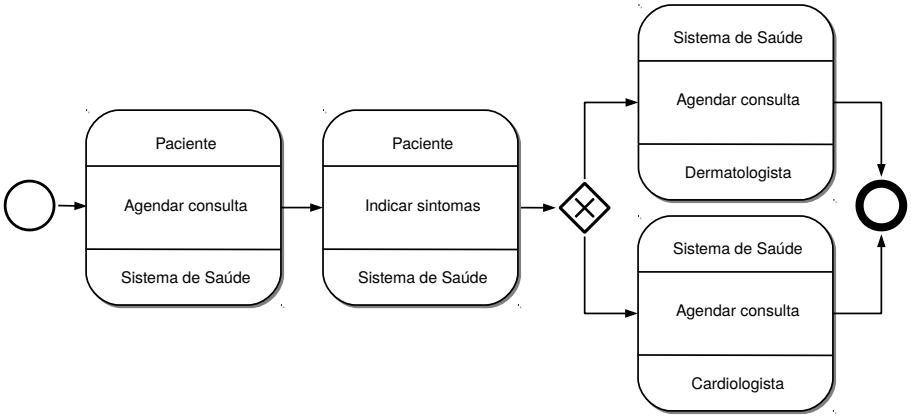
Conectores são usados para controlar fluxos de sequência. Um conector *fork* de conjunção é utilizado para criar fluxos paralelos, ao passo que um conector *join* de conjunção é usado para sincronizar fluxos paralelos.

Um conector *fork* de disjunção divide o fluxo de sequência e ativa um ou mais alternativas de acordo com uma condição. O conector *join* de disjunção aguarda que todos os ramos de entrada ativos sejam concluídos antes de acionar o fluxo de saída. O conector *fork* de disjunção mutuamente exclusiva divide o fluxo de sequência

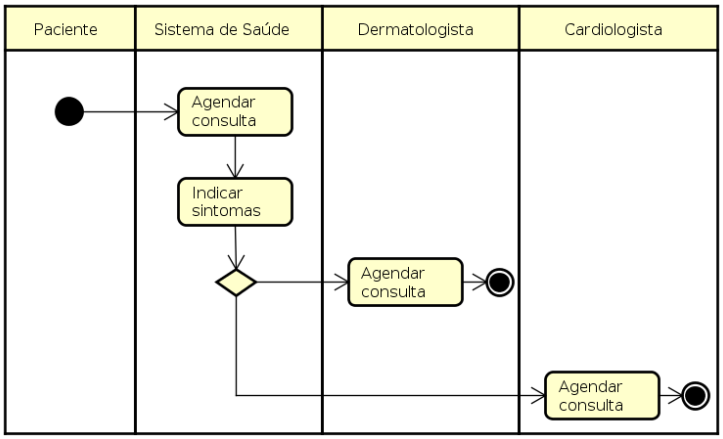
usando como referência o valor de dados ou a ocorrência de eventos, encaminhando-o para exatamente um dos ramos de saída. Por outro lado, o conector equivalente *join* aguarda a conclusão de um ramo de entrada antes de acionar o fluxo de saída.

Neste trabalho consideramos apenas o modelo de interação em BPMN 2.0. Dessa forma, os modelos discutidos no restante da tese apresentam apenas o fluxo de sequência, uma vez que o fluxo de mensagens é inferido pelo próprio diagrama da coreografia.

A Figura 2.7 apresenta um exemplo de coreografia modelada usando essa notação e a representação do mesmo processo em um diagrama de atividades UML.




(a)




(b)

**Figura 2.7:** *(a) Exemplo de coreografia modelada com BPMN 2.0. (b) Representação das tarefas da coreografia em um diagrama de atividades.*

Como pode ser visto, as setas no BPD, ao contrário do diagrama em UML, são usadas para representar unicamente a sequência de tarefas e não indicar como a interação ocorre. Essa informação está implícita na representação da tarefa.

### 2.3.2 Implantação de Coreografias

Enquanto a implantação de orquestrações de serviços baseadas em BPEL é bem consolidada, existindo vários ambientes de execução, coreografias são na maior parte das vezes consideradas artefatos de projeto em vez de artefatos de implementação [135]. Para contornar este problema, muitos trabalhos na literatura têm proposto a composição automática de serviços com base em coreografias [92, 214, 245, 249].

A ideia comum por trás dessas abordagens é utilizar uma especificação de requisitos que a coreografia tem que satisfazer juntamente com uma especificação do comportamento dos serviços participantes na coreografia. Através destas duas especificações, a descrição dos serviços é gerada visando satisfazer os requisitos. Além dessas propostas, há outras que sugerem transformação entre modelos e linguagens [125, 230] para permitir a geração de interfaces através de coreografias.

A implantação de composições de serviços deve lidar com diversos desafios, sendo que o principal está relacionado ao provisionamento de recursos. Ao implantar uma composição, é preciso decidir qual provedor de recursos é mais adequado para cada serviço ainda não implantado, levando em consideração a relação destes com os serviços de terceiros já em execução. Os questionamentos que precisam ser respondidos são:

- Para cada serviço, qual a capacidade de *hardware* que o recurso selecionado deve possuir? Em outras palavras, que tipo de VM deve ser selecionado para cada serviço de forma a atender sua demanda?
- Quantas instâncias do tipo de recurso selecionado precisam ser criadas para atender as restrições impostas sobre as composições?
- Onde alocar cada uma das instâncias de recurso? Incluído nessa última decisão está o problema da heterogeneidade, quando se usa um ambiente com múltiplos provedores. Este problema deve ser resolvido tanto na seleção de recursos quando na sua alocação.

A implantação automatizada e autônoma de coreografias de serviços ainda é tratada de maneira muito incipiente. Geralmente, essa atividade é realizada manualmente, o que acarreta muitos problemas em casos que a pessoa responsável pela implantação mantém a documentação relacionada incompleta, adotando pressupostos não compartilhados por todo o time responsável por uma aplicação ou serviço [128]. Na revisão bibliográfica que realizamos (que será discutida no Capítulo 3) pudemos notar que na maioria das abordagens encontradas há uma preocupação apenas superficial com os recursos necessários para implantar os serviços que compõem as coreografias, sendo a satisfação de restrições não-funcionais tratada apenas na seleção destes serviços, assumindo que há recursos que ofereçam o nível de qualidade suposto.

Para implantar uma ou mais composições usando a abordagem proposta nesta tese não há uma diferenciação conceitual de composições que usam o modelo de coreografia ou composições que usam outro modelo. Isso porque, eventuais coordenadores em outros modelos também são implementados como serviços *Web*. Dessa forma, semelhante a [161], utilizamos os termos "coreografia" e "composição de serviços" indistintamente no restante do texto.

## 2.4 Restrições sobre serviços

Um produto de *software* é desenvolvido visando atender a um conjunto de funcionalidades e atributos gerais que dizem respeito a limitações ou à qualidade com que estas funcionalidades são realizadas. Estes requisitos são comumente chamados de requisitos não-funcionais (RNF) [228], atributos de qualidade [42, 148] ou objetivos [188]. Contudo, estes termos são geralmente empregados em atividades relacionadas ao desenvolvimento do produto de *software* ou quando sua utilização está limitada a um escopo pré-conhecido. Uma vez que em nossa abordagem propomos que serviços podem ser utilizados em cenários distintos, e dado que nos referimos somente a aspectos relacionados à execução dos serviços (que não necessariamente foram pensados durante seu desenvolvimento), preferimos adotar o termo ***"Restrição"***.

As restrições mais comuns são aquelas associadas à qualidade de serviço (QoS). Este termo se refere a quão bem um serviço ou produto é executado, levando em consideração critérios como desempenho, confiabilidade, segurança e usabilidade, entre outros. QoS pode englobar desde o *hardware*, passando pelos sistemas operacionais e protocolos de rede até chegar aos serviços da camada de aplicação [244].

Em nosso trabalho classificamos QoS em três categorias, que são ilustradas na Figura 2.8:

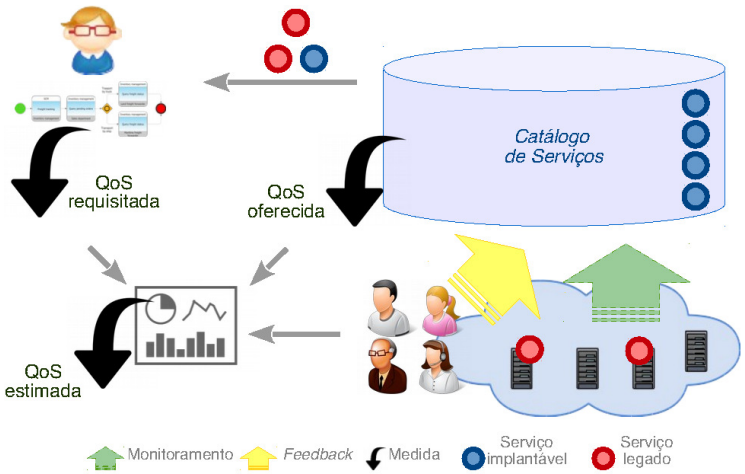



**Figura 2.8:** *Categorização de tipos de QoS e sua fonte de obtenção.*

- **QoS requisitada:** refere-se ao nível de QoS fornecido como restrição na descrição de um serviço ou coreografia. Refere-se ao que o usuário espera obter como qualidade ao solicitar a implantação de uma aplicação, sendo normalmente traduzida para um acordo de nível de serviço (*Service Level Agreement* – SLA) entre o usuário e o provedor de recursos.

- **QoS estimada:** é o nível de QoS esperado para um serviço que ainda não foi implantado. Esse valor pode ser obtido considerando execuções prévias do serviço ou usando alguma estratégia de predição [107, 223, 274]. Assumimos que medições dessa categoria podem ser usadas como parâmetro de decisão no provisionamento de recursos. Contudo, reconhecemos que, por se tratarem de estimativas, adaptações em tempo de execução podem ser necessárias caso a estimativa seja feita de forma imprecisa.

- **QoS oferecida:** é a QoS garantida para um determinado serviço através de um SLA. Os níveis de QoS estabelecidos em SLAs com provedores de nuvem se encaixam nessa categoria. Como exemplo, a Amazon oferece em seu serviço EC2 uma disponibilidade de 99,95% [10]. Este SLA, por sua vez, pode ser firmado como resultado da negociação prévia entre o provedor e o usuário, usando a QoS requisitada como parâmetro; ou ser uma oferta estabelecida pelo provedor, comum a todos os usuários. Usamos essa categoria principalmente para nos referir a serviços de terceiros, para os quais não temos controle. Nossa proposta considera um catálogo de serviços [168], que contém a descrição de um conjunto de serviços disponibilizados, com informações sobre o nível de QoS garantido pelo provedor de cada serviço. Valores de QoS oferecida também podem ser utilizados para estimar a QoS para cenários semelhantes ou cenários que dependem deste serviço.

Não há consenso sobre quais são os critérios de qualidade esperados para um serviço disponibilizado na *Web* [4]. A visão tradicional herdada da comunidade de redes coloca somente desempenho e disponibilidade no conjunto de propriedades de QoS, mas outras propriedades são relevantes também, como manutenibilidade, integridade, confiabilidade e segurança [172].

O oferecimento de QoS é um dos principais desafios em computação em nuvem, e este aspecto é ainda mais crítico ao implantar coreografias de serviços nesses ambientes, devido à existência de múltiplos serviços cooperando entre si e ao possível compartilhamento destes serviços entre as coreografias. Os mecanismos de elasticidade oferecidos por soluções de nuvem, usualmente, buscam controlar aspectos de qualidade de qualquer uma das partes que interagem entre si. Dessa forma, lingua-

gens de especificação de coreografias de serviços devem ser enriquecidas com notações que expressam as propriedades não-funcionais que o serviço coreografado deve apresentar [31]. Essas notações facilitam que os níveis de QoS, estabelecidos entre as partes envolvidas, sejam então formulados em termos de SLAs, mais técnicos e monitoráveis [37], uma vez que seguem uma sintaxe e semântica preestabelecidas.

Não somente restrições de QoS devem ser observadas na implantação e execução de serviços, mas também, aquelas relacionadas a outros aspectos, como, por exemplo, privacidade e segurança. Além disso, uma vez que composições de serviços podem ser formadas extrapolando fronteiras nacionais, deve-se considerar toda restrição relacionada à legislação e mecanismos regulatórios com respeito à privacidade, controle de acesso aos dados, segurança, *etc*.

Relatórios governamentais e pesquisas acadêmicas esboçam as complexidades da criação de serviços em nuvem relacionadas à conformidade com regulamentações e enfatizam a necessidade de soluções que permitam tal conformidade [142, 154, 209]. Como exemplo, as Diretrizes de Proteção a Dados da União Europeia [206] proíbem a transferência de dados pessoais para países não membros, a não ser que sejam garantidos os níveis adequados de proteção. No Brasil, medida semelhante foi adotada com o Marco Civil da *Internet* [53], que estabelece que em operações realizadas por meio da *Internet*, em território nacional, deverão ser obrigatoriamente respeitados a legislação brasileira e os direitos à privacidade, à proteção dos dados pessoais e ao sigilo das comunicações privadas e dos registros.

## 2.5 Exemplo de cenário

Em sistemas orientados a serviços, os serviços são entidades de *software* autônomas que, interagindo com outros serviços em um estilo par-a-par, podem tomar decisões proativamente e engajarem em tarefas objetivando suas próprias necessidades e colaboração global [24]. Para ilustrar a complexidade na implantação e manutenção de um sistema com esta característica, apresentamos um exemplo do cenário trabalhado na tese. Este exemplo constitui uma aplicação de gerenciamento de cadeia de suprimentos.

O gerenciamento da cadeia de suprimentos é a integração dos principais processos de negócio através de fornecedores que proveem produtos e informações que agregam valor aos clientes [156]. À medida que as empresas fortalecem seus relacionamentos e colaboram em um nível intra e inter-organizacional, a própria cadeia ganha mais ligações e, portanto, aumentam os esforços de gestão e coordenação. Como resultado, o foco do gerenciamento da cadeia de suprimentos é cada vez mais transferido

da eficiência da produção para abordagens voltadas aos clientes e direcionadas a sincronização de parcerias [261].

A tecnologia da informação transformou a maneira como as empresas implementam essas abordagens, resultando em diferencial competitivo. Ao adotar soluções orientadas a serviços, através de coreografias implantadas em ambientes de nuvem, as cadeias de suprimentos podem, com mais rapidez e eficiência, colher os benefícios a custos reduzidos.

Uma implantação eficiente de coreografias de serviços nesse cenário é determinada pelo desempenho de toda a cadeia de suprimentos, e não somente de suas entidades individuais. Com isso em mente, a implantação deve levar em conta diferentes restrições impostas em cada serviço. Essa pluralidade de restrições acontece porque cada serviço pode ser direcionado a clientes com necessidades heterogêneas e complexas (motivadas por idade, condição social, *etc.*) e possuir carga sujeita a padrão sazonal e outros padrões de variação, como aumento da demanda devido a promoções (ex.“*Black Friday*”). Além disso, a implantação dos serviços deve enfrentar restrições derivadas de processos internos e externos.

Para simplificar, em nosso exemplo, analisamos apenas dois processos: o cumprimento de pedidos e o rastreamento de frete, descritos na Figura 2.9 usando a linguagem BPMN. O primeiro processo (descrito na Figura 2.9(a)) objetiva o cumprimento da ordem de compra após a seleção do produto. Ele inclui o planejamento de frete e gerenciamento de pagamentos, e é implementado usando uma composição de seis serviços. Para essa coreografia, assumimos que a opção de pagamento é selecionada pelo cliente antes de iniciar o processo (ao submeter o pedido). O segundo processo é um processo interno (sem interação com clientes). Ele inclui as tarefas realizadas para obter um relatório de estado do frete contratado para produtos vendidos e produtos do inventário da organização.

Conforme ilustrado na Figura 2.9(b), assumimos que a empresa usa dois modos de transporte (por terra e por mar) e esse relatório é gerado usando uma composição de quatro serviços. A Tabela 2.1 apresenta a descrição dos serviços para ambas as coreografias.

| **Serviço** | **Descrição** |
|---|---|
| Gerenciamento de inventário | Controla a encomenda e armazenamento de produtos para venda |
| Departamento de vendas | Responsável por tarefas relacionadas a produtos destinados à venda |
| Frete terrestre | Interage com uma empresa que envia bens do fabricante ou produtor até o ponto final de distribuição por terra |
| Frete marítimo | Interage com uma empresa que envia bens do fabricante ou produtor até o ponto final de distribuição por mar |
| Escalonador de frete | Organiza remessas de produtos vendidos usando uma ou mais empresas de frete |
| CRM | Gerencia relacionamento com o cliente (*Customer Relationship Management*) |
| Sistema bancário | Interage com parceiros bancários |
| Sistema de pagamento | Processa pagamentos feitos com cartão de crédito |

**Tabela 2.1:** *Exemplo do cenário: descrição dos serviços.*

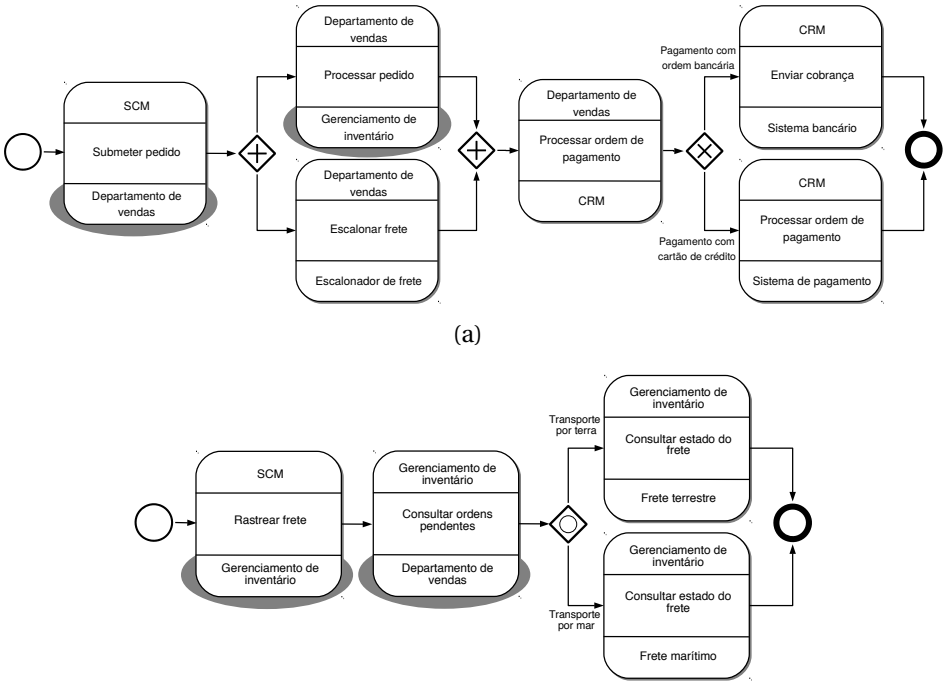



**Figura 2.9:** *Diagrama BPMN das coreografias apresentadas no exemplo do cenário: (a) Coreografia 01: Cumprimento de pedido. (b) Coreografia 02: Rastreamento de frete.*

A implantação dessas coreografias deve lidar com propriedades não-funcionais requeridas em virtude de restrições inerentes do cenário considerado. Assumimos em nosso exemplo as restrições não-funcionais listadas na Tabela 2.2, que relaciona também cada serviço às restrições impostas sobre ele.

Considere que a empresa deseja implantar essas coreografias usando serviços preexistentes. Com isso, o primeiro desafio que surge é a seleção de serviços. Há papéis compartilhados entre as coreografias apresentadas, como gerenciamento de inventário e da venda de produtos (serviços sombreados na Figura 2.9). É desejável que haja compartilhamento de serviços para evitar custos adicionais, como licenças de *software* duplicadas e mais despesas na alocação de recursos, o que motiva o uso de compartilhamento de serviços.

O segundo e principal desafio está relacionado ao gerenciamento de recursos, principalmente às atividades de estimativa, seleção e alocação de recursos. Algumas das dificuldades são:

1. a estimativa e a seleção de recursos devem ser realizadas de forma que todas as restrições não-funcionais sejam satisfeitas. Conforme resumido na Tabela 2.1, além de restrições comuns às duas coreografias, existem restrições específicas para cada uma delas e o compartilhamento de serviços causa interferência entre elas;

| Serviço | Restrições específicas a determinados serviços | Restrições comuns a todos os serviços |
|---|---|---|
| Gerenciamento de inventário | • Na coreografia 01: Objetivando as recomendações sobre experiência de usuário [191], o tempo máximo necessário para atender a cada requisição deve ser 1 (um) segundo.<br>• Na coreografia 02: A emissão de relatórios sobre frete deve manter uma vazão de 10 (dez) requisições por segundo.<br>• Não pode estar indisponível mais do que 12 (doze) horas por ano. | • A implantação de serviços deve ser realizada usando recursos com menor custo financeiro (desde que as demais restrições sejam satisfeitas).<br>• A seleção de recursos deve ser realizada de forma a evitar provedores de nuvem com obrigação de cobrança mínima, visto que a utilização de recursos é desconhecida.<br>• A seleção de recursos deve privilegiar provedores de nuvem com melhor reputação. |
| Departamento de vendas | • Na coreografia 01: O tempo máximo necessário para atender a cada requisição deve ser 1 (um) segundo.<br>• Na coreografia 02: A emissão de relatórios sobre frete deve manter uma vazão de 10 (dez) requisições por segundo. | |
| Frete terrestre | • A emissão de relatórios sobre frete deve manter uma vazão de 10 (dez) requisições por segundo. | |
| Frete marítimo | • A emissão de relatórios sobre frete deve manter uma vazão de 10 (dez) requisições por segundo. | |
| Escalonador de frete | • O tempo máximo necessário para atender a cada requisição (na coreografia 01) deve ser 1 (um) segundo. | |
| CRM | • O tempo máximo necessário para atender a cada requisição (na coreografia 01) deve ser 1 (um) segundo.<br>• Dados sobre clientes brasileiros devem ser armazenados no Brasil. | |
| Sistema bancário | • O tempo máximo necessário para atender a cada requisição (na coreografia 01) deve ser 1 (um) segundo. | |
| Sistema de pagamento | • O tempo máximo necessário para atender a cada requisição (na coreografia 01) deve ser 1 (um) segundo.<br>• Deve ser mantido em uma nuvem privada uma vez que gerencia uma base de dados sobre cartões de crédito. | |

**Tabela 2.2:** *Exemplo do cenário: descrição das restrições não-funcionais.*

2. serviços compartilhados podem desempenhar diferentes papéis em cada coreografia, como o serviço para gerenciamento de inventário que é usado para processar pedido e para rastrear frete. Além disso, há degradação da QoS nesses serviços causada pela concorrência proveniente da agregação da carga. Devido a essa concorrência, é preciso selecionar recursos com maior capacidade para que seja possível manter a QoS dentro dos padrões esperados para o serviço;
3. as contribuições nas restrições fim-a-fim devem ser distribuídas de forma eficiente entre os serviços a que elas dizem respeito. Deve haver um balanceamento da QoS esperada para cada serviço para que, assim, a agregação dos valores de QoS de todos os serviços que compõem a coreografia fique dentro do limite requisitado. Esse balanceamento deve ser realizado de maneira global considerando todas as coreografias em que um determinado serviço é usado;
4. dada a capacidade de recurso estimada para garantir que todas as restrições de QoS sobre o serviço a ser implantado sejam satisfeitas, é necessário decidir onde alocar essa capacidade. Esta tarefa pode iniciar com uma tentativa de alocação usando recursos disponíveis na própria organização. No entanto, quando isso não é possível é necessário fazer o mapeamento para recursos em nuvens públicas. A alocação de recursos também está sujeita às restrições que não têm relação direta com a capacidade, como por exemplo, localização do recurso.

As três primeiras dificuldades relacionadas ao gerenciamento de recursos são

problemas bem conhecidos. Para reforçar a motivação do trabalho, apresentamos uma análise mais detalhada sobre o quarto desafio. A seleção de recursos pode ser realizada usando um ambiente multi-nuvem para evitar *lock-in* de fornecedor, otimizar o desempenho e os custos e facilitar a satisfação das restrições não-funcionais [212]. No entanto, esta não é uma tarefa simples, devido ao grande número de opções de tipos de VM e a interseção entre eles. Atualmente existem mais de 120 provedores de nuvem na categoria IaaS, que oferecem recursos em uma ampla quantidade de tipos [117]. Como exemplo, comparamos os tipos de VM oferecidos pelo provedores líderes de mercado de acordo com o relatório sobre IaaS feito em 2015 pela Gartner [163] – Amazon *Web* Services (AWS) [11], Microsoft Azure [182], Google Compute Engine (GCE) [111] e Rackspace [220].

Como ilustrado na Figura 2.10, há 827 tipos de VM[2] que devem ser considerados ao selecionar onde cada serviço coreografado será implantado, o que dependerá das restrições não-funcionais associadas. Essa seleção não pode ser realizada apenas com base na capacidade de *hardware*, uma vez que tipos com exatamente a mesma capacidade possuem atributos variados (como custo e localização, por exemplo), o que pode interferir na seleção se houver alguma restrição associada a esses atributos. Em virtude disso, selecionar o recurso para cada serviço é um problema de otimização complexo, que se torna ainda mais difícil ao considerarmos diferentes padrões de interação entre os serviços e seu provável compartilhamento em múltiplas coreografias.

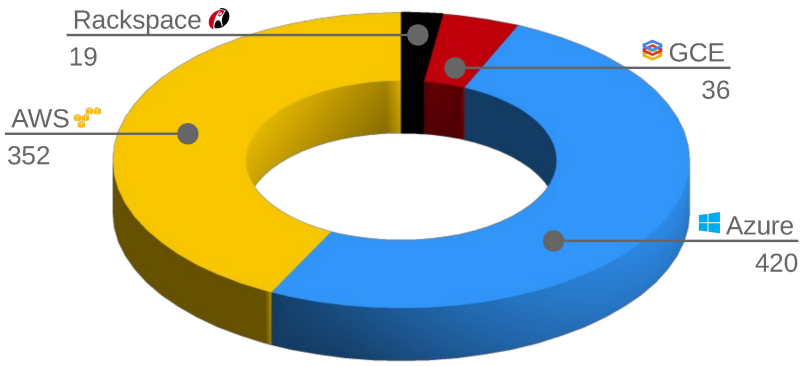



**Figura 2.10:** *Número de tipos de VM disponíveis nos principais provedores de nuvem.*

A seleção de recursos pode ainda explorar o compartilhamento dos próprios recursos, procurando implantar um conjunto de serviços em um único recurso. Isso ocorre quando o recurso selecionado para um certo serviço (ou conjunto de serviços) possui capacidade maior que a demanda desse serviço, permitindo que o recurso possa ser compartilhado. Essa estratégia é ainda mais necessária nos casos onde há

[2]Essa análise foi realizada considerando os tipos de VM disponibilizados pelos provedores considerados em Julho/2016.

grande demanda de comunicação entre os serviços, uma vez que a implantação conjunta de múltiplos serviços em um único recurso elimina a sobrecarga de transmissão de dados entre eles.

## 2.6 Considerações Finais

Neste capítulo, foram discutidos, de maneira introdutória, alguns conceitos que serão utilizados no decorrer da tese: computação em nuvem, coreografia de serviços e restrições sobre serviços. Foram apresentados tópicos relacionados ao espaço do problema, assim como aqueles relacionados à solução proposta.

Computação em nuvem representa um avanço na TI e vem sendo empregada em cenários cada vez mais complexos, nos quais a entrega de soluções que satisfaçam os requisitos não-funcionais das aplicações, representa um desafio. Para lidar com essa complexidade, diversas técnicas estão sendo propostas. Nesse sentido, propomos uma abordagem para o provisionamento eficiente de recursos para coreografias de serviços. Como será detalhado a partir do Capítulo 4, a solução se beneficia da variabilidade de tipos de recursos providos em ambientes de nuvem, assim como da elasticidade em seu uso.

Coreografia de serviços é uma maneira promissora de composição de serviços, possibilitando a interação de múltiplos serviços que cooperam entre si [211]. Para evidenciar a existência dos principais desafios existentes na implantação de aplicações desenvolvidas nesse modelo apresentamos um exemplo do cenário que será usado para ilustrar o problema no decorrer da tese. Argumentamos que a satisfação das restrições impostas sobre os serviços deve ser pensada desde as primeiras atividades de criação de uma coreografia, ou seja, no momento em que é feita sua especificação. Apesar disso, as soluções para modelagem e implantação de coreografias ainda são bastante incipientes para lidar com a dinamicidade inerente dos cenários atuais. No próximo capítulo discutiremos algumas dessas abordagens, destacando suas vantagens e desvantagens.

CAPÍTULO 3

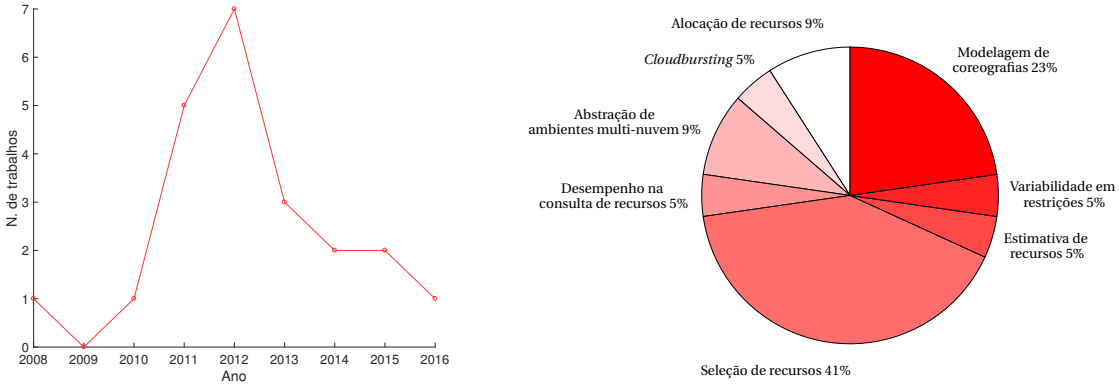
# Trabalhos relacionados

Este capítulo discute o estado da arte na modelagem, seleção e alocação de recursos em nuvem. Buscamos analisar como essas atividades são realizadas para serviços em geral e buscamos evidenciar as especificidades de composições de serviços descritas em coreografias. A análise das abordagens discutidas foi guiada pelas seguintes perguntas:

- As soluções existentes para modelagem de coreografias de serviços são suficientes para permitir a representação de restrições não-funcionais que devem satisfeitas na implantação dos serviços? Caso contrário, que características devem ser incorporadas nessas soluções para permitir a especificação destas restrições?
- Como o trabalho analisado escolhe a capacidade de recursos para implantar os serviços? Quais os critérios utilizados para selecionar o recurso mais apropriado? Qual o impacto dessa decisão, se aplicada à implantação de coreografias?
- Como as restrições não-funcionais são consideradas na implantação do serviço ou coreografia?
- As soluções propostas nos trabalhos analisados são independentes de tipos de aplicação/serviço e de provedor de nuvem? Como elas permitem abstrair a seleção e mapeamento de recursos em nuvens híbridas ou qual seria o efeito colateral de aplicá-las em tais cenários?
- É possível utilizar essas soluções para implantar de maneira eficiente um conjunto de coreografias de serviços, de forma que as interferências nessas coregrafias causadas pelo compartilhamento de serviços sejam absorvidas ou quais os efeitos colaterais de utilizá-lo para tal implantação?

A revisão bibliográfica foi realizada como primeira etapa da pesquisa, sendo reforçada no decorrer do desenvolvimento do trabalho. Foram realizadas buscas nas bases científicas IEEE Xplore [131], ACM Digital Library [2], SciVerse Scopus [84], SpringerLink [247], CiteSeerX [255] e Web of Science [224]. Os resultados obtidos foram complementados com buscas usando o Google Scholar [113]. Durante o desenvolvimento da pesquisa, ao todo foram pré-selecionados 423 trabalhos, de acordo com o título, o

resumo, a introdução e a conclusão. Após uma análise mais criteriosa, esses trabalhos foram filtrados de forma a obter as propostas mais relevantes para a tese. Selecionamos 22 trabalhos relacionados, cuja distribuição de acordo com o ano da publicação mais relevante e principal tema abordado é apresentada na Figura 3.1. Para facilitar a análise e isolar o escopo, os critérios estabelecidos como importantes no problema considerado foram divididos em três grupos – modelagem de coreografias, gerenciamento de recursos em nuvem e implantação de composições de serviços –, que serão discutidos nas próximas seções. Para cada grupo, apresentamos um comparativo desses trabalhos usando os critérios estabelecidos.




(a) (b)

**Figura 3.1:** *Distribuição dos trabalhos relacionados por (a) ano da principal publicação e (b) por tema principal.*

## 3.1 Modelagem de coreografias de serviços

No capítulo anterior foram discutidas as principais categorias de restrição não-funcional, relacionadas à implantação e encenação de coreografias de serviços. Além de diferentes propriedades de QoS, as restrições não-funcionais envolvem características do ambiente de implantação dos serviços, que podem ser diretamente relacionadas ao recurso ou ao provedor de nuvem que o disponibiliza. Também foram apresentadas as principais linguagens existentes para modelagem de coreografia de serviços. Embora as notações propostas sejam suficientes para representar as interações entre os serviços e expressar a lógica do processo de negócio executado, elas não permitem especificar todas as informações necessárias para realizar a implantação destes serviços, visando propriedades não-funcionais. Mais precisamente, as linguagens existentes não permitem a representação das diferentes categorias de restrição de maneira não-ambígua e processável. Para contornar este problema, há algumas propostas que visam estender notações preexistentes, enquanto outras visam estabelecer novas linguagens. Dentre as classificadas como extensões, a maioria se baseia em

BPMN [198], uma vez que esta linguagem é o padrão *de facto* para a representação de coreografias de serviços.

A linguagem *Quality for BPMN* (Q4BPMN) [30] se baseia na anotação de diagramas expressos em BPMN com requisitos de qualidade que os serviços coreografados devem satisfazer. Essas propriedades não-funcionais podem estar relacionadas com uma tarefa, um participante ou a coreografia como um todo. Q4BPMN é uma implementação do *Property Meta-Model* (PMM) [78], um metamodelo para expressar propriedades de qualidade, métricas e eventos observáveis de um sistema. Em [31], os autores propuseram o uso deste modelo BPMN anotado em conjunto com técnicas de *Model Driven Engineering* (MDE) [239] e a ferramenta KlaperSuite [63] para permitir a análise automática de propriedades de qualidade sobre a coreografia. Uma das desvantagens de usar essa linguagem é que as propriedades de qualidade ficam restritas às propriedades inerentes do metamodelo PMM. Apesar dele ser completo o suficiente para expressar os tipos mais comuns de restrições de qualidade, não é possível incluir na modelagem de uma coreografia restrições não consideradas nesse metamodelo. Outra desvantagem é que essa linguagem não permite modelar restrições que são diretamente relacionadas a atributos do recurso (como por exemplo, sua localização), sendo ela limitada a aspectos de qualidade de serviço. Além disso, essa linguagem não permite a modelagem ou a inspeção de múltiplos processos de negócio de maneira conjunta, tornando difícil o desenvolvimento de uma abordagem para a implantação de múltiplas coreografias de maneira coordenada.

*Performability-enabled BPMN* (PyBPMN) [41] é uma extensão do metamodelo de BPMN para a especificação de propriedades de desempenho e confiabilidade usando o perfil UML *Modeling and Analysis of Real-Time Embedded systems* (MARTE) [197] e suas extensões propostas em [35, 36]. De forma semelhante que Q4BPMN, técnicas de MDE associadas à especificação do metamodelo permitem que os modelos desenvolvidos sejam transformados em modelos de análise de desempenho e confiabilidade. Uma limitação desta proposta é que o mecanismo de anotação proposto para especificar os atributos de qualidade é complexo, dificultando o uso por usuários que não estão familiarizados com este padrão. Além disso, ele também só engloba restrições relacionadas ao desempenho e à confiabilidade do serviço em um único processo de negócio. Com isso, essa linguagem apresenta as mesmas limitações apontadas para a Q4BPMN com relação à expressividade de restrições e tratamento conjunto de múltiplas composições, dificultando seu uso no escopo tratado para o problema.

Pavlovski e Zou [208] propuseram o uso de duas construções apresentadas em [283] como uma extensão de BPMN para modelar requisitos não-funcionais sobre o processo de negócio. A primeira construção, denominada **condição operacional**

(*operating condition*) denota uma restrição sobre o processo de negócio, enquanto a segunda, **caso de controle** (*control case*) define critérios de controle para mitigar riscos associados com uma condição de operação. Os autores alegam que a representação dessas características na fase inicial do processo de desenvolvimento de *software* permite determinar de maneira detalhada grande parte dos requisitos não-funcionais. Além de atributos de qualidade, a proposta também permite modelar restrições de regulamentação. As condições operacionais propostas pelo modelo podem ser usadas para decompor as restrições, mas a especificação de ambas construções é feita de maneira informal. Isso torna difícil garantir a implementação automatizada de análises realizadas usando essa linguagem, especialmente considerando de forma conjunta múltiplas composições, visto que a linguagem não oferece nenhuma construção com esse intuito.

BPMN4TOSCA [153] é uma extensão de BPMN usando *OASIS Topology and Orchestration Specification for Cloud Applications* (TOSCA) [194], um padrão para representar, de maneira portátil, topologias de aplicativos em nuvem e seu gerenciamento, que inclui provisionamento de recursos e implantação da aplicação. O modelo gerado usando essa linguagem não é uma descrição de um processo de negócio mas sim um **plano de gerenciamento** (termo utilizado no trabalho), que descreve, por exemplo, quais componentes de *software* devem ser instalados e os *scripts* que orquestram essa instalação. Dessa forma, diferente das propostas anteriores que tem como escopo as etapas de análise e projeto da aplicação, BPMN4TOSCA objetiva as atividades relacionadas à sua implantação. Essa abordagem pode ser empregada quando o usuário é responsável direto pela alocação de recursos. Contudo, nosso objetivo é abstrair completamente as tarefas relacionadas a esta atividade, não justificando assim sua adoção. Outro problema em considerar seu uso é que planos de gerenciamento são tipicamente altamente acoplados a uma única aplicação, sendo difícil reusá-los e gerenciá-los [46].

A linguagem *Extended Process Pattern Specification Language* (EPPSL) [149] tem como objetivo a modelagem de restrições de qualidade usando uma extensão de Diagramas de Atividade da UML 2.0. O modelo gerado indica quais são as restrições de qualidade e em quais atividades elas devem ser aplicadas. De acordo com os autores, a linguagem tem maior capacidade de expressividade por permitir a modelagem de restrições temporais não-determinísticas usando *Computation Tree Logic* (CTL) [85], como, por exemplo, restrições que futuramente não se aplicarão a todos os fluxos de controle em um processo de negócios. Essa capacidade é relevante principalmente para instrumentar mecanismos de adaptação dos recursos face a mudanças em tempo de execução. Contudo, o principal objetivo do trabalho é permitir a verificação do modelo gerado, sendo não recomendado o uso dessa solução visando apenas a criação

de modelos devido à sua alta complexidade.

### 3.1.1 Comparativo

Nesta seção realizamos um comparativo dos trabalhos apresentados, considerando características que acreditamos ser relevantes em uma linguagem de modelagem de coreografias para que ela possa guiar a implantação dos serviços de maneira eficiente e de acordo com as restrições não-funcionais. A Tabela 3.1 divide essas caraterísticas em duas categorias: *escopo*, indica os aspectos que são relevantes em uma linguagem de modelagem no contexto estabelecido; e *propriedades que auxiliam na implantação da aplicação*, indica as características da linguagem que podem facilitar as atividades relacionadas à implantação dos serviços. Apesar de alguns atributos não fazerem parte do domínio de todas as linguagens, podemos notar que nenhum trabalho contempla de maneira completa todos os aspectos considerados.

| | Q4BPMN [30, 78] | PyBPMN [41] | Pavlovski e Zou [208, 283] | BPMN4TOSCA [153] | EPPSL [149] |
|---|---|---|---|---|---|
| **Escopo** | | | | | |
| Modelagem do processo de negócio | ✓ | ✓ | ✓ | ☑ | ✓ |
| Modelagem de restrições sobre determinada tarefa (serviço) | ✓ | ✓ | ✓ | ✓ | ✓ |
| Modelagem de restrições fim-a-fim | ✓ | ☑ | ☑ | | |
| Representação de múltiplas categorias de restrições | | | ✓ | | ✓ |
| Representação de múltiplas propriedades de QoS | ✓ | ☑ | ✓ | ■ | ■ |
| **Propriedades que auxiliam na implantação da aplicação** | | | | | |
| Linguagem definida de maneira formal | ✓ | ✓ | | ✓ | ✓ |
| Representação explícita das dependências entre as entidades | ☑ | ☑ | ☑ | ■ | |
| Representação conjunta de múltiplos processos de negócio | | | | ✓ | |
| Representação de componentes associados à avaliação das restrições | | | | | |

***Legenda:*** ✓ Contemplado, ☑ Contemplado com limitações, ■ Fora do domínio do trabalho.

**Tabela 3.1:** *Comparativo das principais propostas para modelagem de coreografias de serviços.*

Entre as principais deficiências encontradas no escopo das propostas existentes, está a impossibilidade de modelar de maneira não ambígua restrições fim-a-fim, sobre o processo de negócio. Em alguns casos, como em PyBPMN, essa funcionalidade é oferecida de maneira indireta através da agregação de restrições a tarefas individuais, sendo necessário a quebra do valor-alvo da restrição entre as tarefas ou subprocessos que fazem parte desta restrição. Outro problema é que os trabalhos são voltados principalmente para restrições de QoS, geralmente permitindo a modelagem de um subconjunto predeterminado de propriedades desse tipo, sendo outras restrições (como

aquelas relacionadas à regulamentação) desconsideradas. Dessa forma, o uso dessas propostas não permite cobrir todo o possível conjunto de restrições que identificamos.

Com relação ao suporte para realizar a implantação, uma das limitações visualizadas é que as dependências entre os serviços não são facilmente identificadas em alguns casos. Como exemplo, nas linguagens baseadas em BPMN para identificar com quais outros participantes um certo membro interage é preciso percorrer tarefa a tarefa, o que pode tornar a avaliação das restrições ineficiente e dificultar as análises que se baseiam em uma visão global do processo. Outra informação que não é facilmente obtida é a possibilidade de compartilhamento de participantes, uma vez que modelos são exclusivos para cada processo.

Por fim, nenhuma das linguagens consideradas permite especificar componentes associados à avaliação das restrições, como por exemplo, indicar a implementação de uma funcionalidade que pode ser utilizada para estimar as propriedades relacionadas à verificação de uma determinada restrição. Apesar de ser natural assumir que a especificação dessas informações está fora do escopo da linguagem de modelagem, seu fornecimento pode ser útil nas decisões de implantação. Isso se dá porque nos casos em que essas informações não são especificadas, deve-se assumir que os valores das propriedades são previamente conhecidos ou que um mesmo componente é sempre usado para cada propriedade. Contudo, o uso de componentes únicos preestabelecidos pode ser problemático. Como exemplo, ao especificar uma restrição relacionada à latência para processar uma determinada requisição, não é possível garantir para todos os casos quais variáveis estão envolvidas nessa estimativa e nem qual a técnica mais apropriada para obtê-la uma vez que algumas técnicas podem não oferecer a precisão necessária em alguns casos, ao passo que são suficientes em outros.

## 3.2 Gerenciamento de recursos

Conforme discutido no Capítulo 2, o principal problema das ferramentas atualmente disponíveis para gerenciamento de recursos em nuvem é que elas não possibilitam, de maneira eficaz, que a estimativa e o provisionamento de recursos seja realizado considerando diferentes provedores. Em contrapartida, a melhor estratégia para favorecer a satisfação de todas as restrições requisitadas e obter índices melhores de desempenho e menor custo é utilizar múltiplos provedores de nuvem. Diante disso, alguns trabalhos visam oferecer soluções neste cenário, contornando uma ou mais limitações das ferramentas comerciais. A seguir apresentamos alguns desses trabalhos, para os quais as aplicações constituem componentes de *software* ou serviços isolados, que são independentes uns dos outros. As especificidades do gerenciamento de recursos para composições de serviços serão discutidas na próxima seção.

Um dos grandes desafios no provisionamento automatizado de recursos é abstrair detalhes da interação com o provedor de recursos, o que é ainda mais complexo ao se considerar o uso de múltiplos provedores pois, neste caso, é preciso lidar com o dinamismo típico de um ambiente multi-nuvem. Uma técnica comumente adotada para prover essa abstração e facilitar as atividades de gerenciamento de recursos é utilizar modelos em tempo de execução (*model@runtime*, ou simplesmente m@rt) que consiste em uma abstração do sistema que é manipulada em tempo de execução para um propósito específico [33]. Um m@rt pode ser definido como uma auto representação causalmente conectada do sistema que representa sua estrutura e comportamento ou objetivos do ponto de vista do espaço do problema. Isto significa que o sistema "conhece" a si mesmo e que mudanças nos modelos são propagadas para o sistema em execução e vice-versa. Com isso, adaptações em tempo de execução podem ser realizadas nos modelos sem a necessidade de se conhecer a implementação do sistema em si.

Nas abordagens baseadas em m@rt, os modelos passam a fazer parte significativa do sistema em questão. Com isso, é possível o projeto, evolução e verificação de sistemas de *software* de maneira contínua [39]. Além da vantagem de conferir uma maior adaptabilidade ao sistema, o uso dessa abordagem facilita tomadas de decisões, permite tarefas mais precisas de validação e monitoramento, além de melhorar a sincronização entre os artefatos de projeto e a implementação do *software* [105]. Um exemplo de abordagem neste sentido é *Communication Virtual Machine* (CVM) [76], uma máquina de execução de modelos usada para apoiar a coordenação automática de serviços de comunicação centrados no usuário, tendo como base o uso de modelos em tempo de execução.

Dentre os trabalhos relacionados que se baseiam no uso de m@rt, *Cloud Modelling Framework* (CloudMF) [89, 56, 90] provê uma *Domain-Specific Modeling Language* (DSML) para implantação de aplicações em sistemas multi-nuvem, que facilita a implantação e adaptação de aplicações em tempo de execução. Contudo, no CloudMF não há preocupação em satisfazer de forma ampla restrições impostas sobre a aplicação, uma vez que a principal motivação em se usar um ambiente multi-nuvem neste caso é dispor de mais opções de recurso para atender a uma capacidade de *hardware* preestabelecida pelo usuário. Dessa forma, é o usuário que deve especificar a capacidade dos recursos necessários para a aplicação, além de regras para seu provisionamento. O problema em delegar ao usuário essa escolha é que os recursos podem ser super- ou subdimensionados, além de dificultar a implantação da aplicação ao incluir uma tarefa adicional. Outra limitação no CloudMF é o fato de usarem apenas nuvens públicas, fazendo com que a solução não se beneficie de recursos já existentes na organização. O CloudMF é um subprojeto do projeto *MOdel-Driven Approach for design*

*and execution of applications on multiple Clouds* (MODAClouds) [185, 19], que objetiva prover um sistema de suporte a decisões, uma IDE *open source* e um ambiente de tempo de execução para o projeto, a prototipação, a geração de código e a implantação automática de aplicações com garantias de QoS. Contudo, mesmo com as extensões consideradas no projeto ainda há limitações, como delegar ao usuário a responsabilidade por definir toda a infraestrutura a ser utilizada.

*Open source API and Platform for multiple Clouds* (mOSAIC) [87, 235, 212] é um projeto no qual é proposta uma API *open source* e uma plataforma para desenvolvimento de aplicações multi-nuvem. Seu objetivo é prover abstração para que serviços em nuvem possam ser facilmente desenvolvidos e implantados, utilizando um sistema multi agente [236] e computação autonômica. Neste projeto também é proposta uma ontologia [186] usada na anotação de recursos com propriedades funcionais e não-funcionais. Para auxiliar na estimativa da infraestrutura, mOSAIC oferece também um arcabouço para o desenvolvimento de *benchmarks* customizados para medir o desempenho da aplicação sobre cargas preestabelecidas. Contudo, esses *benchmarks* devem ser criados de maneira *ad-hoc* para cada aplicação.

A abstração e dinamicidade obtida com o uso de modelos em tempo de execução não é suficiente para resolver também o problema de seleção de recursos. Esta atividade é tratada de maneira isolada em alguns trabalhos usando processo analítico hierárquico (*Analytic Hierarchy Process* – AHP) [232]. AHP é um método amplamente usado para resolver problemas relacionados à tomada de decisão usando múltiplos critérios, o que é definido como *Multiple Criteria Decision Making* (MCDM) [281]. Este método consiste em decompor problemas complexos e mal estruturados, organizando os fatores que influenciam na decisão a ser tomada em uma estrutura hierárquica formada por subproblemas que podem ser analisados independentemente. Uma vez que esta estrutura é construída, o resultado é obtido através da avaliação sistemática de seus elementos, comparando-os uns aos outros par a par, com relação ao seu impacto sobre um elemento acima deles na hierarquia. Um peso ou prioridade é derivado para cada elemento da hierarquia, permitindo que elementos diversos e muitas vezes incomensuráveis sejam comparados entre si de uma maneira racional e consistente [233]. O método pode ser resumido como:

1. Inicialmente é feita a modelagem do problema como uma hierarquia contendo o objetivo da decisão, as alternativas para alcançá-lo e os critérios para avaliar as alternativas.
2. O responsável pela decisão indica a significância relativa entre os atributos. Por exemplo, em nosso cenário, o responsável pela decisão pode preferir o custo de utilização do recurso sobre a localização e a localização sobre a reputação do provedor.

3. Similarmente, para cada atributo, e para cada par de alternativas o responsável pela decisão especifica suas preferências (por exemplo, se a localização da alternativa *A* é melhor do que a da *B*).
4. As preferências são então sintetizadas para produzir um conjunto de prioridades gerais para a hierarquia. Nessa etapa também é verificada a consistência das preferências.
5. A decisão final é obtida com base nos resultados deste processo.

SMICloud [100, 101] usa AHP para comparar parâmetros de diferentes provedores de nuvem através de um método de classificação que reduz o custo na utilização de VMs. O trabalho propõe um conjunto de métricas para sistematicamente medir todos os atributos de QoS propostos pelo *Cloud Service Measurement Index Consortium* (CSMIC) [69], um consórcio lançado pela *Carnegie Mellon University* com o intuito de desenvolver o *Service Measurement Index* (SMI). O SMI é um arcabouço que padroniza um conjunto de características, atributos e medidas que tomadores de decisão podem aplicar para permitir a comparação de múltiplos provedores de nuvem. Usando esse arcabouço é feito o ranqueamento de recursos em nuvem com base nos atributos estabelecidos.

Outra proposta baseada em AHP é CloudGenius [180], construído sobre o arcabouço $(MC^2)^2$ [181]. Este sistema automatiza o processo de tomada de decisão baseando-se em um modelo para migração de servidores *Web* para a nuvem, que estabelece o peso para cada parâmetro da decisão. Para a seleção e combinação de soluções, CloudGenius constrói um modelo formal que descreve requisitos, além de atributos numéricos e não numéricos.

O uso de AHP em ambas abordagens requer uma quantidade significativa de dados que devem ser fornecidos pelo usuário para priorizar os diferentes requisitos. Outra limitação em usar essas propostas é o fato delas não constituírem estratégias de emparelhamento que leva em consideração a relação de um serviço (ou uma aplicação) com outros, ignorando a influência que pode haver entre elas. Essas abordagens apenas constituem uma forma de filtragem de recursos usando critérios que devem ser estabelecidos pelo usuário. Em outras palavras, elas apenas auxiliam na seleção de recursos, o que é realizado de maneira restrita pois não se considera o uso de nuvem privada e a seleção ocorre de forma isolada para cada serviço. As demais atividades de gerenciamento de recursos está fora do escopo dessas abordagens. Essa abordagem também não considera o uso de nuvem privada e não oferece soluções para as demais atividades de gerenciamento de recursos.

Dentre os trabalhos que usam técnicas diferentes de AHP para selecionar recursos, Sundareswaran *et al.* [252] usam um modelo de *broker* de nuvem que extrai informações de provedores e as indexam em uma estrutura chamada de Índice *Cloud*

*Service Provider* (CSP). Essa estrutura é então usada para realizar busca eficiente a consultas de usuários, considerando um número grande de provedores. Dada uma certa consulta, um número predeterminado de recursos candidatos é selecionado. Esse resultado é então refinado e reduzido usando um algoritmo que os autores demonstraram ser até 100 vezes mais eficiente que uma estratégia de busca exaustiva. Apesar disso, o foco do trabalho é o desempenho na obtenção dos resultados, não considerando outros critérios como o balanceamento ou interferência de resultados sobre múltiplas consultas.

Sun *et al.* [267] propuseram um modelo baseado em restrições para especificação e seleção de recursos em ambientes com múltiplas nuvens públicas. Da mesma maneira que em nossa proposta, o modelo é focado na aplicação, de forma que os recursos são especificados usando uma linguagem independente de provedor. Uma camada de abstração é usada para descobrir os recursos de infraestrutura mais adequados para uma aplicação [251], com base em uma especificação pré-elaborada.

O projeto InterCloud [49] é uma das primeiras iniciativas a explorar QoS em um ambiente multi-nuvem. Ele estende esforços anteriores do projeto InterGrid [77, 155] para permitir compartilhamento de recursos entre provedores de nuvem. A arquitetura é centralizada em uma entidade chamada *Cloud Exchange* (CEx), que age como um *marketplace*, em que provedores podem vender recursos. Os compradores podem ser outros provedores ou usuários desejando implantar aplicações. VMs podem ser implantadas em diferentes tipos de recurso, como infraestruturas privadas ou nuvens públicas.

Como pode ser notado, estes trabalhos objetivam apenas a seleção de recursos, assumindo que a estimativa da infraestrutura que será utilizada foi previamente realizada. Outra suposição assumida é que a QoS obtida ao selecionar um determinado recurso também é conhecida, não sendo possível ao usuário especificar estratégias de estimativa de QoS que sejam mais adequadas para o cenário considerado. Para permitir gerenciamento de recursos de maneira completamente automatizada é preciso também considerar as atividades relacionadas à estimativa de recursos e, consequentemente, de QoS. Dentre as técnicas utilizadas para obter essa estimativa, está a adotada no projeto mOSAIC, ou seja, estabelecer *benchmarks* customizados para cada aplicação. Outra técnica é utilizar estratégias de predição de uso para atributos de *hardware* específicos [72, 158, 273]. Dentre os trabalhos que consideram a estimativa de recursos, Wu *et al.* [268] propõe uma estratégia de mapeamento dos requisitos solicitados pelo usuário para parâmetros de infraestrutura, visando minimizar o custo da infraestrutura e violações de SLA. Contudo, a estimativa é somente com relação à quantidade de recursos de um tipo específico, assumindo também que a QoS é conhecida.

### 3.2.1 Comparativo

Nesta seção apresentamos um comparativo dos trabalhos discutidos. Embora os trabalhos considerados possam ter objetivos distintos, buscamos identificar as similaridades com nossa proposta. Nesta análise não incluímos aspectos que são relevantes apenas para composições de serviços, pois estes serão discutido na próxima seção. Na Tabela 3.2 os critérios considerados são divididos em quatro categorias: escopo, critérios de provisionamento, abstração e características não-funcionais da solução.

| | MODAClouds [89, 185] | mOSAIC [87, 212] | SMICloud [100, 101] | CloudGenius [180, 181] | Índice CSP [252] | Sun *et al.* [267, 251] | InterCloud [49, 155, 77] | Wu *et al.* [268] |
|---|---|---|---|---|---|---|---|---|
| **Escopo** | | | | | | | | |
| Estimativa de recursos | | ✓ | | | | | | |
| Descoberta de recursos | ✓ | ✓ | | | | ✓ | ✓ | |
| Seleção de recursos | ☑ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Alocação de recursos | ✓ | ✓ | | ✓ | | | ✓ | ✓ |
| Uso de nuvem privada | | ✓ | | | | | | ☑ |
| Uso de múltiplas nuvens públicas | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ☑ |
| Considera a localidade do recurso | ☑ | ☑ | ☑ | ✓ | ✓ | ✓ | ☑ | ☑ |
| **Critérios de provisionamento** | | | | | | | | |
| QoS | | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ |
| Restrições relacionadas ao ambiente de implantação (localidade do recurso, reputação do provedor, *etc.*) | | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| Minimização do custo | | ☑ | ☑ | ☑ | ☑ | | ☑ | ✓ |
| Maximização da eficiência | | | | | | | | ✓ |
| **Abstração** | | | | | | | | |
| Independência de aplicação | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Independência de ambiente de nuvem | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Manipulação de modelos de recursos | | | | | ✓ | | | |
| Isolamento da estimativa de QoS | | | | | | | | |
| **Características não-funcionais da solução** | | | | | | | | |
| Flexibilidade | ☑ | ✓ | ■ | ✓ | ■ | ■ | ✓ | ✓ |
| Manutenibilidade | ✓ | ☑ | ☑ | | ✓ | | ✓ | ✓ |
| Usabilidade | ✓ | ✓ | | | ✓ | ✓ | ✓ | ✓ |
| Escalabilidade | ✓ | ✓ | | | | ☑ | | |

***Legenda:*** ✓ Contemplado, ☑ Contemplado com limitações, ■ Fora do domínio do trabalho.

**Tabela 3.2:** *Comparativo das principais propostas para gerenciamento de recursos (considerando apenas serviços isolados).*

O *escopo*, indica alguns aspectos relevantes em uma solução que considera as principais atividades relacionadas ao gerenciamento de recursos, desde sua estimativa até a alocação do recurso no ambiente selecionado. Os *critérios de provisionamento* indicam os parâmetros que propomos que sejam usados nas decisões relacionadas ao

provisionamento de recursos. A *abstração* indica critérios que são essenciais para tratar a complexidade de ambientes de nuvem, o que inclui mecanismos para lidar de forma eficiente com a multiplicidade de tipos disponíveis e com a estimativa customizável de QoS. Por fim, as *características não-funcionais da solução* relacionam alguns critérios que caracterizam a qualidade da solução proposta: *flexibilidade* indica se há independência do mecanismo de decisão do restante da plataforma, *manutenibilidade* estabelece a capacidade de evoluir de acordo com novos cenários, *usabilidade* sugere se o provisionamento é realizado sem que o usuário tenha que fornecer informações além daquelas essencialmente necessárias e *escalabilidade* indica se a abordagem considera a adição de instâncias adicionais para os serviços como estratégia para absorver cargas elevadas. Como pode ser visto, nenhuma solução contempla todos os aspectos considerados.

Com relação ao escopo, apenas mOSAIC implementa todas as atividades que permitem uma completa automatização do gerenciamento de recursos considerando múltiplos provedores. Os demais trabalhos focam essencialmente na seleção de recursos, dado um conjunto de critérios. Grande parte das propostas assume que a estimativa e a descoberta de recursos são realizadas como atividades independentes, pelo próprio usuário ou usando alguma ferramenta desenvolvida em outro contexto. Com respeito ao ambiente de nuvem considerado, o uso de múltiplos provedores é considerado em todas as soluções, embora a exploração da heterogeneidade obtida com a disponibilidade de múltiplas regiões de um mesmo provedor seja tratada apenas de maneira superficial. Contudo, poucas soluções exploram os casos em que é usada também uma infraestrutura privada, não considerando o cenário mais comumente encontrado na maioria das organizações.

Dentre os critérios de decisão utilizados, a maioria se baseia em QoS e também permite a definição de restrições de outras categorias, como localidade do recurso. Analisamos também se o custo dos recursos é considerado nas decisões tomadas e a maioria dos trabalhos analisados considera essa variável como sendo um atributo de QoS. Embora essa categorização seja comum, consideramos que o custo dos recursos tem um impacto relevante no resultado fornecido, uma vez que o usuário em nosso cenário é o provedor de serviços, que visa maximizar o lucro ao oferecer soluções. Dessa forma, acreditamos que o custo deve necessariamente considerado, mesmo que não seja explicitamente incluído como restrição na modelagem das coreografias. Também verificamos, nos trabalhos considerados, os casos em que o provisionamento de recursos é realizado visando o uso eficiente dos recursos. Esse aspecto só é tratado de maneira eficaz na proposta de Wu *et al.* [268], que considera a consolidação de recursos. A seleção de recursos nos demais trabalhos ignora a possibilidade de haver capacidade excedente no recurso selecionado, o que faz com que a capacidade

residual não seja aproveitada.

Quanto à abstração, quase todos os trabalhos se baseiam no uso de modelos para tornar a solução independente de aplicação ou de provedor de nuvem específico. Contudo, com exceção do Índice CSP que faz processamento do modelo de recursos visando melhorar o tempo de obtenção do resultado, nenhuma outra considera alguma manipulação do modelo de recursos visando melhor desempenho ou resultados mais satisfatórios. Isso se caracteriza como uma necessidade, uma vez que avaliar todo o vasto conjunto de tipos de recurso disponíveis em um ambiente multi-nuvem pode demandar um tempo não aceitável em alguns cenários. De maneira complementar, nenhum trabalho considera tornar a estimativa de QoS algo customizável, o que pode limitar a solução nos casos em que a estimativa disponível está desatualizada ou não é adequada para a aplicação sendo implantada.

Sobre as características não-funcionais da solução, é possível verificar um bom nível de flexibilidade e manutenibilidade. Porém, a usabilidade não é ideal em todos os casos. Em algumas soluções, além da especificação dos componentes de *software* a serem implantados e restrições associadas, o usuário também deve fornecer um conjunto (muitas vezes exaustivo) de informações que são usadas como critérios de decisão, como por exemplo o peso das restrições.

## 3.3 Implantação de composições de serviços

A criação e subsequente implantação de composições de serviços em ambientes de nuvem deve considerar a seleção de serviços e a seleção de recursos. A seleção de serviços consiste em escolher instâncias concretas de serviços que implementam as tarefas abstratas definidas na composição, o que é realizado geralmente com base na QoS garantida por cada instância. Todavia, a nomenclatura adotada em ambientes de nuvem considera que qualquer entidade oferecida ao usuário (seja ela referente a *software* ou a *hardware*) é denominada como *serviço*. Em virtude disso, a seleção de serviços e de recursos são comumente agregadas em um único termo denominado *seleção de serviços*. Nesse sentido, assume-se que os recursos necessários estarão disponíveis, de forma que a QoS obtida com cada instância do serviço seja garantida. Contudo, as escolhas no provisionamento de recursos afetam diretamente a seleção de serviços [275].

Dentre os trabalhos que tratam a seleção de serviços (sem considerar explicitamente os recursos), Alrifai *et al.* [8] propõem uma estratégia para decompor restrições QoS fim-a-fim em restrições locais com limites conservadores superiores e inferiores. Essas restrições locais são então resolvidas usando uma estratégia de seleção distribuída. Wang *et al.* [260] apresentam um modelo para lidar com propriedades não-

funcionais quantitativas e qualitativas e dois algoritmos para seleção de serviços com base nesse modelo. O primeiro algoritmo combina otimização global com seleção local e o segundo consiste em um algoritmo genético. Wu *et al.* [269] propõem uma estratégia que baseia-se em selecionar serviços sob demanda, de forma a escolher aqueles que oferecem QoS mais próxima da requisitada, ao contrário da estratégia comumente adotada de selecionar serviços com maior QoS (que os autores chamam de "melhor-esforço"). Em todos estes trabalhos, ao assumir uma QoS garantida com o provisionamento de recursos eles não conseguem explorar alguns aspectos que são possíveis ao utilizar recursos como entidades de primeira classe na solução, como a variação da QoS obtida usando a elasticidade provida em ambientes de nuvem.

Ye *et al.* [275] é um dos poucos trabalhos que consideram a distinção entre seleção de serviços e de recursos. Eles propõem um algoritmo genético para fazer a seleção de recursos, no qual escolhas aleatórias são usadas para selecionar cromossomos para executar uma operação de *crossover*, tendo como base a QoS obtida sobre os recursos e a demanda de comunicação entre eles. Além da seleção de recursos, os autores também propõem uma estratégia de escalonamento de tarefas da composição, o que não é necessário no caso de coreografias, uma vez que a própria coordenação entre os serviços se encarrega dessa atividade. Uma vantagem dessa proposta é que eles consideram o provisionamento de recursos em três níveis: VM, banco de dados e rede, ao passo que em nosso modelo consideramos apenas VMs. No entanto, não está claro se essa abordagem permite a implantação visando atender a uma QoS requerida, ou apenas selecionar recursos predeterminados. Outra limitação é que a seleção de recursos é realizada para cada composição isoladamente, sem levar em consideração os efeitos causados pelo compartilhamento de serviços.

Trabalhos em áreas como escalonamento de *workflows* [276, 277] apresentam características semelhantes ao problema aqui tratado, como o objetivo de balancear a contribuição de cada serviço presente no *workflow* para satisfazer restrições fim-a-fim. Contudo, não incluímos grande parte desses trabalhos porque eles consideram algumas suposições que não fazem sentido na implantação de coreografias. A principal delas é que a composição por meio de *workflows* geralmente é estática, não contemplando todos os desafios que tratamos. Ademais, o escalonamento das tarefas é coordenado por um agente central, o que só é válido no modelo de orquestração.

QuARAMRecommender [246], parte do arcabouço QuARAM [176], é uma plataforma de seleção auto adaptativa que recomenda uma lista de recursos para implantação de aplicações em nuvem, tendo como base requisitos das aplicações e preferências dos usuários. O processo de recomendação se inicia tentando identificar em uma base de dados casos semelhantes à requisição submetida. Depois de recuperar os casos semelhantes, o sistema de recomendação envia os casos recuperados, juntamente

com aqueles especificados, para um adaptador que busca fazer modificações até que a seleção seja o mais próximo possível do que foi requisitado. Economia de recursos é outro objetivo deste trabalho. Para tal, os autores propõem consolidação de recursos como forma de diminuir o número de instâncias necessárias. A consolidação é realizada com base nas preferências do usuário e atributos de custo e desempenho dos recursos, mas sem levar em consideração a demanda de comunicação entre os serviços implantados nas entidades consolidadas. Os resultados apresentados indicam que o sistema atinge uma precisão de recomendações em 71% dos casos. Contudo, apesar deste trabalho considerar que uma aplicação pode ser formada por diferentes entidades, as formas de interação entre elas não são consideradas. Isso faz com que essa abordagem não seja ideal para coreografias, dado que há diferentes padrões de composição nesse modelo, que podem influenciar na satisfação de restrições fim-a-fim. Este trabalho também não considera os efeitos do compartilhamento de serviços, podendo inviabilizar os resultados para casos em que isso ocorre.

Charrada *et al.* [55] propõem o uso de nuvens híbridas para a implantação de serviços. Eles apresentaram um algoritmo que obtém solução próxima da ótima, tendo como base os custos de comunicação e de utilização dos recursos. O algoritmo consiste em mover serviços que não podem ser alocados na nuvem privada para nuvem pública. De maneira complementar, serviços implantados na nuvem pública são movidas para a nuvem privada quando há recursos disponíveis. O balanceamento entre os dois ambientes de nuvem também é considerado ao implantar novos serviços. Contudo, a análise realizada no algoritmo é bem limitada pois considera apenas a demanda de recurso e a minimização dos custos de comunicação, não sendo possível estabelecer outros critérios ou restrições.

Mao *et al.* [175] modelam a implantação de serviços como um jogo de congestionamento [184], onde cada serviço, formado por múltiplos componentes, é considerado como sendo um jogador que compartilha recursos com outros jogadores. A estratégia do jogador é então considerada como selecionar o subconjunto de recursos para implantar seus componentes. Com base no jogo de congestionamento, um método teórico de jogo é proposto para otimizar tanto o custo global quanto a qualidade. Para resolver o jogo de congestionamento, algoritmos são propostos pelos autores para alcançar o equilíbrio em tempo polinomial. Contudo, a solução é limitada ao uso dos critérios de tempo de execução e custo de implantação. Outra limitação é o reconhecimento dos recursos disponíveis como homogêneos, o que não retrata um ambiente real de nuvem (especialmente com múltiplos provedores).

Ye *et al.* [274] consideram o provisionamento de recursos com restrições variáveis. Um modelo econômico baseado em Redes Bayesianas [139] discretas é apresentado para caracterizar o comportamento dos usuários a longo prazo. Em seguida, o

problema de implantação de serviços com base em QoS é resolvido por Diagramas de Influência [241], seguido por experimentos analíticos e simulação. De acordo com os autores, os atributos funcionais e de QoS em uma requisição de longo prazo são variáveis. Como exemplo, o usuário pode preferir um recurso com maior desempenho no primeiro ano mas a partir do segundo pode passar a privilegiar recursos com menor custo para diminuir seus gastos. Em nossa proposta, consideramos como trabalho futuro a possibilidade do usuário fazer adaptações nas restrições para coreografias previamente implantadas. Contudo, em nossa proposta é necessário que mudanças sejam explicitamente requisitadas, pois não tratamos a variabilidade como entidade de primeira classe, da forma como é proposto por Ye *et al.* Uma limitação desse trabalho é que eles também consideram que os recursos são homogêneos.

Huang e Shen [127] propõem uma estratégia para implantação de composições de serviços visando reduzir o tempo de execução, levando em consideração os custos de comunicação entre os serviços e também paralelismo entre as tarefas executadas. A estratégia se baseia na modelagem dos custos de comunicação entre os serviços usando uma estrutura chamada grafo de dependências entre serviços (*Service Dependency Graph* – SDG), e do paralelismo entre os serviços em outra estrutura chamada grafo de concorrência de serviços (*Service Concurrence Graph* – SCG). Esses dois grafos são então integrados em uma única estrutura chamada grafo de relacionamento entre serviços (*Service Relationship Graph* – SRG), e o problema de implantação de serviços é então resolvido usando o problema $k$-corte mínimo [120]. A limitação desse trabalho é que ele não tem como objetivo satisfazer restrições estabelecidas pelo usuário, sendo que procura unicamente melhorar o tempo de execução. Apesar de considerarem a carga suportada ao realizar a consolidação de recursos, eles não resolvem o problema nos casos em que instâncias únicas dos serviços são insuficientes para atender a demanda, mantendo esse aspecto como trabalho futuro. Outra limitação é que também consideram que os recursos são homogêneos.

Amato e Moscato [9] se baseiam em soluções de uma área de pesquisa denominada orquestração de recursos (*Resource Orchestration* – RO) [262, 167], que desenvolve novos mecanismos para gerenciamento de recursos (principalmente alocação) visando QoS. Neste trabalho os autores mostraram como uma descrição baseada em padrões pode ser usada para coordenar serviços compostos, considerando todas as categorias de serviço em nuvem (SaaS, PaaS e IaaS). A metodologia explora técnicas de transformação de modelos para construir modelos formais que são usados para a análise de propriedades da orquestração. O objetivo do trabalho é a definição da linguagem formal usada para orquestrar os recursos, sendo que decisões sobre a implantação são tratadas de maneira secundária e superficial.

O componente *Enactment Engine* (EE) [159, 161], desenvolvido dentro do

contexto do projeto CHOReOS [258], consiste em um sistema de *middleware* que fornece uma plataforma como serviço (PaaS) para a implantação distribuída e automatizada de coreografias de serviços *Web* de grande escala em ambientes de nuvem. CHOReOS EE recebe uma especificação declarativa da coreografia, realiza sua implantação e devolve ao usuário informações sobre a localização de cada serviço implantado. A especificação da coreografia é uma descrição arquitetural, que deve ser desenvolvida pelo usuário que faz a implantação. A consideração de aspectos de QoS, tanto na seleção dos recursos quanto no monitoramento da aplicação, foi proposta, mas não implementada. Outra limitação é que a especificação dos recursos a serem utilizados deve ser feita pelo usuário, não havendo estimativa automática de recursos. Em nosso trabalho utilizamos este componente como parte da implementação da atividade de alocação de recursos.

### 3.3.1 Comparativo

Como nas seções anteriores, apresentamos um comparativo dos trabalhos discutidos considerando os aspectos mais relevantes do problema. De maneira especial, consideramos as mesmas categorias apresentadas na análise dos trabalhos sobre gerenciamento de recursos para serviços isolados, apresentada na seção anterior. Contudo, acrescentamos outros critérios que são exclusivos de composições de serviços. A Tabela 3.3 apresenta os resultados dessa análise.

Com relação ao escopo, são assumidas suposições semelhantes às estabelecidas nos trabalhos da seção anterior. O foco dos trabalhos considerados está na seleção de recursos sujeita a restrições locais (que se referem apenas a serviços específicos), sendo as demais atividades de gerenciamento de recursos tratadas apenas pontualmente em alguns trabalhos. Com isso, ao adotar essas abordagens o usuário deve ele próprio realizar a quebra de restrições fim-a-fim em restrições específicas sobre cada serviço para que a estratégia de seleção de recursos proposta seja utilizada; além de realizar manualmente (ou considerando outra abordagem) as demais atividades de gerenciamento de recursos.

A maioria dos trabalhos descreve recursos de maneira genérica não incluindo atributos de algum provedor ou ambiente específico. Nesse caso, pode-se assumir que a proposta teoricamente seria válida para qualquer provedor e mesmo para uma nuvem privada. Contudo, ao admitir esse tipo de abordagem e não incluir tratamentos que tem relação com os atributos do recurso concreto utilizado, a solução não pode ser considerada completa pois algumas suposições podem invalidar o resultado. Como exemplo, a quantidade limitada de recursos em uma nuvem privada faz com que a suposição de quantidade ilimitada de recursos não possa ser aplicada nesse caso.

| | Ye *et al.* [275] | QuARAMRec. [176, 246] | Charrada *et al.* [55] | Mao *et al.* [175] | Ye *et al.* [274] | Huang e Shen [127] | Amato e Moscato [9] | CHOReOS EE [159, 161] |
|---|---|---|---|---|---|---|---|---|
| **Escopo** | | | | | | | | |
| Estimativa de recursos | | ☑ | | | | | | |
| Descoberta de recursos | | | | | | | ☑ | |
| Seleção de recursos | ✓ | ✓ | ✓ | ☑ | ✓ | ☑ | ☑ | ☑ |
| Alocação de recursos | | | | | | | | ✓ |
| Restrições fim-a-fim | ✓ | | | | | | ☑ | |
| Restrições locais | ✓ | ✓ | ☑ | ☑ | ✓ | | ☑ | ☑ |
| Nuvem privada | ☑ | | ✓ | ☑ | ☑ | ☑ | ☑ | ✓ |
| Múltiplas nuvens públicas | ☑ | ✓ | ☑ | ☑ | ☑ | ☑ | ☑ | ✓ |
| Implantação conjunta de múltiplas composições | | | | ✓ | | ✓ | | |
| **Critérios de implantação** | | | | | | | | |
| QoS | ✓ | ✓ | | ☑ | ✓ | ☑ | ✓ | ☑ |
| Restrições relacionadas ao ambiente de implantação (localidade do recurso, reputação do provedor, *etc.*) | ☑ | ✓ | | | | | | |
| Localidade do recurso | | ✓ | | | | ☑ | | |
| Minimização do custo de utilização dos recursos | ☑ | ✓ | ✓ | | ✓ | | | |
| Minimização do atraso de comunicação entre os serviços | ☑ | | ✓ | | | ✓ | | |
| Maximização da eficiência | | ✓ | ✓ | ✓ | | ✓ | | |
| **Abstração** | | | | | | | | |
| Independência de aplicação | ✓ | ✓ | ✓ | ☑ | ✓ | ✓ | ✓ | ✓ |
| Independência de ambiente de nuvem | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Manipulação de modelos de recursos | | ✓ | | | | | | |
| Isolamento da estimativa de QoS | | | ■ | ■ | | ■ | ■ | ✓ |
| Considera diferentes padrões de composição (não apenas sequencial) | ✓ | | | | ✓ | ☑ | ✓ | ☑ |
| **Características não-funcionais da solução** | | | | | | | | |
| Flexibilidade | ■ | ✓ | ✓ | ■ | ■ | ■ | ■ | ☑ |
| Manutenibilidade | ✓ | ✓ | ☑ | ☑ | ✓ | ☑ | ☑ | |
| Usabilidade | ☑ | | ✓ | ✓ | | ✓ | ✓ | ✓ |
| Escalabilidade | | | ✓ | | | | | |

***Legenda:*** ✓ Contemplado, ☑ Contemplado com limitações, ■ Fora do domínio do trabalho.

**Tabela 3.3:** *Comparativo das principais propostas para gerenciamento de recursos (considerando composições de serviços).*

Com relação ao compartilhamento de serviços entre múltiplas composições e o impacto disso na implantação, apenas dois trabalhos avaliados levam esse aspecto em consideração. O grande problema em ignorá-lo é que a QoS esperada para cada serviço só será garantida se houver uma instância desse serviço dedicada para cada composição, o que pode representar desperdício devido à subutilização de algumas instâncias.

Quase todos os trabalhos consideram apenas restrições de QoS na implantação de composições, sendo o custo de utilização dos recursos considerado explicitamente só em alguns casos. A maioria dos trabalhos ignoram restrições relacionadas a

atributos do ambiente de implantação dos serviços, o que constitui uma grande limitação pois muitas das restrições atualmente impostas sobre serviços dizem respeito a essa categoria. Ademais, devido à possível demanda elevada de comunicação entre os serviços que compõem a composição sendo implantada, a seleção de recursos deveria considerar também esse aspecto, usando estratégias como a seleção de recursos em um mesmo ambiente ou com maior proximidade física possível. A localidade dos recursos alocados só é levada em consideração no QuARAMRecommender e na proposta de Huang e Shen, embora nesse último a localidade não seja tratada de maneira explícita pois se considera apenas dados já disponíveis sobre o atraso de comunicação entre os serviços.

Um dos grandes problemas encontrados é que algumas soluções consideram que os recursos são homogêneos, fazendo com que a satisfação de restrições seja limitada ao tipo adotado como padrão. Essa abordagem não é realista ao considerar um ambiente com múltiplos provedores de nuvem, onde há uma pluralidade grande de tipos. Essa pluralidade deveria ser considerada também ao propor o modelo de representação dos recursos, de forma que alguma estratégia de manipulação desse modelo seja empregada para melhorar o desempenho das análises realizadas usando ele. Estratégia desse tipo só é proposta no QuARAMRecommender.

A existência de diferentes padrões de interação entre serviços não é considerada em alguns trabalhos, sendo o tratamento nestes casos realizado somente para fluxos sequenciais. Essa abordagem torna a solução não aplicável em cenários realistas, nos quais diferentes valores de QoS podem ser obtidos ao considerar fluxos executados em paralelo ou que são acionados com base em alguma condição ou evento.

Por fim, as soluções propostas consideram apenas o uso de instâncias únicas dos serviços, com exceção da proposta de Charrada *et al.*, que aloca novas instâncias dos serviços para absorver demandas mais elevadas. A grande deficiência dos trabalhos que assumem esse posicionamento é que eles passam a ter um limite de carga suportada, acima do qual a proposta não oferece nenhuma solução.

## 3.4 Considerações Finais

Neste capítulo apresentamos os principais trabalhos relacionados à abordagem proposta. Inicialmente discutimos as propostas para modelagem de coreografias de serviços com restrições não-funcionais associadas. Grande parte das linguagens propostas são extensões a BPMN e focam na especificação de restrições de QoS impostas sobre serviços específicos. De acordo com nossa análise, além da deficiência na especificação de restrições, outro grande problema das notações atualmente disponíveis é o fato delas não permitirem uma análise imediata das dependências entre os

serviços e, principalmente, a especificação de compartilhamento de um mesmo serviço entre múltiplas composições.

Também analisamos os trabalhos sobre provisionamento de recursos, considerando duas categorias. Inicialmente avaliamos propostas para aplicações que são formadas por serviços isolados e depois consideramos trabalhos voltados para composições de serviços. As conclusões obtidas em ambas as categorias foram semelhantes. Podemos notar que o principal problema discutido é a seleção de recursos em nuvens públicas, visando atender requisitos de QoS. Na maioria dos casos assume-se que a estimativa de recursos já foi realizada e que a QoS garantida é conhecida, sendo que o uso eficiente de recursos não é geralmente um objetivo. Com relação aos critérios específicos de composições de serviços, é geralmente adotada uma simplificação do problema, assumindo-se que composições são criadas usando instâncias únicas de serviços que interagem apenas sequencialmente.

Diante dessa análise, podemos perceber que há diversas lacunas nas propostas encontradas. Essas limitações dizem respeito principalmente à ausência de uma visão mais pragmática do problema, na qual é assumida a necessidade da implantação ser realizada considerando o compartilhamento dos serviços entre composições. Para preencher essas lacunas propomos uma abordagem que leva em consideração essa questão e as demais limitações discutidas nesse capítulo, contemplando todas as atividades do gerenciamento de recursos. Essa contribuição é discutida a partir do próximo capítulo.

# Parte III

# Contribuição

CAPÍTULO 4

# Representação de múltiplas coreografias de serviços com restrições não-funcionais

Anteriormente, discutimos o estado da arte do tema trabalhado nesta tese e apresentamos, de maneira abstrata, o problema de implantação de coreografias de serviços visando satisfazer restrições não-funcionais. Conforme discutido, há alguns trabalhos que apresentam soluções relacionadas a este problema, mas sem considerar todos as necessidades desejadas.

Em nossa proposta, consideramos o problema de provisionamento de recursos na implantação de coreografias de serviços como um problema de emparelhamento. Mais precisamente, obtemos a estimativa e a seleção de recursos usando uma estratégia que realiza o mapeamento dos serviços que compõem as coreografias para os recursos disponíveis, de acordo com as restrições não-funcionais especificadas. Para tal, é necessário que todos os elementos envolvidos no problema sejam modelados usando uma representação não subjetiva e não ambígua. Em virtude disso, formalizamos os principais conceitos envolvidos na definição e na solução do problema.

Inicialmente, apresentamos a formalização da representação de serviços, recursos e restrições não-funcionais. Em seguida, propomos uma notação para a representação de coreografias de serviços e um arcabouço para a especificação de restrições não-funcionais. Com base na notação proposta, definimos uma estrutura para a representação conjunta de múltiplas coreografias de serviços, juntamente com as restrições não-funcionais associadas. A definição desta estrutura, assim como a descrição do procedimento para sua obtenção também são apresentados neste capítulo.

As variáveis usadas na formalização proposta são sumarizadas no Apêndice A.

## 4.1 Modelando uma coreografia com restrições

O principal objetivo desta seção é definir a formalização de uma notação para representação de coreografias de serviços e apresentar um arcabouço proposto para a especificação de restrições não-funcionais associadas aos serviços que fazem parte

destas coreografias. Contudo, para alicerçar a formalização desta notação e o desenvolvimento deste arcabouço, primeiramente formalizamos os elementos fundamentais relacionados ao problema tratado. Para ser mais específico, na próxima seção formalizamos a representação de serviços e recursos e, em seguida, categorizamos as restrições não-funcionais consideradas em nossa abordagem, apresentando a formalização e o processo de avaliação de restrições para cada uma das categorias definidas.

### 4.1.1 Elementos fundamentais do problema

Formalmente, as informações sobre os serviços são definidas como:

$\mathcal{S}$ – representa uma coleção de $n$ serviços $\{s_1, s_2, \ldots, s_n\}$;
$\mathcal{O}_{s_i}$ – define o papel que o serviço $s_i$, $(1 \leq i \leq n)$ pode desempenhar em uma coreografia, através da especificação do conjunto de operações que ele implementa;
$o \in \mathcal{O}_{s_i}$ – é uma operação que pode ser usada ao compor coreografias de serviços; e
$d[]_o$ – representa um vetor que contém informações usadas para estimar a demanda de recursos para o processamento de uma operação.

Limitamos a informação representada em $d[]_o$ à complexidade da operação (expressa em milhões de instruções - MI), ao uso de memória (expressa em bytes) e aos dados de saída permanentes (expresso em bytes). Para simplificar, assumimos que a largura de banda é grande o suficiente para ignorar o tempo de transmissão na comunicação entre serviços e não especificamos informações sobre o tamanho dos dados transmitidos. Assumimos que esse vetor é fornecido como metadado na especificação do serviço. Uma estratégia para automatizar a obtenção das informações nele representadas é usar ferramentas de perfilamento como VisualVM [240] e JProfiler [83], em Java. Contudo, a automação desse aspecto requer um estudo detalhado sobre os serviços coreografados e suas possíveis entradas, o que designamos como trabalho futuro.

Os dados sobre os recursos disponíveis para serem usados na implantação dos serviços são definidos como:

$\mathcal{V}$ – uma coleção de $t$ tipos de VM $\{v_1, v_2, \ldots, v_t\}$;
$\zeta[]_v$ – um vetor que especifica a capacidade de um tipo de VM $v \in \mathcal{V}$. Esse vetor contém as seguintes informações:

- capacidade de processamento: expresso em número de núcleos de CPU e velocidade de *clock* da CPU;
- memória: expressa em GB; e
- armazenamento: expresso em GB.

$\iota[]_v$ – um vetor de atributos que caracterizam um tipo de VM $v \in \mathcal{V}$, de forma que $\iota[]_v \supset \zeta[]_v$, ou seja, $\iota[]_v$ contém todos os atributos especificados em $\zeta[]_v$. Além dos atributos que descrevem a capacidade, esse vetor contém as seguintes informações:

- $c_v$ – custo associado ao tipo de VM $v \in \mathcal{V}$. Este custo representa o custo monetário de utilização por uma hora de uma instância criada sob demanda com este tipo. No caso desse tipo de VM ser instanciado em uma nuvem privada, o custo de utilização é substituído por uma proporção do custo para manter em execução a máquina física em que o tipo de VM é instanciado, que é estimado pelo seu consumo de energia.
- $\ell_v$ – localização do tipo de VM $v \in \mathcal{V}$, que equivale ao endereço da região onde esse tipo de VM é disponibilizado. Esse endereço, por sua vez, é formado por um identificador do país e por um complemento (quando disponível) que representa o estado e/ou a cidade na qual a região está instalada. De maneira alternativa, visando maior precisão, o endereço pode ser especificado usando as coordenadas geográficas (latitude e longitude), quando esta informação está disponível (por exemplo, quando o tipo de recurso é instanciado em uma nuvem privada). Contudo, quando utiliza-se uma nuvem pública não é possível obter as coordenadas de uma determinada região pois os provedores não divulgam essa informação por questão de segurança;
- $p_v$ – um identificador do provedor de nuvem que disponibiliza o tipo de VM $v \in \mathcal{V}$.

Embora $\zeta[]$ contenha as principais informações sobre a capacidade de *hardware*, reconhecemos que esta representação é uma simplificação pois alguns atributos, como por exemplo, o tamanho da *cache* e largura de banda, não são representadas. O aperfeiçoamento deste modelo, de forma a considerar um conjunto mais amplo de atributos, é tido como trabalho futuro. Argumentamos que os elementos atualmente representados são suficientes para permitir a implantação dos serviços, dadas as suposições assumidas no cenário considerado.

No problema tratado, o provisionamento de recursos deve ser implementado com o objetivo de satisfazer um conjunto de restrições não-funcionais estabelecidas pelo usuário responsável pela implantação da(s) coreografia(s). Distinguimos duas categorias de restrições, como ilustrado na Figura 4.1. Estabelecemos essa categorização com base nas restrições comumente impostas na implantação de serviços, como aquelas citadas no exemplo do cenário, apresentado no Capítulo 2.

**Restrições de QoS** descrevem dimensões qualitativas que são aplicáveis aos serviços e que são geralmente perceptíveis pelo cliente. Elas são avaliadas usando o

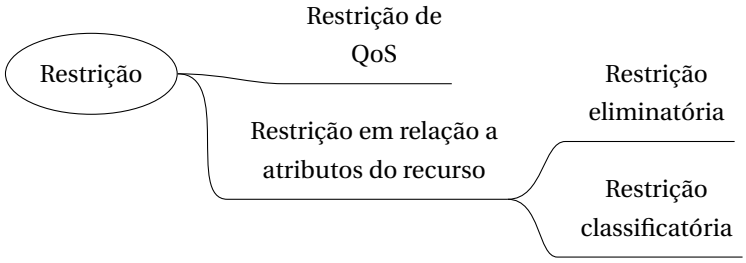



**Figura 4.1:** *Taxonomia de restrições.*

conhecimento sobre o contexto de uso, que inclui a capacidade dos recursos e a carga sobre os serviços.

**Restrições em relação a atributos do recurso** são completamente independentes da implementação dos serviços e capacidade dos recursos. Elas são especificadas usando características do ambiente de execução, que inclui informações sobre a localização da VM e seu custo, e informações sobre o provedor de nuvem, como sua reputação. Dividimos essa categoria em duas subcategorias: **restrições eliminatórias**, que são usadas para suprimir tipos de VM na seleção de recursos, usando um valor alvo para um certo atributo; e **restrições classificatórias**, que ordenam os tipos de VM usando um determinado atributo. As categorias desta taxonomia são formalizadas a seguir.

A Tabela 4.1 apresenta exemplos de restrições em cada uma das categorias estabelecidas. Essas restrições foram originalmente apresentadas no exemplo de cenário, apresentado no Capítulo 2 (Tabela 2.2).

| **Categoria** | **Exemplos de Restrição** |
|---|---|
| Restrição de QoS | • O tempo máximo necessário para atender a cada requisição deve ser 1 (um) segundo.<br>• A emissão de relatórios deve manter uma vazão de 10 (dez) requisições por segundo. |
| Restrição Eliminatória | • O recurso não pode estar indisponível mais do que 12 (doze) horas por ano.<br>• A seleção de recursos deve ser realizada de forma a evitar provedores de nuvem com obrigação de cobrança mínima.<br>• Dados sobre clientes brasileiros devem ser armazenados no Brasil.<br>• O serviço deve ser implantado em uma nuvem privada. |
| Restrição Classificatória | • A implantação de serviços deve ser realizada usando recursos com menor custo financeiro.<br>• A seleção de recursos deve privilegiar provedores de nuvem com melhor reputação. |

**Tabela 4.1:** *Exemplo de restrições em cada uma das categorias definidas.*

Restrições de QoS são avaliadas usando informações sobre a execução do serviço, de acordo com uma métrica de QoS. A definição de métrica de QoS segue a notação especificada por Rosario *et al.* [229].

**Definição 4.1** ***(Métrica de QoS)*** *Uma métrica de QoS é uma tupla* $m = (\mathbb{D}, \leq, \oplus, \bigwedge, \bigvee)$*:*

$\mathbb{D}$ *– é o domínio da métrica.*

$\leq$ *– define a ordem dos valores do domínio.*

$\oplus$ – *é uma função* $\oplus : \mathbb{D} \times \mathbb{D} \rightarrow \mathbb{D}$ *que define como QoS é incrementada. Satisfaz as seguintes condições:*

- $\oplus$ *possui um elemento neutro* 0 *que satisfaz* $\forall \ell \in \mathbb{D} \Rightarrow \ell \oplus 0 = 0 \oplus \ell = \ell$.
- $\oplus$ *é monotônico:* $\ell_1 \leq \ell'_1$ *e* $\ell_2 \leq \ell'_2$ *implica em* $(\ell_1 \oplus \ell_2) \leq (\ell'_1 \oplus \ell'_2)$.

$\bigwedge$ – *representa o limite inferior, significando que qualquer* $\ell \subseteq \mathbb{D}$ *tem um único limite inferior* $\bigwedge_\ell$. *Quando a melhor QoS é obtida de acordo com a ordem* $\leq$, *a melhor QoS possível é obtida com* $\bigwedge$.

$\bigvee$ – *representa o limite superior, significando que qualquer* $\ell \subseteq \mathbb{D}$ *tem um único limite superior* $\bigvee_\ell$. *Ao comparar dois valores da métrica, o operador* $\bigvee$ *é usado para obter a pior QoS de acordo com a ordem* $\leq$.

De acordo com esta definição, o domínio $\mathbb{D}$ considerado varia de acordo com a métrica de QoS. Como exemplo, pode ser números reais positivos $\mathbb{R}^+$ para latência, ou um domínio genérico $\mathbb{Q} \in$ {baixa, média, alta} para segurança. A ordem parcial neste domínio, definida pelo operador $\leq$, pode significar "menos é melhor" ($\leq$ em $\mathbb{R}^+$ para latência) ou "mais é melhor" ($\geq$ em $\mathbb{R}^+$ para vazão). A função de agregação $\oplus$ pode significar adição, para métricas como latência, ou a pior QoS, definida pelo operador $\bigvee$, para métricas com domínio genérico, como segurança.

Isolamos a estimativa dos valores de QoS. Este isolamento foi proposto através da definição de funções de utilidade responsáveis por obter os valores de QoS, dado um serviço e o recurso em que este serviço será implantado. O objetivo em adotar essa estratégia é permitir a generalização da técnica usada para obter os valores da métrica e, assim, permitir a personalização desta técnica de acordo com a aplicação sendo implantada. Dessa forma, uma função de utilidade pode ser definida como:

**Definição 4.2** ***(Função de utilidade)*** *Uma função de utilidade é uma função* $\mathcal{U} : \mathcal{S} \times \mathcal{V} \rightarrow \mathbb{D}_m$, *que fornece a estimativa de QoS de acordo com a métrica* $m$, *quando um serviço* $s \in \mathcal{S}$ *é implantado no recurso* $v \in \mathcal{V}$.

A estimativa de QoS é obtida de acordo com a carga agregada das operações utilizadas no serviço, considerando todas as coreografias em que ele está inserido. Isso faz com que toda a demanda ao serviço seja analisada e, assim, a sobrecarga causada pelo seu provável compartilhamento seja refletida na estimativa de QoS. Esta estratégia permite obter maior eficiência na estimativa e seleção de recursos do que abordagens que levam em consideração apenas as operações, e suas respectivas cargas, utilizadas em uma única coreografia.

A função de utilidade é implementada de maneira distinta para cada métrica de QoS. Por exemplo, para latência ou vazão podemos usar Teoria de Filas [158] para

implementar as funções de utilidade, de forma que modelos analíticos sejam usados para estimar os valores de QoS para essas métricas, de acordo com propriedades do serviço e do recurso. Por outro lado, para cada métrica de QoS pode haver diferentes funções de utilidade que a implementam.

Dada a definição de métrica de QoS e de função de utilidade, uma restrição de QoS pode ser formalmente definida como:

**Definição 4.3** ***(Restrição de QoS)*** *Uma restrição de QoS é uma tupla* $(m, \mathcal{U}, \Omega, \square, \tau)$*:*

$m$ *– é uma métrica de QoS;*

$\mathcal{U}$ *– é a função de utilidade usada para obter os valores da métrica de QoS. Usamos a notação* $\mathcal{U}_{ijk}$ *para nos referir ao valor dado pela função de utilidade usada na restrição de QoS* $k$ *quando o serviço* $i$ *é implantado no recurso* $j$*.*

$\Omega$ *– é um conjunto de serviços* $(\Omega \in \mathcal{S})$*, sobre os quais a restrição deve ser aplicada;*

$\square$ *– é um operador relacional tal que:*

- $\square \in \{<, \leq, =, \neq, >, \geq\}$ *se houver relação de ordem total em* $\mathbb{D}_m$*, ou*
- $\square \in \{=, \neq\}$ *se não houver relação de ordem total em* $\mathbb{D}_m$*; e*

$\tau$ *– é o valor alvo para a métrica* $(\tau \in \mathbb{D}_m)$*.*

Assumimos que restrições de QoS são especificadas usando métricas para as quais $\oplus$ é um operador comutativo e associativo, como $\oplus = +$ para latência. Em virtude disso, a sequência de execução dos serviços não é relevante para a implantação, sendo omitida mesmo quando $\Omega$ não é um conjunto unitário.

Utilizando a função de utilidade definida para a métrica é possível verificar a satisfação da restrição de QoS. Definimos essa informação como $x_k^q$, uma variável inteira que indica se a restrição de QoS $k$ será satisfeita:

$$x_k^q = \begin{cases} 1, & \text{se } \mathcal{U}_{11k} \oplus \ldots \oplus \mathcal{U}_{1tk} \oplus \mathcal{U}_{21k} \oplus \ldots \oplus \mathcal{U}_{2tk} \oplus \ldots \oplus \mathcal{U}_{ntk} \leq \tau; \\ 0, & \text{caso contrário.} \end{cases} \tag{4-1}$$

Na Definição 4-1, assumimos que $\mathcal{U}_{ijk}, 1 \leq i \leq n, 1 \leq j \leq t$, é avaliado como o elemento neutro da métrica quando o par referente o serviço $s_i$ e o recurso $v_j$ não for considerado na composição de valores para a restrição $k$. Como pode ser visto, o valor 1 indica que a restrição será satisfeita, ou seja, a agregação dos valores obtidos com a função de utilidade ao considerar todos os pares existentes de serviço e recurso deve estar dentro do limite imposto pelo valor-alvo da restrição. De maneira complementar, o valor 0 denota a não-satisfação da restrição.

Quanto a restrições em relação a atributos do recurso, uma restrição eliminatória é definida como:

**Definição 4.4** ***(Restrição eliminatória)*** *É representada por uma tupla* $(\Omega, \phi, \square, \tau)$*, onde:*

$\Omega$ *– é um conjunto de serviços* $(\Omega \subseteq \mathcal{S})$ *para os quais a restrição deve ser aplicada;*
$\phi$ *– é um atributo que caracteriza recursos* $(\phi \subset \iota[])$*. Usamos a notação* $\phi_v$ *para referir ao valor deste atributo para o recurso* $v$*;*
$\square$ *– é um operador relacional tal que:*

- $\square \in \{<, \leq, =, \neq, >, \geq\}$ *se houver relação de ordem total no domínio de* $\phi$*, ou*
- $\square \in \{=, \neq\}$ *se não houver relação de ordem total no domínio de* $\phi$*; e*

$\tau$ *– é o valor-alvo para esta restrição.*

De forma análoga a restrições de QoS, definimos $x_k^e$, uma variável inteira que indica se a restrição eliminatória $k$ será satisfeita:

$$x_k^e = \begin{cases} 1, & \text{se } \phi_v \square \tau \text{ é verdadeiro para um ou mais } v \in \mathcal{V}; \\ 0, & \text{caso contrário.} \end{cases} \tag{4-2}$$

Uma restrição classificatória é definida como:

**Definição 4.5** ***(Restrição classificatória)*** *É representada por uma tupla* $(\Omega, \phi, \boxminus)$*, onde:*

$\Omega$ *– é um conjunto de serviços* $(\Omega \subseteq \mathcal{S})$ *para os quais a restrição deve ser aplicada;*
$\phi$ *– é um atributo que caracteriza recursos* $(\phi \subset \iota[])$*; e*
$\boxminus$ *– é um operador de classificação:* $\boxminus \in$ *{ASCENDANT, DESCENDANT}, onde ASCENDANT indica que os recursos devem ser ordenados em ordem ascendente de acordo com o valor do atributo; e DESCENDANT indica que os recursos devem ser ordenados em ordem descendente de acordo com o valor do atributo.*

Uma vez que restrições classificatórias são usadas somente para ordenar recursos candidatos, restrições desse tipo sempre serão satisfeitas. Mesmo assim, para uso posterior na formalização, definimos $x_k^r$, uma variável inteira que indica se uma restrição classificatória $k$ será satisfeita:

$$x_k^r = \begin{cases} 1, & \text{sempre.} \end{cases} \tag{4-3}$$

Os elementos fundamentais formalizados nessa seção formam a base da notação proposta para a representação de coreografias de serviços e do arcabouço proposto para a especificação de restrições não-funcionais. Nas próximas seções iremos apresentar estas propostas.

### 4.1.2 Notação para representação de coreografias de serviços

Há na literatura e na indústria um número considerável de linguagens que foram propostas para permitir a modelagem de coreografias de serviços, das quais a maioria constitui variações de BPMN [198]. Isso ocorre porque BPMN se tornou o padrão *de facto* para notação gráfica de modelagem de processos de negócios, com ênfase no fluxo de controle em um nível independente de implementação [73].

Um dos principais objetivos de BPMN é fornecer uma notação que seja facilmente compreensível por todos os usuários envolvidos na definição do processo de negócios, sendo uma boa candidata para fornecer uma notação gráfica para modelagem de coreografias de serviços. No entanto, este nem sempre é o caso. As especificações BPMN podem ser muito complexas e a especificação padrão introduz limitações que um projetista de coreografias deve obedecer. Contudo, o padrão fornece somente uma descrição textual para estas limitações, tornando difícil seu correto entendimento [23]. Nas especificações de coreografias usando BPMN, as dependências entre os serviços são "ocultas", o que torna difícil seu uso na tomada de decisões sobre gerenciamento de recursos. Além disso, BPMN (e suas variações) não são capazes de modelar explicitamente a estrutura de uma coreografia, assim como as restrições não-funcionais impostas sobre os serviços. Por fim, o uso de BPMN como única linguagem de modelagem restringiria nossa solução a essa linguagem, tornando imperativa sua utilização.

Em virtude dos motivos descritos, e das limitações encontradas em outras propostas existentes (descritas na Seção 3.1), propomos uma nova notação para modelagem de coreografias de serviços. Esta notação tem como finalidade a representação de coreografias de serviços de forma não ambígua e permitir que a demanda sobre cada serviço, assim como as dependências entre os serviços, sejam mais facilmente identificadas e processadas. Outra motivação em adotar essa notação é que ela, juntamente com o arcabouço que será apresentado na próxima seção, permitem o vínculo dos serviços às restrições não-funcionais especificadas sobre as coreografias.

Nosso objetivo ao propor essa notação não é definir uma linguagem de propósito geral, mas a consideramos apenas como uma representação interna em nossa proposta. Sua adoção não elimina o uso de BPMN ou outra linguagem de modelagem, pois ela constitui apenas uma representação abstrata da coreografia. Dessa forma, as coreografias são inicialmente especificadas usando BPMN (ou outra linguagem com esse propósito) e então são automaticamente traduzidas para a representação interna com a inclusão das restrições associadas. Utilizamos essa representação interna para guiar as decisões sobre gerenciamento de recursos, ao passo que a encenação da coreografia continua sendo guiada pela linguagem originalmente utilizada.

Neste capítulo, definimos formalmente a notação proposta e mostramos

como transformar modelos de coreografia descritos em BPMN para modelos que usam essa representação. Apesar de consideramos BPMN como linguagem de modelagem, argumentamos que qualquer outra linguagem poderia ser usada, uma vez que um adaptador esteja disponível.

A topologia de uma coreografia de serviços é representada usando um grafo de processo [179], que é definido a seguir:

**Definição 4.6** ***(Nós predecessores e sucessores)*** *Seja $N$ o conjunto de nós e $E \subseteq N \times N$ uma relação binária sobre $N$ que define as arestas. Para cada nó $n \in N$ definimos o conjunto de nós predecessores $\bullet n = \{x \in N | (x,n) \in E\}$ e o conjunto de nós sucessores $n\bullet = \{x \in N | (n,x) \in E\}$.*

**Definição 4.7** ***(Grafo de processo)*** *Um grafo de processo $PG$ consiste em uma tupla $(b, Z, \mathcal{S}, L, E)$, onde:*

- *$b$ – denota o nó inicial, de forma que $|b\bullet| = 1$ e $|\bullet b| = 0$, ou seja, o nó inicial possui apenas um nó sucessor e nenhum nó predecessor.*
- *$Z$ – denota o conjunto de nós finais, de forma que $|Z| \geq 1$ e $\forall z \in Z : |\bullet z| \geq 1$ e $|z\bullet| = 0$, ou seja, cada nó final tem um ou mais nós predecessores e nenhum nó sucessor. Múltiplos nós finais são modelados para representar diferentes estados de finalização (sucesso, falha, etc.).*
- *$\mathcal{S}$ – denota o conjunto de serviços, de forma que $\forall s \in \mathcal{S} : |\bullet s| \geq 1$ e $|s\bullet| \geq 1$, ou seja, cada serviço tem um ou mais nós predecessores e um ou mais nós sucessores.*
- *$L$ – denota o conjunto de conectores. $|L| \equiv 0 \ (mod\ 2)$, e pode ser particionado em conjuntos disjuntos $L^s$ = {$AND^s$ (split de conjunção), $OR^s$ (split de disjunção), $XOR^s$ (split de disjunção mutuamente exclusiva)} e $L^j$ = {$AND^j$ (join de conjunção), $OR^j$ (join de disjunção), $XOR^j$ (join de disjunção mutuamente exclusiva)}, de forma que $\forall \ell^s \in L^s : (|\bullet \ell^s| = 1 \ e \ |\ell^s \bullet| > 1), \forall \ell^j \in L^j : (|\bullet \ell^j| > 1 \ e \ |\ell^j \bullet| = 1)$, $\forall \ell^s \in L^s \Rightarrow \exists \ell^j \in L^j$, ou seja, o tamanho do conjunto $L$ é um número par, conectores split são usados para separar o fluxo do processo enquanto conectores join são usados para agregar partes do fluxo do processo, e para cada conector split há um conector join equivalente. Para cada $\ell^s$ ={$XOR^s$, $OR^s$} $\in L^s$, $\ell^s\bullet = \{G_1, ..., G_n\}$, ou seja, uma requisição pode ser encaminhada para um subgrafo $G_1, ..., G_n$. Estabelecemos que há uma probabilidade $p$ para cada uma das possibilidades. Para um $OR^s$, em adição à probabilidade de cada subgrafo, há uma probabilidade $p$ da requisição ser encaminhada simultaneamente para todos os subgrafos em um dado subconjunto da combinação de $G_1, ..., G_n$ considerando $k$ elementos, onde $2 \leq k \leq n$. Como exemplo, com dois subgrafos como sucessores para um conector $OR^s$, uma requisição pode ser encaminhada para os subgrafos $G_1$, $G_2$ ou simultaneamente para ambos, com probabilidade $p_1$, $p_2$ e $p_{12}$, respectivamente.*

*E – é um conjunto de arestas que definem o fluxo como um grafo direcionado. Cada aresta* $\mathfrak{e} \in E$ *é uma tupla* $(\underrightarrow{\mathfrak{e}}, \overrightarrow{\mathfrak{e}}, \mathfrak{o})$*, onde* $\underrightarrow{\mathfrak{e}} \subseteq (b \cup \mathcal{S} \cup L)$ *é a origem da aresta,* $\overrightarrow{\mathfrak{e}} \subseteq (Z \cup \mathcal{S} \cup L)$ *é o destino da aresta, e* $\mathfrak{o}$ *é a operação sendo requisitada no serviço modelado como destino. Se* $\overrightarrow{\mathfrak{e}} \in \{Z \cup L\}$*, então* $\mathfrak{o}$ *é nula.*

Como forma de facilitar a apresentação, definimos de maneia adicional uma representação gráfica destes elementos, conforme apresentado na Tabela 4.2.

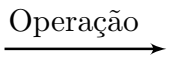
| **Elemento** | **Notação** *PG* | **Elemento** | **Notação** *PG* |
|---|---|---|---|
| Evento inicial |  | Evento final |  |
| Serviço |  | | |
| Aresta quando $\overrightarrow{e} \subseteq \mathcal{S}$ |  | Aresta quando $\overrightarrow{e} \subseteq (Z \cup L)$ |  |
| Conector *fork* de conjunção |  | Conector *join* de conjunção |  |
| Conector *fork* de disjunção |  | Conector *join* de disjunção |  |
| Conector *fork* de disjunção mutuamente exclusiva |  | Conector *join* de disjunção mutuamente exclusiva |  |

**Tabela 4.2:** *Representação gráfica da notação do grafo de processo.*

Nessa definição, restringimos o conjunto de possíveis padrões de composição a um subconjunto dos conectores usados com mais frequência na modelagem de processos de negócios (sequência, conjunção, disjunção e disjunção mutuamente exclusiva). Essa limitação foi assumida porque construções mais complexas podem, a princípio, ser obtidas a partir da combinação das construções deste subconjunto. Contudo, a comprovação da representatividade da notação proposta é tida como trabalho futuro. Para tal, propomos uma análise baseada nos padrões de interação entre serviços (*Service Interaction Patterns*) [29], um conjunto de padrões de projeto que são considerados como um *benchmark* para avaliação de linguagens de modelagem de coreografias de serviços. Eles capturam os cenários típicos de interação entre dois ou mais participantes.

Um grafo de processo representa uma única coreografia de serviços, destacando as tarefas executadas por cada serviço para alcançar o objetivo da coreografia. O principal objetivo no uso desta notação é facilitar a identificação de dependências entre os serviços e a análise da carga neles agregada.

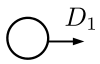
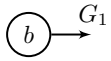
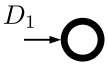
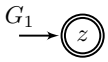
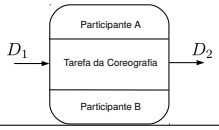
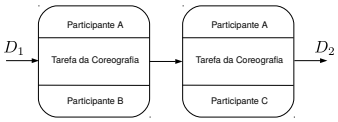
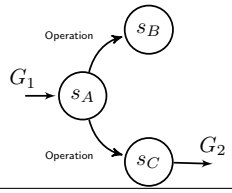
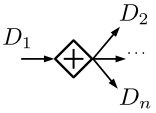
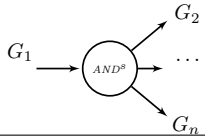
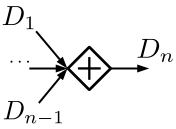
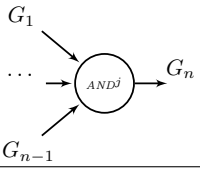
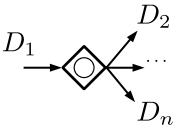
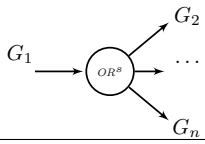
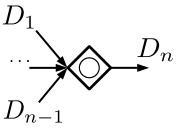
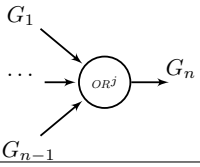
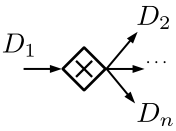
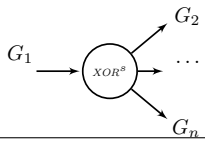
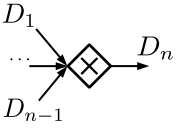
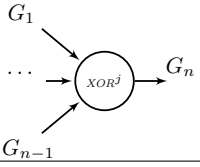
| Elemento | Descrição | Notação BPMN | Notação $PG$ |
|---|---|---|---|
| Evento inicial | Evento sem conotação especial que indica o ponto de partida |  |  |
| Evento final | Evento sem conotação especial que indica o ponto de encerramento |  |  |
| Tarefa | Representa a troca de mensagens entre dois participantes |  |  |
| Sequência de tarefas | Representa a sequência da troca de mensagens |  |  |
| Conector *fork* de conjunção | Divide o fluxo, sendo todos os fluxos de saída acionados simultaneamente |  |  |
| Conector *join* de conjunção | Combina fluxos paralelos aguardando que todos os fluxos de entrada se completem antes de acionar o fluxo de saída |  |  |
| Conector *fork* de disjunção | Divide o fluxo, e ativa um ou mais fluxos de saída |  |  |
| Conector *join* de disjunção | Aguarda que todos os fluxos de entrada ativos se completem antes de acionar o fluxo de saída |  |  |
| Conector *fork* de disjunção mutuamente exclusiva | Divide o fluxo encaminhando-o para exatamente um fluxo de saída |  |  |
| Conector *join* de disjunção mutuamente exclusiva | Aguarda que o fluxo de entrada ativo se complete antes de acionar o fluxo de saída |  |  |

**Tabela 4.3:** *Tradução dos principais elementos BPMN para a notação do grafo de processo.*

Ao traduzir modelos de coreografias descritas em BPMN para a notação de grafo de processo, limitamos a estratégia de mapeamento entre modelos a um subconjunto de elementos de BPMN, considerando aqueles elementos que são mais comumente utilizados na representação de coreografias. Além disso, a associação de conectores com valores de probabilidade foram incluídos em nosso modelo uma vez que elas são quantificações do comportamento do processo de negócio.

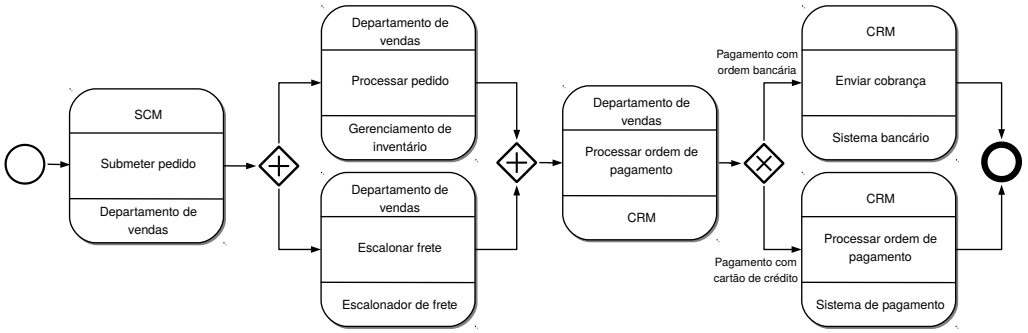
O mapeamento dos elementos BPMN em componentes do grafo de processo é compactamente apresentado na Tabela 4.3, em que $D_1, \ldots, D_n$ são subdiagramas, $G_1, \ldots, G_n$ são subgrafos, $p_i, i \in \{1, \ldots, n\}$ é a probabilidade de uma requisição ser encaminhada ao subgrafo $G_i$, e $p_{12}$ é a probabilidade da requisição ser encaminhada aos subgrafos $G_1$ e $G_2$, simultaneamente. Usando as regras de mapeamento propostas é possível implementar um adaptador que gera modelos na notação *PG* a partir de modelos em BPMN.

Conforme descrito, alguns elementos da BPMN não são representados na notação *PG*. Contudo, essa representação interna tem como objetivo fornecer uma visão estrutural da coreografia, evidenciando as dependências entre os serviços e a carga agregada em cada um deles.

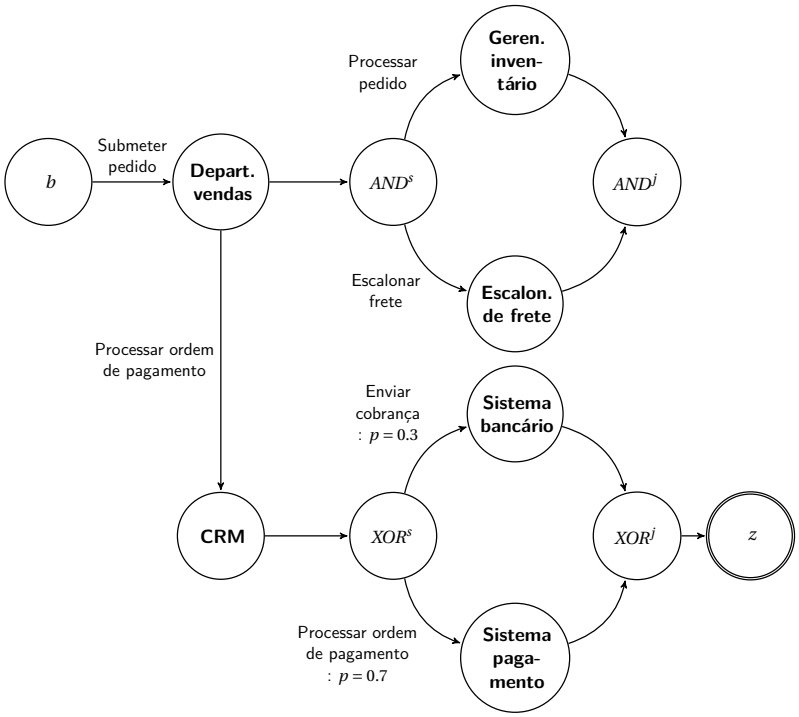
O uso do grafo de processo é restrito às atividades relacionadas à **implantação** da coreografia, propósito para o qual os elementos representados são suficientes. A encenação da coreografia ou outras atividades para as quais a notação do grafo de processo não é suficiente, podem ser realizadas usando o modelo original em BPMN (ou outra linguagem utilizada).

As Figuras 4.2 e 4.3 apresentam os grafos de processo gerados a partir dos modelos em BPMN das coreografias do exemplo de cenário, apresentado no Capítulo 2. A notação em BPMN é novamente apresentada para facilitar a correspondência entre os dois modelos.

Em um modelo de coreografia usando a notação do grafo de processo é possível obter mais facilmente a carga agregada sobre as operações requisitadas em cada serviço, assim como os padrões de interação entre eles. No entanto, é necessário estabelecer alguma forma de representação das restrições sobre as operações. Para isso, propomos um arcabouço para representação dos principais atributos das definições de restrições apresentadas na Seção 4.1.1. Além disso, o arcabouço facilita a inclusão de novos tipos de restrições na modelagem de coreografias.




(a)



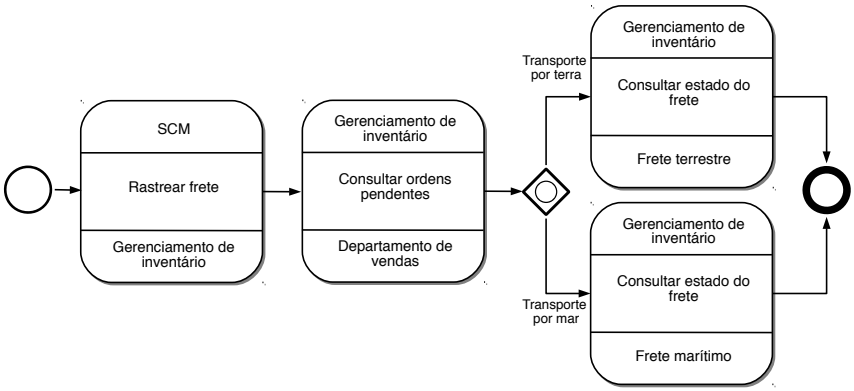

(b)

**Figura 4.2:** *Coreografia para cumprimento de pedido: (a) usando a notação BPMN. (b) usando a notação PG.*

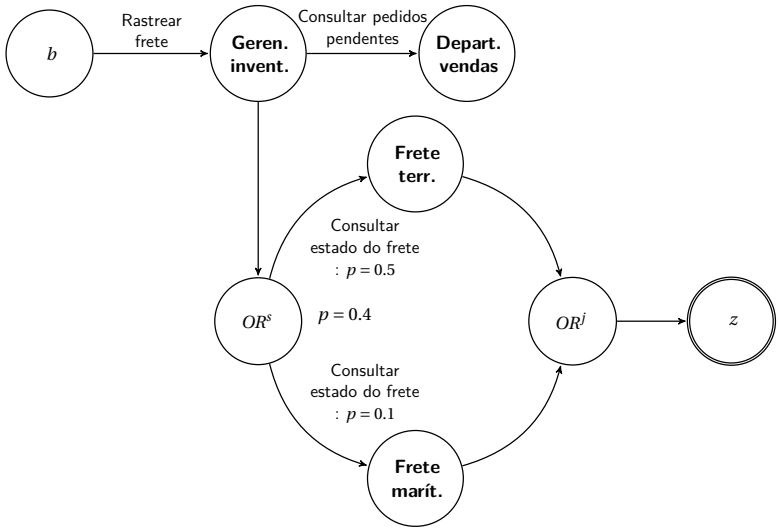
### 4.1.3 Arcabouço para especificação de restrições não-funcionais sobre serviços

No arcabouço proposto para representação de restrições não-funcionais, presumimos que as informações sobre serviços estão disponíveis em um repositório global compartilhado, sendo acessadas por meio de um identificador do serviço.

Ao propor um modelo para a representação de restrições de QoS, temos que generalizar a representação de todos os seus elementos. Um dos maiores desafios para tal é abstrair a forma pela qual os valores da função de utilidade são obtidos. Para podermos focar no problema do gerenciamento de recursos, optamos por tratar a estimativa de QoS em nossa abordagem como um problema isolado. Dessa forma,




(a)




(b)

**Figura 4.3:** *Coreografia para rastreamento de frete: (a) usando a notação BPMN. (b) usando a notação PG.*

assumimos que para cada métrica de QoS há um **estimador de QoS** que fornece a estimativa para quando um serviço for implantado em um determinado recurso.

Os valores de QoS podem ser estimados usando atributos que variam para cada métrica de QoS. Por exemplo, as estimativas da latência e da vazão são baseadas na carga das operações requisitadas, assim como na demanda de recurso para processar cada requisição. Em virtude disso, propomos que a implementação do estimador da métrica utilize um outro componente que definimos como **construtor de atributos**, o qual é responsável por obter os dados necessários para a estimativa da QoS. Esses dados podem ser obtidos usando atributos do serviço e do recurso sobre os quais a estimativa está sendo realizada, assim como atributos da carga sobre o serviço e do seu relacionamento com os demais serviços representados no grafo de dependências.

A relação dos componentes propostos para a generalização de estimativas de QoS é ilustrada na Figura 4.4. Consequentemente, para incluir uma restrição de QoS na modelagem de uma coreografia o usuário deve especificar uma implementação a

ser usada para cada um destes componentes.

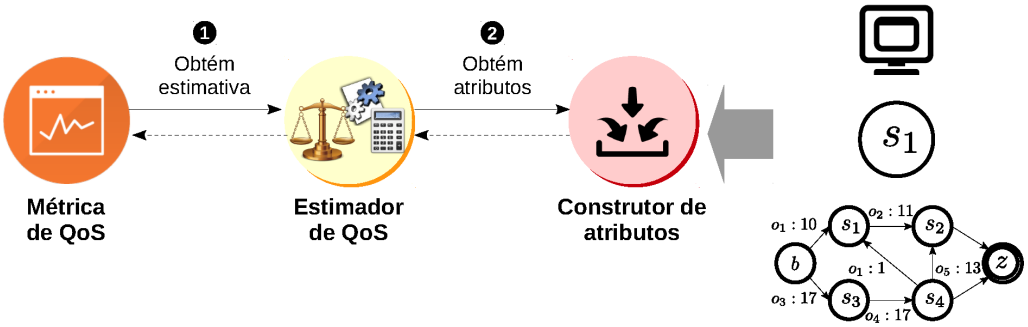



**Figura 4.4:** *Componentes para a generalização de estimativas de QoS.*

A representação de restrições eliminatórias e classificatórias pode ser implementada mais facilmente porque elas são descritas usando um conjunto predefinido de elementos e são avaliadas usando operadores predeterminados. Em virtude disso, para representar os principais elementos na definição de restrições em relação a atributos do recurso, propomos uma linguagem cuja gramática[1] é descrita no Código 4.1 na forma de Backus-Naur:

**Código 4.1** Gramática da linguagem usada na especificação de restrições em relação a atributos do recurso

| | | |
|---|---|---|
| ⟨*expression*⟩ | ::= | ⟨*term*⟩ \| ⟨*term*⟩ ⟨*or*⟩ ⟨*term*⟩ |
| ⟨*term*⟩ | ::= | ⟨*factor*⟩ \| ⟨*factor*⟩ ⟨*and*⟩ ⟨*factor*⟩ |
| ⟨*factor*⟩ | ::= | ⟨*resource att*⟩ ⟨*relational op*⟩ ⟨*comparable*⟩ |
| | \| | ⟨*rank order*⟩ ⟨*resource att*⟩ |
| | \| | '(' ⟨*expression*⟩ ')' |
| ⟨*comparable*⟩ | ::= | ⟨*number*⟩ \| ⟨*string*⟩ |
| ⟨*number*⟩ | ::= | [⟨*sign*⟩] {⟨*digit*⟩}* ['.'] {⟨*digit*⟩}$^+$ |
| ⟨*sign*⟩ | ::= | '+' \| '−' |
| ⟨*digit*⟩ | ::= | '0-9' |
| ⟨*string*⟩ | ::= | {'A-Z'}$^+$ ⟨*string*⟩ \| {'a-z'}$^+$ ⟨*string*⟩ \| $\epsilon$ |
| ⟨*or*⟩ | ::= | '\|' |
| ⟨*and*⟩ | ::= | '&' |
| ⟨*resource att*⟩ | ::= | 'LOCATION' \| 'COST' \| 'CLOUD_PROVIDER' \| ... |
| ⟨*rank order*⟩ | ::= | 'ASCENDANT' \| 'DESCENDANT' |
| ⟨*relational op*⟩ | ::= | '<' \| '<=' \| '=' \| '==' \| '!=' \| '<>' \| '>' \| '>=' |

[1]Na gramática apresentada, o conjunto de atributos que descrevem o recurso e que, consequentemente, podem ser usados para descrever restrições não-funcionais, não é apresentado completamente.

De acordo com a Definição 4.4, que descreve restrições eliminatórias, usamos a linguagem proposta para especificar $\iota$ (o atributo do recurso sobre o qual a restrição deve ser aplicada), $\square$ (um operador relacional), e $\tau$ (o valor alvo da restrição, que pode ser uma sequência de caracteres ou um número). Por simplicidade, não incluímos na linguagem a definição da unidade de medida. Por exemplo, para especificar que um certo serviço deve ser implantado em recursos localizados no Brasil, a restrição deve ser especificada como:

```
(LOCATION = Brazil)
```

Adicionalmente, de acordo com a Definição 4.5 de restrição classificatória, usamos a linguagem proposta para especificar $\iota$ (o atributo do recurso sobre o qual a restrição deve ser aplicada) e $\boxminus$ (operador de classificação, {*ASCENDANT*, *DESCENDANT*}). Por exemplo, para selecionar recursos em ordem ascendente de custo financeiro, isto é, dando preferência a recursos menos onerosos, a restrição deve ser representada com:

```
(ASCENDANT COST)
```

A linguagem proposta descreve restrições em relação a atributos do recurso usando operações lógicas na forma normal disjuntiva. O objetivo dessa representação é permitir as características descritas a seguir.

- O usuário pode especificar múltiplas restrições a serem aplicadas a um mesmo objeto. Ex.: a restrição a seguir especifica que a implantação deve ocorrer usando recursos localizados no Brasil, e cujo custo financeiro deve ser até $3.5.

```
(LOCATION = Brazil) & (COST <= 3.5)
```

- O usuário pode especificar condições alternativas a serem aplicadas a um mesmo objeto. Ex.: a restrição a seguir especifica que a implantação deve ocorrer usando recursos localizados no Brasil ou usando recursos localizados nos Estados Unidos, mas que tenham custo financeiro de até $1.0.

```
(LOCATION = Brazil) | ((LOCATION = USA) & (COST <= 1.0))
```

## 4.2 Representação de múltiplas coreografias e restrições

A satisfação de restrições em coreografias de serviços é ainda mais difícil se considerarmos o compartilhamento de serviços entre elas e os diferentes papéis que os serviços compartilhados implementam em cada uma das coreografias. Por exemplo, um serviço de mapas pode ser usado em aplicações como guias de rotas de condução e marcação de localização de imagens. Para cada uma dessas aplicações, o serviço pode ter restrições não-funcionais diferentes. Este cenário é equivalente a um dançarino participando em uma coreografia de um *mashup*[2] [109]: ele deve ser capaz de realizar a coreografia corretamente e com qualidade, lidando com prováveis diferentes ritmos de dança. No mesmo sentido, para uma dada aplicação (o que é análogo ao *mashup* em nossa metáfora) pode haver várias coreografias (ritmos em nossa metáfora), com serviços compartilhados (análogos aos dançarinos) sujeitos a diferentes restrições. Portanto, não é possível fazer um gerenciamento eficiente dos recursos sem considerar para cada serviço todas as coreografias em que ele participa.

Em nossa solução, analisamos uma coreografia de serviços usando um modelo de Rede de Filas (*Queuing Network model*) [158] no estado estável. Além disso, supomos que o gerenciamento de recursos é realizado considerando uma determinada fatia de tempo. Fazendo isso, podemos confiar em alguns resultados bem estabelecidos e assumir que o teorema de Burke [45] é válido em nosso cenário. De acordo com este teorema as chegadas de requisições seguem um processo de *Poisson* com taxa $\lambda$; e a taxa com que essas requisições são atendidas também é um processo de *Poisson* com taxa $\lambda$.

Considerando as suposições apresentadas, propomos a criação de um modelo para a representação de múltiplas coreografias de serviços. Esse modelo é gerado a partir da representação das coreografias usando a notação do grafo de processo. O objetivo em propô-lo é desenvolver uma visão conjunta de todas as coreografias, sem considerar a lógica de coreografias específicas, mas somente a dependência entre os serviços e os fatores que afetam cada operação executada.

O primeiro passo no desenvolvimento do modelo conjunto de coreografias é remover os conectores entre os serviços em cada coreografia, pois eles representam a lógica dessa coreografia específica. Nomeamos esse passo como ***redução do grafo de processo***. Para remover um conector é preciso avaliar como ele influencia na avaliação de restrições não-funcionais. Como restrições em relação a atributos do recurso e restrições de QoS sobre serviços específicos são avaliadas considerando cada serviço

[2] Um *mashup* é uma canção ou composição criada a partir da mistura de duas ou mais canções pré-existentes, normalmente pela transposição do vocal de uma canção em cima do instrumental de outra, de forma a se combinarem [263].

isoladamente, os conectores só tem influência na avaliação de restrições de QoS fim-a-fim. Nestes casos, a avaliação de cada conector é realizada usando o operador de agregação e a função de utilidade das métricas de QoS em questão, de acordo com as regras apresentadas na Tabela 4.4, onde $G_1, G_2, \ldots, G_g$ são subgrafos do grafo de processo que representa a coreografia.

| **Padrão** | **Regra de composição** |
|---|---|
| Sequência | $\mathcal{U}(G_1) \oplus \mathcal{U}(G_2)$ |
| $XOR^s$ / $XOR^j$ | $\bigoplus_{i=1}^{g} p_i \mathcal{U}(G_i)$ |
| $AND^s$ / $AND^j$ | $\bigvee\{\mathcal{U}(G_1), \mathcal{U}(G_2), \ldots, \mathcal{U}(G_g)\}$ |
| $OR^s$ / $OR^j$ | $\oplus_{i=1}^{g} p_i \mathcal{U}(G_i) \oplus \bigoplus_{k=2}^{n} p_c \bigvee\{\mathcal{U}(G_1), \ldots, \mathcal{U}(G_k)\}$, onde $c = \begin{pmatrix} g \\ k \end{pmatrix}$ |

**Tabela 4.4:** *Regras de composição de valores de QoS fim-a-fim.*

O operador $\oplus$ define como a QoS é incrementada por cada novo evento/operação (para uma coreografia). O valor composto da função de utilidade para subgrafos conectados sequencialmente é obtido agregando o valor de utilidade de cada subgrafo usando $\oplus$. Para o padrão de disjunção mutuamente exclusivo é utilizada uma estratégia semelhante, mas aplicando a probabilidade de execução de cada subgrafo no seu valor de utilidade. Para o padrão de conjunção, o paralelismo deve ser levado em conta. Neste caso, somente a pior QoS é tomada entre os ramos executados em paralelo usando o operador $\bigvee$ definido na métrica. Para o padrão de disjunção, o valor composto da função de utilidade é calculado agregando os valores da função de utilidade, considerando cada subgrafo multiplicado pela probabilidade de executar apenas este subgrafo. Além do valor de utilidade na execução de cada subgrafo isolado, a execução paralela de subgrafos também deve ser incluída neste caso. Isso é realizado restringindo o pior valor de QoS com a probabilidade de executar os subgrafos em paralelo. Esse valor é calculado para cada combinação de todos os subgrafos tomados $k$ a $k$ ($k = \{2, \ldots, g\}$, em que $g$ é o número de subgrafos).

Com base nas regras de composição apresentadas, ao fazer a redução de um grafo de processo, considerando o padrão de sequência, é suficiente especificar a carga esperada para cada operação. O mesmo é verdadeiro para o padrão de disjunção mutuamente exclusiva se aplicarmos sobre a carga a probabilidade. Contudo, para os padrões de conjunção e disjunção é preciso preservar a informação sobre a execução paralela, já que apenas indicar a carga não é suficiente para compor a QoS final.

Como exemplo, considere o fragmento da coreografia de rastreamento de frete, apresentado na Figura 4.5(a). Supondo que a carga $\lambda$ é constante de 100 requisições por unidade de tempo, ao reduzir esse subgrafo para eliminar os conectores $OR^s$ / $OR^j$, levando em conta apenas a aplicação da probabilidade sobre esta carga, o resultado será o grafo apresentado na Figura 4.5(b), uma vez que a carga sobre o

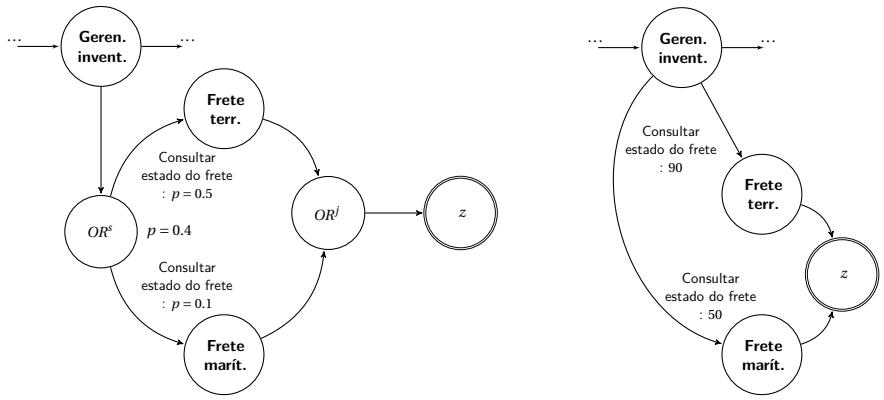



**Figura 4.5:** *Exemplo para ilustrar a necessidade de representação de paralelismo: (a) fragmento da coreografia de rastreamento de frete e (b) sua redução através da eliminação dos conectores.*

serviço "Frete terrestre" será $p_{\text{Frete terrestre}} \times \lambda + p_{\text{ambos}} \times \lambda = 0.5 \times 100 + 0.4 \times 100 = 90$ e sobre o serviço "Frete marítimo" será $p_{\text{Frete marítimo}} \times \lambda + p_{\text{ambos}} \times \lambda = 0.1 \times 100 + 0.4 \times 100 = 50$. Contudo, a informação sobre a quantidade real de carga executada em paralelo não é apresentada no subgrafo gerado, levando à interpretação errônea de que 50 requisições são executadas em paralelo. Isso faria com que a QoS composta fosse computada como[3]: $0.5 \bigvee \{\mathcal{U}_{\text{Frete terrestre}}, \mathcal{U}_{\text{Frete marítimo}}\} \oplus 0.4\mathcal{U}_{\text{Frete terrestre}}$, em vez de $0.4 \bigvee \{\mathcal{U}_{\text{Frete terrestre}}, \mathcal{U}_{\text{Frete marítimo}}\} \oplus 0.5\mathcal{U}_{\text{Frete terrestre}} \oplus 0.1\mathcal{U}_{\text{Frete marítimo}}$, como deveria.

Para contornar o problema apresentado, definimos outra estrutura para auxiliar na representação das requisições que devem ser avaliadas em paralelo. Denominamos essa estrutura como **árvore de paralelismo**.

**Definição 4.8** ***(Árvore de Paralelismo)*** *Uma árvore de paralelismo $PT$ é um grafo acíclico conectado representado por uma tupla $(b, \mathcal{L}, \mathcal{S}', E)$, em que:*

- *$b$ – denota o nó raiz, $|b\bullet| \geq 2$ e $|\bullet b| = 0$, ou seja, o nó raiz tem dois ou mais nós sucessores e nenhum nó predecessor.*
- *$\mathcal{L}$ – denota o conjunto de descritores de carga, $b \in \mathcal{L}$, $|\mathcal{L}| \geq 1$ e $\forall l \in \mathcal{L} : l = c$, onde $c$ é uma constante e $\forall l \in \mathcal{L} : l = b \rightarrow |\bullet l| = 0$ e $l \neq b \rightarrow |\bullet l| = 1$ e $|l\bullet| \geq 2$, ou seja, descritores de carga tem dois ou mais nós sucessores; não tem nó predecessor se este for o nó raiz e tem necessariamente um nó predecessor nos outros casos.*
- *$\mathcal{S}'$ – denota o conjunto de serviços, em que $\forall s \in \mathcal{S}' : s \in \mathcal{S}$ e $|\bullet s| = 1$ e $0 \leq |s\bullet| \leq 1$, ou seja, serviços têm um nó predecessor e tem um ou nenhum nó sucessor (neste último caso são os nós folha).*

[3] Nesse exemplo os índices $j$ e $k$ na definição da função de utilidade $\mathcal{U}_{ijk}$ foram omitidos para facilitar a leitura.

*$E$ – é um conjunto de arestas. Cada aresta $\mathfrak{e} \in E$ é uma tupla $(\underrightarrow{\mathfrak{e}}, \overrightarrow{\mathfrak{e}}, \mathfrak{o})$, onde $\underrightarrow{\mathfrak{e}} \subseteq \mathcal{L} \cup \mathcal{S}'$ é o nó origem da aresta, $\overrightarrow{\mathfrak{e}} \subseteq \mathcal{L} \cup \mathcal{S}'$ é o nó destino da aresta, e $\mathfrak{o}$ é a operação sendo requisitada quando $\overrightarrow{\mathfrak{e}}$ é um serviço ou nulo quando $\overrightarrow{\mathfrak{e}}$ é uma constante e $\forall \mathfrak{e} \in E : \underrightarrow{\mathfrak{e}} \in \mathcal{L} \rightarrow \overrightarrow{\mathfrak{e}} \in \mathcal{S}'$ e $\overrightarrow{\mathfrak{e}} \in \mathcal{L} \rightarrow \underrightarrow{\mathfrak{e}} \in \mathcal{S}'$, ou seja, descritores de carga não podem ser conectados com outros descritores de carga.*
*O fato de ser um grafo simples implica que $\forall v \in \mathcal{L} \cup \mathcal{S}' : (v, v) \notin E$ (sem arestas reflexivas) e que $\forall x, y \in \mathcal{L} \cup \mathcal{S}' : |\{(x, y) | (x, y) \in E\}| = 1$ (sem múltiplas arestas para um mesmo par).*

A Figura 4.6 ilustra fragmentos de grafos de processo e a árvore de paralelismo equivalente usando uma notação gráfica. Como pode ser visto, cada subgrafo que contém operações executadas em paralelo é representado como uma árvore cujo nó raiz é a carga e os ramos são os subgrafos ligados pelo conector de conjunção.

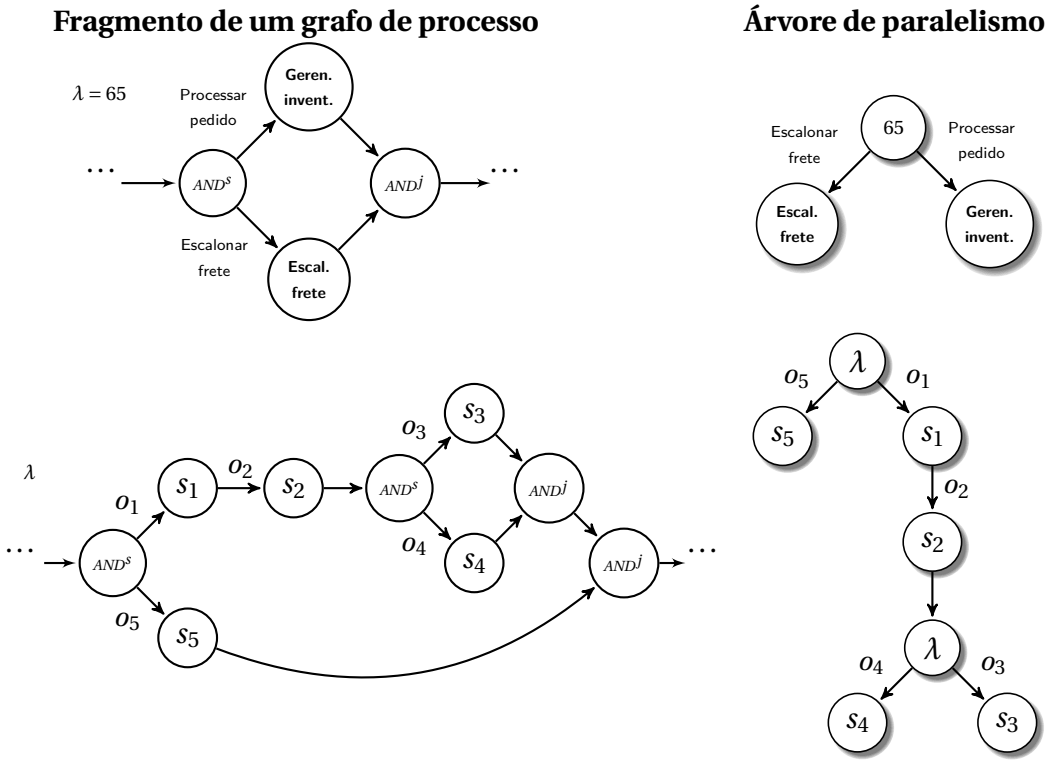



**Figura 4.6:** *Exemplos de subgrafos de processo e equivalentes árvores de paralelismo.*

Usando a estrutura de árvore de paralelismo para preservar a informação sobre operações executadas em paralelo, é possível realizar a redução de um grafo de processo de forma a eliminar todos os conectores. A Tabela 4.5 ilustra como isso é realizado para cada padrão de composição, em que $\lambda$ é a carga de entrada e $PT$ é a árvore de paralelismo gerada na redução.

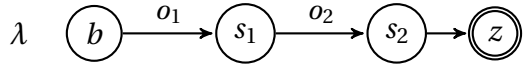
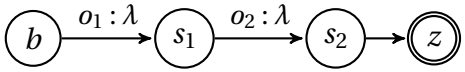
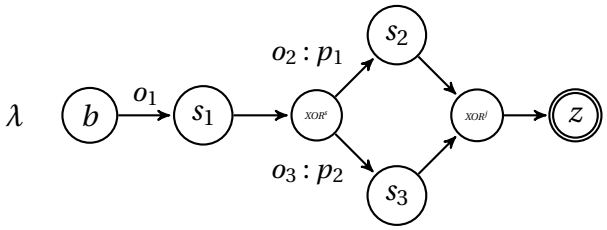
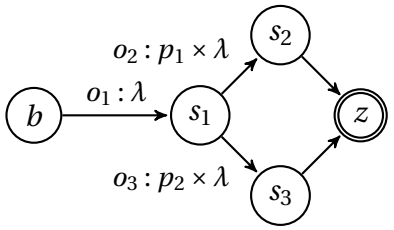
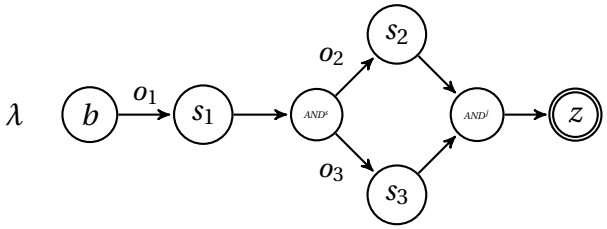
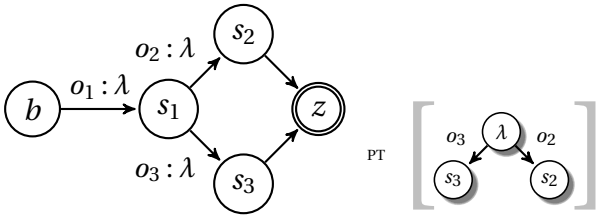
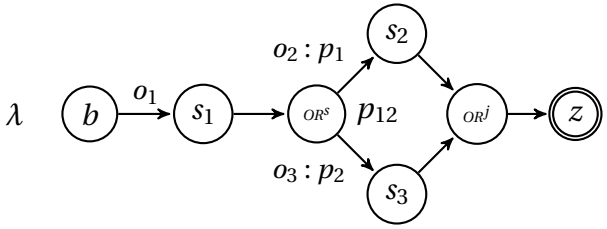
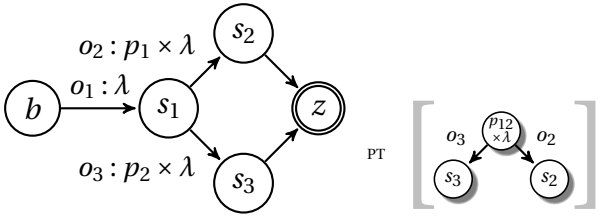
| Padrão | Exemplo de grafo de processo | Redução |
|---|---|---|
| Sequência |  |  |
| Disjunção mutuamente exclusiva |  |  |
| Conjunção |  |  |
| Disjunção |  |  |

**Tabela 4.5:** *Redução de grafos de processo de acordo com o padrão de composição.*

Embora árvores de paralelismo sejam usadas para especificar quais operações devem ser executadas em paralelo, a simples existência dessa estrutura não é suficiente para representar essa informação de maneira completamente adequada. O principal motivo é que os serviços descritos em uma árvore de paralelismo podem ser usados em outras coreografias (ou inclusive na mesma coreografia em outro ponto da composição) através de outros padrões de composição e/ou sujeitos a outra carga. Desta forma, representar árvores de paralelismo isoladas torna impossível saber em qual restrição ela deve ser aplicada, uma vez que os conectores são eliminados na redução do grafo de processo. Devido a isso, as árvores de paralelismo devem estar relacionadas com a restrição para a qual o paralelismo deve ser aplicado.

Especificamos outro conceito, denominado **Grupo em Restrição**, para representar a restrição juntamente com as informações de paralelismo necessárias para avaliar restrições de QoS fim-a-fim.

**Definição 4.9** ***(Grupo em Restrição)*** *Um grupo em restrição $\mathcal{K}$ é representado por um par $(k, \Lambda)$, em que:*

*$k$ – é uma restrição; e*

*$\Lambda$ – é um conjunto de árvores de paralelismo relacionadas à restrição. $\Lambda = \emptyset$ quando $k$ é uma restrição de QoS cujos serviços não devem ser avaliados em paralelo, ou quando $k$ é uma restrição em relação a atributos do recurso.*

A descrição dos grupos em restrição é representada internamente incluindo, quando necessário, a informação sobre as árvores de paralelismo na descrição da restrição. Isso é realizado automaticamente, levando em consideração a carga esperada e a topologia da coreografia descrita no grafo de processo.

Para agregar todas as coreografias de serviços de entrada em uma estrutura única, é necessário decidir como tratar os grupos em restrição gerados. A junção de dois grupos em restrição apenas pode ser realizada se eles são *grupos equivalentes*, isto é, possuem exatamente os mesmos atributos, exceto pelo valor alvo. Caso essa condição não seja verificada, ambos os grupos em restrição devem ser mantidos na estrutura conjunta, uma vez que eles têm escopos diferentes.

Assumimos que o tratamento de conflito está fora do nosso escopo e quando restrições conflitantes são detectadas uma mensagem de erro é enviada ao usuário, sendo ele/ela responsável por resolver o conflito optando por uma das restrições conflitantes. Como exemplo, suponhamos que o usuário submeta duas coreografias. Suponhamos também que, devido à localidade dos clientes, o usuário requisite que os serviços da primeira coreografia sejam implantados usando recursos localizados no Brasil, e que os serviços da segunda coreografia sejam implantados usando recursos localizados nos Estados Unidos. Caso haja compartilhamento de serviços entre estas

duas coreografias, haverá um conflito, tendo o usuário que escolher qual das duas localidades será usada na implantação dos serviços compartilhados.

A junção de grupos envolvendo restrições de QoS deve considerar como o valor alvo é tratado nas métricas equivalentes. Ao considerar duas restrições de QoS definidas pela mesma métrica, a junção destas restrições pode considerar o valor alvo mais restritivo (como para latência), ou a composição dos valores alvo (como para vazão), ou ainda a intersecção de conjuntos quando o valor alvo é definido usando grupos de valores (como para segurança, quando são definidos diferentes níveis aceitáveis). Dessa forma, uma vez que a nossa abordagem propõe a abstração da representação da métrica de QoS, é preciso abstrair também a forma como é feita a junção dos valores alvos. Para tal, definimos a função a seguir.

**Definição 4.10** ***(Função de junção de valores alvos)*** *É uma função* $\odot : \mathbb{D}_m \times \mathbb{D}_m \rightarrow \mathbb{D}_m$*, definida de maneira específica para a métrica* $m$*, que obtém o valor resultante da junção de dois valores alvos estabelecidos sobre o domínio* $\mathbb{D}$ *desta métrica.*

A função $\odot$ deve ser modelada para cada métrica de QoS e usada no processo de junção de grupos equivalentes. Por exemplo, no caso da métrica se referir à latência, $\odot = \bigwedge$, significando que para juntar dois grupos equivalentes restritos pela métrica de latência é suficiente manter o melhor valor alvo de latência, dado que o outro objetivo é obtido como consequência. Como outro exemplo, para a métrica vazão, $\odot = +$, significando que para juntar dois grupos equivalentes restritos pela métrica vazão, é necessário estabelecer como novo valor alvo a soma dos valores alvos anteriores.

Para restrições eliminatórias, também ocorre junção de grupos equivalentes quando o atributo representa uma variável numérica. Neste caso, a junção também é realizada conservando o valor alvo mais restrito, que é determinado pelo operador relacional usado.

Nos casos em que o valor alvo é uma sequência de caracteres ou quando a restrição é classificatória, a junção só ocorre para grupos exatamente iguais (grupos equivalentes com mesmo valor alvo). Assim como em restrições de QoS, assumimos que não há tratamento de conflito para restrições em relação a atributos do recurso.

Aplicando as técnicas de redução do grafo de processo e junção de restrições, geramos a estrutura para representação de um conjunto de coreografias. Essa estrutura conjunta é chamada de ***grafo de dependências sujeito a restrições***, ou simplesmente ***grafo de dependências***.

**Definição 4.11** ***(Grafo de dependências sujeito a restrições)*** *Um grafo de dependências sujeito a restrições DG é um grafo direcionado representado por uma tupla* $(b, z, \mathcal{S}, E, \mathbb{K})$*, em que:*

$b$ – *denota o nó inicial,* $|b\bullet| = 1$ *e* $|\bullet b| = 0$*, ou seja, o nó inicial possui apenas um nó sucessor e nenhum nó predecessor.*

$z$ – *denota o nó final,* $|\bullet z| \geq 1$ *e* $|z\bullet| = 0$*, ou seja, o nó final tem um ou mais nós predecessores e nenhum nó sucessor.*

$\mathcal{S}$ – *denota o conjunto de serviços,* $\forall s \in \mathcal{S} : |\bullet s| \geq 1$ *e* $|s\bullet| \geq 1$*, ou seja, serviços tem um ou mais nós predecessores e um ou mais nós sucessores.*

$E$ – *é um conjunto de arestas que definem as dependências como um grafo direcionado. Cada aresta* $\mathfrak{e} \in E$ *é uma tupla* $(\underrightarrow{\mathfrak{e}}, \overrightarrow{\mathfrak{e}}, \mathfrak{o}, \lambda)$*, onde* $\underrightarrow{\mathfrak{e}} \in (b \cup \mathcal{S})$ *é a origem da aresta,* $\overrightarrow{\mathfrak{e}} \in (z \cup \mathcal{S})$ *é o destino da aresta,* $\mathfrak{o}$ *é a operação sendo requisitada, e* $\lambda$ *é a carga de requisições para a operação. Tem-se que* $\overrightarrow{\mathfrak{e}} = z \rightarrow \{\mathfrak{o}$ *é nulo e* $\lambda$ *é nulo*$\}$*,* $\overrightarrow{\mathfrak{e}} \in \mathcal{S} \rightarrow \{\mathfrak{o} \in \mathcal{O}_{\overrightarrow{\mathfrak{e}}}$ *e* $\lambda$ *é a carga estimada de acordo com a redução do grafo de processo*$\}$*.*

$\mathbb{K}$ – *é um conjunto de grupos em restrição.*

O Algoritmo 4.1 implementa a geração de um grafo de dependências a partir de um conjunto de grafos de processo. Seu funcionamento se baseia em analisar cada coreografia (ou seja, o grafo de processo que a representa) por vez, adicionando novos vértices e arestas no grafo de dependências de maneira incremental e fazendo a junção das restrições quando possível.

O algoritmo inicia com a eliminação de conectores através da redução do grafo de processo (linha 4). A operação `reduce` é responsável por eliminar conectores no grafo de processo e (se necessário) construir árvores de paralelismo a serem incluídas na descrição das restrições de QoS especificadas para a coreografia modelada. O resultado é um conjunto de grupos em restrição. Cada aresta no grafo de processo reduzido é então avaliada para verificar se ela contém algum vértice ainda não incluído no grafo de dependências resultante, de forma a haver apenas um único nó inicial e um único nó final.

Caso a aresta contenha um desses nós para o grafo de processo, ele é então rotulado como o nó equivalente no grafo de dependências (linhas 6 e 9). Em seguida, caso não exista aresta no grafo de dependências equivalente, a aresta que está sendo avaliada é adicionada juntamente com a carga propagada para a operação em questão. Caso já exista uma aresta equivalente no grafo de dependências, à carga nela contida é adicionada a carga propagada da aresta sendo avaliada. No grafo de dependências gerado, a carga isolada de cada coreografia não é mantida, uma vez que ela é transposta para as arestas do grafo.

**Algoritmo 4.1:** Geração de um grafo de dependências a partir de um conjunto de grafos de processo

**Entrada:** Conjunto de grafos de processo $PG[]$.
**Saída:** Grafo de dependências $DG$.

```
1  dg.addNode(b)
2  dg.addNode(z)
3  para cada pg_i em PG[] faça
4      K_i ← reduce(pg_i)
5      para cada e ∈ E_{pg_i} faça
6          se e→ = b_{pg_i} então
7              e→ ← b
8          fim
9          se e⃗ ⊆ Z_{pg_i} então
10             e⃗ ← z
11         fim
12         se DG não contém vértice e→ então
13             DG.addNode(e→)
14         fim
15         se DG não contém vértice e⃗ então
16             DG.addNode(e⃗)
17         fim
18         se DG não contém aresta (e→, e⃗, o) então
19             DG.addEdge(e→, e⃗, o, λ)
20         senão
21             Incremente a carga da aresta com λ (se não nulo)
22         fim
23         se conflict(DG.K, K_i) então
24             retorna CONFLICT_ERROR
25         fim
26         mergeConstraintGroups(DG.K, K_i)
27     fim
28 fim
29 retorna dg
```

A ação seguinte no algoritmo é avaliar se há conflito entre os grupos em restrição gerados com aqueles já existentes no grafo de dependências (linha 23). Caso algum conflito seja detectado, uma mensagem de erro é enviada ao usuário, para

que ele possa solucionar o conflito. A última etapa no algoritmo é realizar a junção das restrições conforme discutido anteriormente (linha 26). A saída da operação `mergeConstraintGroups` são as restrições combinadas juntamente com aquelas que permaneceram isoladas.

A Figura 4.7 apresenta o grafo de dependências gerado através desse algoritmo ao considerar as coreografias do estudo de caso. Neste caso, assumimos que a carga ($\lambda$) é constante de 30 requisições na coreografia para cumprimento de pedido e de 80 requisições na coreografia para rastreamento de frete. Para tornar o exemplo compacto, a representação das restrições não é apresentada.

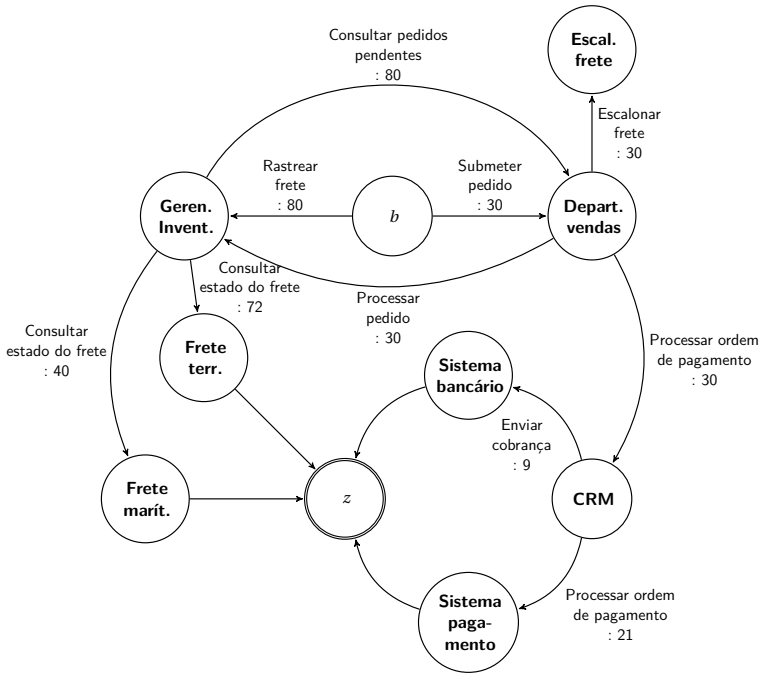



**Figura 4.7:** *Grafo de dependências gerado a partir das coreografias (grafos de processo) das Figuras 4.2(b) e 4.3(b).*

## 4.3 Considerações finais

As principais dificuldades encontradas na implantação de composições de serviços estão relacionadas à estimativa e à seleção de recursos. Isso se dá porque as restrições não-funcionais requeridas têm que ser analisadas face a um conjunto vasto de opções de recursos. Essa tarefa é ainda mais complexa ao considerar múltiplas composições com compartilhamento de serviços, visto que a contribuição dos serviços compartilhados deve ser analisada considerando todas as composições. Além disso, as restrições fim-a-fim têm que ser eficientemente balanceadas entre os serviços a que elas se referem. Essas dificuldades motivaram o tratamento do problema discutido nessa tese, cujos principais elementos foram formalizados nesse capítulo.

Inicialmente, apresentamos a definição de serviços e de recursos, juntamente com uma categorização dos tipos de restrições não-funcionais que consideramos na especificação de coreografias. A formalização desses elementos teve como objetivo definir de maneira precisa os atributos que incluímos no escopo do problema e generalizar a solução que foi desenvolvida, permitindo, assim, o tratamento do problema de forma mais ampla.

Apesar da existência de várias linguagens para modelagem de coreografias de serviço, nenhuma delas consegue expressar restrições não-funcionais no nível que propomos. Em virtude disso, e devido a outros motivos, como independência e tratabilidade do modelo da coreografia, propomos uma abordagem para representação de coreografias de serviços sujeitas a restrições não-funcionais.

A notação proposta também foi apresentada de maneira formal neste capítulo, comparando-a com a linguagem BPMN. Em complemento à notação proposta, propomos um arcabouço que abstrai a especificação das restrições não-funcionais.

Tendo como base a notação proposta, apresentamos uma estratégia para a representação conjunta de múltiplas coreografias. Descrevemos a estrutura proposta juntamente com o procedimento para obter tal representação. A geração de um modelo conjunto das coreografias tem como objetivo estabelecer uma visão singular dos serviços a serem implantados e, dessa forma, tornar explícitas as dependências entre eles, facilitando a identificação dos efeitos de seu provável compartilhamento.

A Figura 4.8 resume as etapas realizadas para gerar a representação conjunta proposta. Em cada etapa são apresentadas as notações utilizadas na representação do modelo das coreografias e das restrições não-funcionais, bem como as tarefas desempenhadas para gerar a representação de saída. Como pode ser visto, as coreografias são inicialmente fornecidas em BPMN, ou outra linguagem com este propósito, e transformadas gradativamente até que o modelo conjunto, representado pelo grafo de dependências, seja gerado.

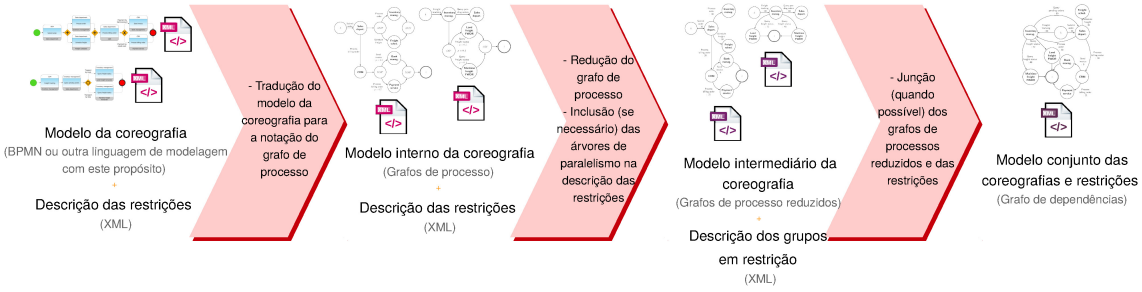



**Figura 4.8:** *Etapas na geração da representação combinada de múltiplas coreografias.*

Além da representação conjunta das coreografias e das restrições, o provisionamento de recursos se baseia em um modelo que representa os recursos disponíveis.

Dessa forma, a representação conjunta das coreografias e restrições não-funcionais, proposta neste capítulo e o modelo de recursos formam a base para o processo de síntese de recursos, que consiste em estimar a capacidade necessária e selecionar o recurso ideal para implantar cada serviço. No próximo capítulo, apresentamos nossa proposta para criação do modelo de recursos, ao passo que o procedimento de síntese será discutido no Capítulo 6.

CAPÍTULO 5

# Modelagem de recursos

Uma estratégia para auxiliar na seleção de recursos ao implantar uma composição de serviços é decompor restrições fim-a-fim em restrições locais. Neste caso, as decisões são tomadas de forma isolada para cada serviço, sem considerar os demais. No entanto, essa estratégia restringe excessivamente as decisões sobre a seleção de recursos, uma vez que soluções possíveis com outras decomposições serão desconsideradas, embora elas possam representar escolhas mais eficientes. Para superar esta limitação, propomos uma abordagem holística de decomposição das restrições fim-a-fim, baseada numa estratégia de emparelhamento de serviços e recursos. Em virtude disso, é necessário que os tipos de recurso disponíveis sejam explicitamente representados.

A representação de recursos proposta em nossa abordagem considera a intersecção existente entre os tipos de VM disponíveis em provedores de nuvem pública e o fato de que diferentes tipos podem satisfazer uma dada demanda. Dessa forma, propomos a modelagem de recursos que utiliza um modelo auxiliar com a representação de somente instâncias representativas dos tipos disponíveis. Neste capítulo descrevemos a motivação para adotar essa estratégia e apresentamos o procedimento para gerar a representação proposta. Como forma de ilustrar a técnica apresentada e instrumentar os experimentos para avaliação da proposta que serão discutidos adiante (no Capítulo 8), geramos o modelo de recursos usando os tipos disponíveis em alguns provedores de nuvem relevantes.

## 5.1 Tipos de recurso

A representação dos tipos de recurso é utilizada como entrada nas atividades de estimativa e seleção de recursos. A estimativa de recursos depende somente do poder computacional expresso pela capacidade de *hardware*, ao passo que a seleção de recursos pode envolver outros atributos, como custo, localidade, provedor de nuvem que disponibiliza o tipo, dentre outros.

A caracterização de tipos de recurso adotada em provedores de nuvem pública especifica cada tipo de recurso em função apenas de seus atributos de *hardware*. Com base na especificação do tipo de recurso, a cobrança é realizada de acordo com a região escolhida para instanciar este tipo, sendo assim custo e localidade variáveis.

Em virtude das características descritas acima, a modelagem de recursos proposta em nossa abordagem considera dois níveis de representação. No primeiro nível são descritos todos os atributos que caracterizam um tipo de recurso em nossa modelagem, ao passo que o segundo nível da representação considera apenas os atributos de *hardware*. Por simplicidade, nos referimos a estes dois níveis unicamente como ***modelo concreto de recursos***, sendo que instâncias desse modelo são chamadas de ***tipos concretos*** de recurso ou simplesmente ***tipos de VM***. A adoção desses dois níveis de representação visa fazer com que a estimativa de recursos seja realizada de maneira mais eficiente, pois, dessa forma, não necessitará levar em consideração múltiplas representações de uma mesma capacidade de *hardware*, ocasionadas pela repetição de um tipo de recurso nas regiões disponibilizadas por um mesmo provedor.

Uma vez que cada provedor de nuvem usa atributos de *hardware* diferentes para caracterizar os tipos de recurso disponibilizados, generalizamos a representação usando apenas os atributos mais comumente encontrados: CPU (número de núcleos e velocidade do *clock*), tamanho da RAM, tamanho do disco e desempenho de rede. Cada um desses atributos é padronizado usando uma nomenclatura única para todos os provedores de nuvem considerados. Essa nomenclatura foi estabelecida de maneira *ad hoc* para os provedores considerados nesta tese mas, como trabalho futuro, propomos o uso de uma ontologia para descrição dos atributos que definem tipos de recurso, como em [54, 190, 219].

O modelo concreto de recursos também inclui informações sobre recursos instanciados em um ambiente privado. Neste caso, os tipos de recurso são criados de acordo com a capacidade disponível nos recursos físicos, sendo o custo estabelecido neste caso como uma proporção do gasto em energia para seu funcionamento.

Ao analisarmos o segundo nível da representação proposta, mesmo considerando apenas atributos de *hardware* para descrever os tipos, a estimativa de recursos pode se tornar complexa devido à possível grande quantidade de tipos concretos disponíveis. Além disso, esses tipos possuem intersecção entre eles. Como exemplo, analisamos a intersecção entre os tipos disponíveis nos provedores AWS, Microsoft Azure, GCE e Rackspace. Ao considerar apenas o segundo nível da representação, há no total 133 tipos disponibilizados nesses provedores, dos quais 3,01% têm exatamente os mesmos atributos de *hardware*. Apesar da quantidade aparentemente insignificante de tipos idênticos, a intersecção na prática é bem maior, pois, ao implantar um serviço, deve-se também considerar os tipos que não são idênticos, embora sejam aceitáveis,

ou seja, que possuem atributos de *hardware* com maior capacidade. Embora esses tipos possam significar desperdício de recurso, pois consideram capacidade adicional na alocação, eles farão parte da solução nos casos em que tipos com exatamente a capacidade de *hardware* necessária não podem ser usados em virtude das demais restrições não-funcionais estabelecidas sobre o serviço. A Tabela 5.1 apresenta o percentual de tipos nesses provedores de acordo com a quantidade de tipos aceitáveis. Como exemplo, para 93,98% dos tipos considerados há ao menos um outro tipo aceitável, ao passo que para 69,17% há 10 ou mais tipos aceitáveis, e assim suscetivamente.

| **Tipos aceitáveis** | **% de tipos** |
|---|---|
| ≥ 1 | 93,98 |
| ≥ 10 | 69,17 |
| ≥ 20 | 50,38 |
| ≥ 30 | 41,35 |
| ≥ 40 | 31,58 |
| ≥ 50 | 20,30 |
| ≥ 60 | 15,04 |
| ≥ 70 | 8,27 |
| ≥ 80 | 3,76 |
| ≥ 90 | 2,26 |

**Tabela 5.1:** *Relação entre a quantidade de tipos aceitáveis e o percentual de tipos concretos para os quais esta quantidade existe nos provedores considerados.*

Como pode ser visto na tabela, a relação entre a quantidade de tipos aceitáveis e o percentual de tipos concretos que apresentam essa quantidade é significativa. Como exemplo, dos 133 tipos concretos considerados, para 69,17% desses tipos é possível obter 10 outros tipos que possuem a mesma capacidade de *hardware* ou mais. Em virtude dessas intersecções, não é preciso considerar todos os tipos concretos para realizar a estimativa de recursos. Por esse motivo, propomos a inclusão de um nível auxiliar na representação do modelo de recursos, que consiste em uma representação sintetizada do segundo nível dos tipos concretos. Denominamos esse nível auxiliar de ***modelo canônico de recursos***. As instâncias que compõem esse modelo são denominadas ***tipos canônicos*** de recurso. A quantificação de atributos de *hardware* nessas instâncias é especificada de acordo com os tipos concretos considerados em um procedimento que será descrito na próxima seção.

A Figura 5.1 ilustra os níveis de representação propostos na modelagem de recursos, e como eles se relacionam. A nomenclatura utilizada para referir a tipos concretos no primeiro nível da representação consiste no identificador usado pelos provedores que os disponibilizam, como por exemplo *t2.nano* e *Standard_A0*, seguido pelo símbolo @ e pelo identificador da região, como por exemplo *t2.nano@us-west-*

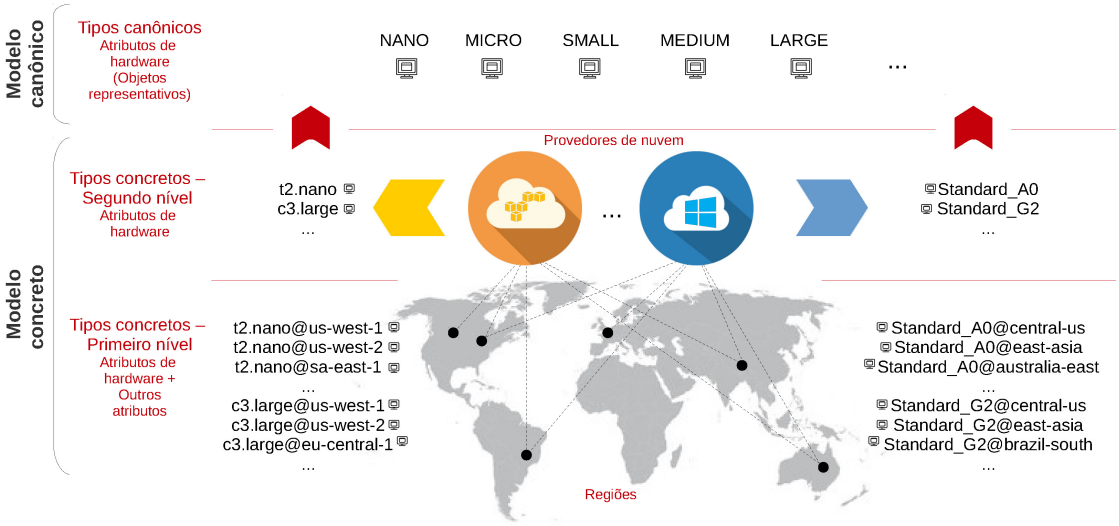



**Figura 5.1:** *Modelo de recursos.*

*1* e *Standard_A0@central-us*, *etc.* De maneira adicional, para referir ao mesmo tipo concreto, mas no segundo nível da representação, usamos apenas o identificador do tipo. Quanto aos tipos canônicos, usamos identificadores genéricos relacionados à capacidade de *hardware* provida pelo tipo, como por exemplo *SMALL*, *LARGE*, *etc.*

## 5.2 Geração de tipos canônicos de recurso

Ao definir a estratégia para gerar o modelo canônico de recursos, inicialmente investigamos o uso de clusterização [34]. Propusemos usar o $k$-means [123], um algoritmo de particionamento computacionalmente eficiente para agrupar conjuntos de dados $n$-dimensionais em $k$ clusters através da redução da variância dentro da classe; e o $x$-means [210], a versão estendida de $k$-means, usada para estimar o número de grupos. Modelamos cada atributo de *hardware* como uma dimensão, sendo variáveis linguísticas mapeadas para intervalos numéricos. Considerando como parâmetro as dimensões modeladas, usamos o *software* de aprendizagem de máquina Weka [122] para gerar os centroides dos *clusters*. Dessa forma, cada centroide seria definido como um tipo canônico. No entanto, esta estratégia apresentou alguns problemas.

O primeiro problema encontrado foi decidir qual medida estatística a ser usada para gerar o centroide de cada *cluster*, uma vez que qualquer categoria de média descartaria tipos cuja capacidade de *hardware* estivesse abaixo do centroide resultante. Este descarte deve ser feito porque a estimativa de recursos seria realizada com base nos valores dos centroides, e qualquer capacidade inferior seria então considerada insuficiente. Uma forma de lidar com esse problema, seria usar o valor mínimo em cada *cluster* para gerar o centroide, mas os *clusters* gerados em nossas avaliações

apresentaram intersecção entre eles, ou seja, o valor mínimo em cada *cluster* gerado era menor que o valor máximo do *cluster* imediatamente inferior. Por causa disso, os valores mínimos representariam instâncias de múltiplos *clusters*.

A estratégia adotada como alternativa para a geração do modelo canônico de recursos inicia com a normalização dos atributos de *hardware* dos tipos concretos para o intervalo $[1-10]$, usando os valores mínimo e máximo em cada atributo. Em seguida, é realizada a junção dos valores normalizados para cada tipo, resultando em um vetor que quantifica os atributos de *hardware* do tipo concreto. Os vetores são então ordenados, sendo os tipos canônicos gerados a partir deles. Apesar dos problemas descritos anteriormente, usamos o algoritmo X-means como uma heurística para determinar o número de tipos canônicos a serem gerados. Cada tipo canônico é gerado usando percentis e tipos extremos na série ordenada, da seguinte forma:

- Cluster `NANO`: tipo concreto menos robusto (tipo com menor capacidade de *hardware* entre todos os tipos considerados).
- Cluster `MICRO`: $20^{o}$ percentil.
- Cluster `SMALL`: $40^{o}$ percentil.
- Cluster `MEDIUM`: $60^{o}$ percentil.
- Cluster `LARGE`: $80^{o}$ percentil.
- Cluster `XLARGE`: tipo concreto mais robusto (tipo com maior capacidade de *hardware* entre todos os tipos considerados).

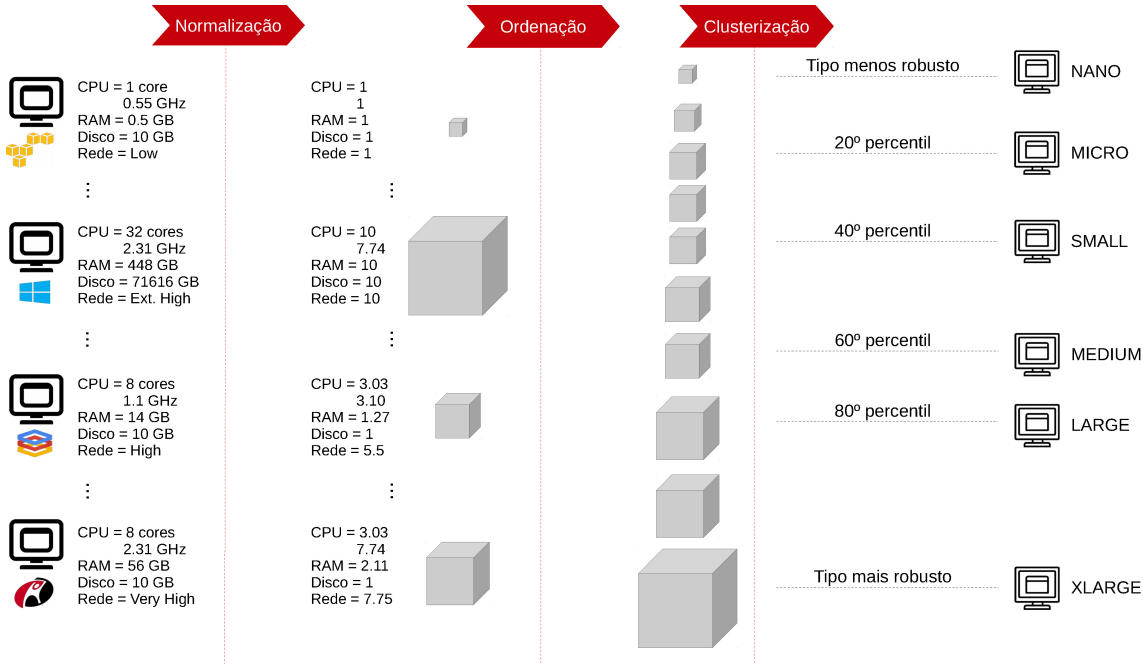



**Figura 5.2:** *Procedimento utilizado na geração do modelo canônico de recursos.*

A Figura 5.2 ilustra o procedimento utilizado na geração do modelo canônico de recursos. As hastes indicam cada uma das tarefas realizadas: normalização dos valores dos atributos, ordenação dos tipos concretos considerando o valor total dos atributos normalizados e clusterização usando as propriedades descritas acima. Cada cubo simboliza o valor total da normalização da capacidade de *hardware*, que é usada na geração dos tipos canônicos. Para ser considerado como pertencente ao conjunto representado por um determinado tipo canônico, um tipo concreto deve ter a soma de seus atributos de *hardware* normalizados, sendo maior ou igual à soma dos atributos estabelecidos para este tipo canônico. Além disso, essa soma deve ser menor que a soma do tipo canônico subsequente.

Usando esse procedimento, os problemas descritos anteriormente são solucionados. Outra vantagem é que a quantidade de tipos em cada conjunto se torna balanceada. A única exceção é para o conjunto *XLARGE*, que contém apenas tipos concretos cuja capacidade de *hardware* é a especificada pelo tipo concreto mais robusto.

## 5.3 Exemplo considerando provedores relevantes

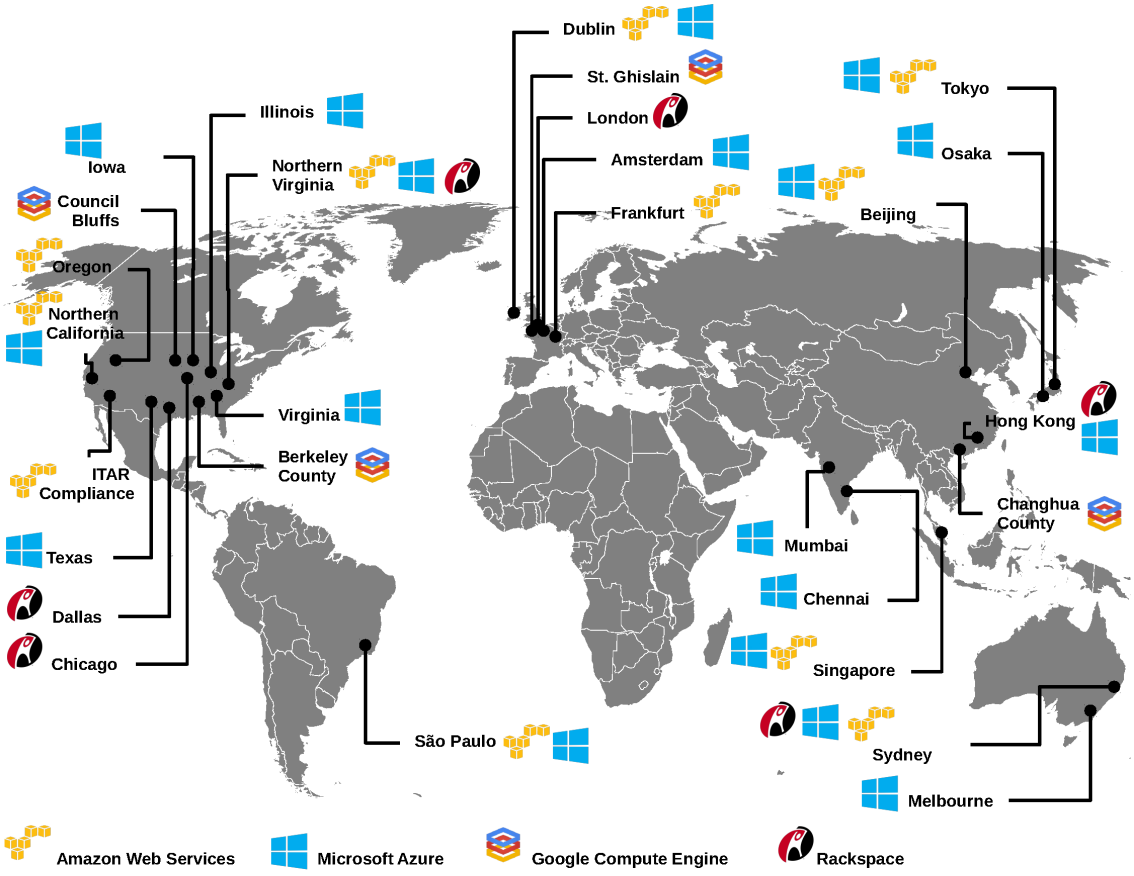



**Figura 5.3:** *Regiões utilizadas na exemplo de geração do modelo canônico de recursos.*

Para ilustrar o procedimento descrito na seção anterior, geramos o modelo canônico de recursos usando os tipos concretos disponibilizados pelos provedores AWS, Microsoft Azure, GCE e Rackspace em maio de 2016. A Figura 5.3 descreve a localização das regiões disponíveis para cada provedor considerado. Para reduzir a complexidade, combinamos zonas de disponibilidade em regiões únicas.

A Tabela 5.3 resume a disponibilidade de recursos para cada provedor de nuvem considerado. A segunda e a terceira coluna mostram a quantidade de tipos concretos disponíveis em cada provedor considerando o primeiro e o segundo nível da representação, respectivamente. As colunas seguintes apresentam o valor mínimo e máximo para cada atributo de *hardware* para tipos disponibilizados por este provedor. Alguns atributos, como desempenho de rede, não foram especificados por todos os provedores considerados.

A Tabela 5.2 apresenta os tipos canônicos gerados através do procedimento descrito ao considerar os tipos concretos disponibilizados pelos provedores considerados. A primeira coluna é o nome usado para identificar o tipo canônico. As colunas seguintes são os valores usados para quantificar cada atributo de *hardware*. Abaixo de cada valor (entre parênteses) estão os valores mínimo e máximo para esse atributo ao considerar os tipos concretos representados por este tipo canônico. Como pode ser visto, cada atributo de *hardware* do tipo canônico é definido como o valor mínimo do atributo equivalente nos tipos concretos representados. O uso de valores extremos nesta estratégia pode ser considerado porque os conjuntos gerados não têm intersecção entre eles.

| **Tipo canônico** | **# CPU *cores*** | **CPU *clock speed* (GHz)** | **RAM (GB)** | **Disco (GB)** | **Desempenho de rede** |
|---|---|---|---|---|---|
| NANO | 1<br>(1 – 8) | 0,55<br>(0,55 – 2,4) | 0,5<br>(0,5 – 30) | 10<br>(10 – 10.240) | Low<br>(Low – Moderate) |
| MICRO | 2<br>(2 – 16) | 1,1<br>(1,1 – 2,4) | 3,5<br>(3,5 – 60) | 10<br>(10 – 10.240) | Moderate<br>(Moderate – High) |
| SMALL | 4<br>(4 – 16) | 1,1<br>(1,1 – 2,4) | 7<br>(7 – 104) | 10<br>(10 – 10.240) | High<br>(High – High) |
| MEDIUM | 8<br>(8 – 32) | 1,1<br>(1,1 – 2,4) | 14<br>(14 – 208) | 10<br>(10 – 16.973) | High<br>(High – Very High) |
| LARGE | 8<br>(8 – 40) | 1,76<br>(1,76 – 2,9) | 56<br>(56 – 244) | 10<br>(10 – 64.000) | Very High<br>(Very High – Extremely High) |
| XLARGE | 32<br>(32 – 32) | 2,31<br>(2,31 – 2,31) | 448<br>(448 – 448) | 71.616<br>(71.616 – 71.616) | Extremely High<br>(Extremely High – Extremely High) |

**Tabela 5.2:** *Atributos que caracterizam os tipos canônicos gerados no exemplo.*

| **Provedor de nuvem** | **# tipos concretos (1º nivel)** | **# tipos concretos (2º nivel)** | **# CPU *cores*** | **CPU *clock speed* (GHz)** | **RAM (GB)** | **Disco (GB)** | **Desempenho de rede** |
|---|---|---|---|---|---|---|---|
| Amazon Web Services | 352 | 42 | 1 – 40 | 2,3 – 3,3 | 0,5 – 244 | 10 – 48.000 | Low – Extremely high |
| Microsoft Azure | 420 | 61 | 1 – 32 | 0,55* – 2,53* | 0,75 – 448 | 1.043 – 71.616 | Low – Extremely high |
| Google Compute Engine | 36 | 21 | 1 – 32 | 2,4 – 2,4 | 0.6 – 208 | 3.072 – 10.240 | - |
| Rackspace | 19 | 9 | 1 – 32 | - | 1 – 240 | 20 – 1.268 | - |
| **Total** | 827 | 133 | 1 – 40 | 0.55 – 3,3 | 0,5 – 448 | 10 – 71.616 | Low – Extremely high |

*Microsoft Azure usa a medida *Azure Compute Unit* (ACU) para especificar a velocidade de *clock*. ACU é atualmente padronizada em uma VM do tipo *Small* (Standard_A1) sendo 100. Para mapear ACU para a velocidade de *clock* usamos a mesma estratégia de mapeamento usada pela Amazon para especificar o desempenho através de sua medida *Elastic Compute Unit* (ECU): uma ECU é a potência equivalente de uma CPU com processador 1.0-1.2 GHz 2007 Opteron ou Xeon [201].

**Tabela 5.3:** *Atributos dos tipos concretos disponíveis nos provedores de nuvem pública considerados.*

## 5.4 Considerações finais

Neste capítulo descrevemos a representação utilizada na modelagem de recursos. Propomos um modelo que representa os atributos mais comumente encontrados nos tipos de VM disponibilizados em provedores públicos e criamos diferentes níveis de abstração desses tipos. Como forma de lidar com a variabilidade de tipos, levando em consideração a intersecção entre eles, além de reduzir o tempo necessário para a estimativa de recursos, criamos uma técnica para gerar tipos de recurso representativos, que nomeamos como tipos canônicos.

Apesar do modelo proposto para representação de recursos não considerar todos os atributos que caracterizam uma VM, argumentamos que ele é suficiente para guiar a implantação de composições. Os tipos de recurso propostos nesse capítulo são usados como entrada em nossa abordagem de síntese de recursos, que será descrita no próximo capítulo.

CAPÍTULO 6

# Síntese de recursos

A principal contribuição desta tese é propor uma abordagem de estimativa e seleção de recursos visando à satisfação de restrições não-funcionais na implantação de múltiplas coreografias de serviços. Definimos a realização dessas atividades como ***síntese de recursos*** e descrevemos neste capítulo como elas são realizadas com base na representação conjunta das coreografias e na modelagem de recursos descritos nos capítulos anteriores. Como forma de elucidar os principais elementos deste problema e definir de maneira precisa seu escopo, inicialmente apresentamos sua formalização, tendo como base os conceitos formalizados no Capítulo 4.

Em nossa abordagem, assumimos que os recursos são instanciados sob demanda, ou seja, não há um conjunto de instâncias de VM pré-criadas. Por esta razão, discutimos apenas as atividades relacionadas à estimativa, à descoberta, à seleção e à alocação de recursos, estando estratégias para escalonamento de recursos fora do nosso escopo.

A estimativa de recursos é realizada com base nas restrições de QoS, pois elas têm influência direta sobre a capacidade de *hardware* estabelecida para a implantação de cada serviço. De maneira especial, as restrições fim-a-fim dessa categoria permitem variar o balanceamento da contribuição de cada serviço relacionado a essas restrições e, assim, obter ganhos maiores na alocação de recursos. A qualidade geral na estimativa de recursos é medida como uma função global, calculada a partir do valor de utilidade das métricas de QoS consideradas e do ganho ao selecionar um determinado recurso.

Desta forma, o objetivo da nossa abordagem de estimativa de recursos é indicar uma capacidade apropriada para cada serviço coreografado que satisfaça as restrições impostas, ao mesmo tempo em que maximiza o valor desta função. Por outro lado, restrições em relação a atributos do recurso devem ser avaliadas na seleção de recursos, uma vez que elas podem eliminar tipos de recurso candidatos e classificar os tipos de recurso, de forma a definir quais têm maior preferência na decisão sendo tomada.

Este capítulo tem como objetivo apenas apresentar as técnicas propostas para

realizar a síntese de recursos. Detalhes de como essas técnicas foram implementadas serão discutidos no próximo capítulo, onde apresentaremos a arquitetura geral da abordagem.

## 6.1 O problema de síntese de recursos

O problema de síntese de recursos sujeita a restrições para múltiplas coreografias de serviços (*Constraint-aware resource Synthesis for multiple service Choreography Problem* – CSDP) consiste em, dado um conjunto de coreografias de serviços, com possível compartilhamento de serviços entre elas, estimar e selecionar os recursos necessários para implantar os serviços pertencentes às coreografias, de forma a satisfazer um conjunto de restrições não-funcionais especificadas como parte da entrada.

O conjunto de restrições a serem satisfeitas é definido como $\kappa$, o qual possui $\rho$ restrições $\{k_1, k_2, \ldots, k_\rho\}$. Definimos $\rho^q$, $\rho^e$ e $\rho^r$ como o número de restrições de QoS, restrições eliminatórias e restrições classificatórias, respectivamente, onde $\rho^q + \rho^e + \rho^r = \rho$. O resultado do CSDP é representado como $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$, onde $f_i$, $1 \leq i \leq n$, é um par $(s_i, v_j), s_i \in \mathcal{S}, v_j \in \mathcal{V}$ que representa o mapeamento de um serviço para um recurso que será usado para sua implantação. Consequentemente, $f_i.s$ e $f_i.v$ representam o serviço e o recurso no mapeamento, respectivamente.

Para comparar duas alternativas de mapeamento, definimos $\varphi_{ij}$, uma função que representa o ganho oferecido ao usuário ao alocar o recurso $j$ para implantar o serviço $i$. Assim sendo, definimos $\varphi(\mathcal{F})$ como o ganho composto de todos os recursos em $\mathcal{F}$. Dessa forma, a solução ótima para o problema é aquela que satisfaz todas as restrições e apresenta o melhor valor de $\varphi(\mathcal{F})$. Esta definição permite que diferentes critérios possam ser utilizados para representar o ganho. Em nossa abordagem consideramos que o custo de utilização dos recursos é usado como critério para determinar o ganho na seleção.

Como ilustrado na Figura 6.1, CSDP consiste em decidir o melhor emparelhamento de acordo com as restrições não-funcionais especificadas. Essa decisão é tomada avaliando a satisfação das restrições, tendo como base os serviços que compõem as coreografias e o padrão de conexão entre eles. Apesar de supormos que essas informações são representadas no grafo de dependências, gerado através do procedimento descrito no Capítulo 4, na resolução do problema outra notação poderia ser utilizada como entrada.

De maneira mais detalhada, dado o conjunto de serviços $\{s_1, s_2, \ldots, s_n\}$ pertencentes às coreografias especificadas, assim como as interligações entre eles; e o conjunto de recursos $\{v_1, v_2, \ldots, v_t\}$ disponíveis para implantar esses serviços, deve-se encontrar um mapeamento de cada serviço para um recurso de forma que todas as res-

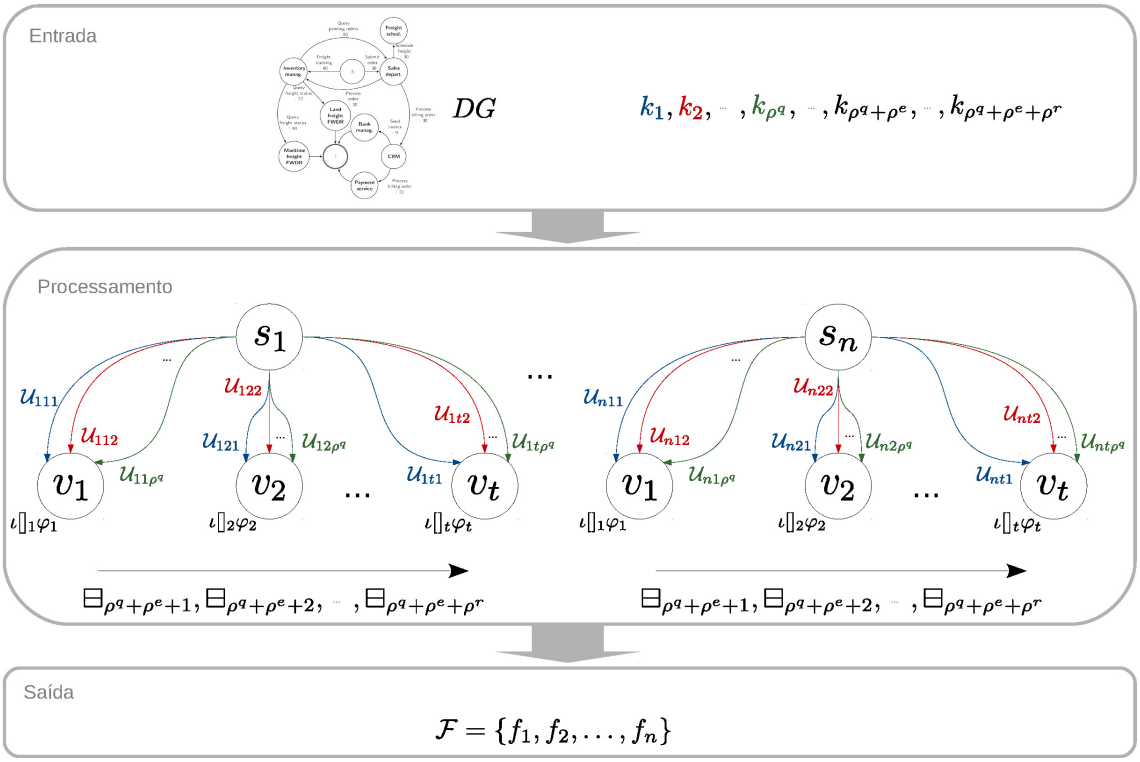



**Figura 6.1:** *Elementos do problema CSDP.*

trições não-funcionais sejam satisfeitas. Por simplicidade não representamos na figura a interligação entre os serviços.

O mapeamento de um serviço para um determinado recurso gera um valor para cada restrição de QoS. Quando o serviço $i, 1 \leq i \leq n$ é mapeado para o recurso $j, 1 \leq j \leq t$, o valor para a restrição de QoS $k, 1 \leq k \leq \rho^q$ é representado como $\mathcal{U}_{ijk}$, onde $\mathcal{U}$ é a função de utilidade especificada para essa restrição. A agregação dos valores de QoS, considerando todos os serviços incluídos na restrição em questão, deve estar dentro do limite definido pelo valor-alvo $\tau_k, 1 \leq k \leq \rho^q$ para que ela seja satisfeita. Esse mapeamento também considera o ganho $\varphi_j, 1 \leq j \leq t$ oferecido ao selecionar cada recurso.

Restrições de QoS, assim como restrições em relação a atributos do recurso, são especificadas para coreografias isoladas. Assim, um determinado serviço pode ser incluído em diversas restrições, das quais algumas podem ser baseadas na mesma métrica de QoS ou estar relacionadas ao mesmo atributo do recurso, nos casos em que o serviço é compartilhado entre múltiplas coreografias. Para lidar com essa duplicidade, propomos na Seção 4.2 um procedimento para a geração de uma representação conjunta das coreografias, que analisa a possibilidade de junção de restrições. Dessa forma, nos casos em que não é possível realizar a junção, cada restrição será analisada separadamente, mesmo que estas restrições sejam baseadas na mesma métrica de QoS ou no mesmo atributo do recurso. Como exemplo, na notação apresentada na

Figura 6.1, $\mathcal{U}_{121}$ e $\mathcal{U}_{122}$ representam os valores estimados para as restrições de QoS 1 e 2 quando o serviço 1 é mapeado para o recurso 2. As restrições 1 e 2, por sua vez, podem ser baseadas na mesma métrica de QoS, significando que não foi possível realizar a junção dessas restrições, ou cada uma ser baseada em uma métrica distinta.

Além das restrições de QoS, o mapeamento deve ser realizado considerando as restrições eliminatórias estabelecidas em função dos atributos $\iota[]_j, 1 \leq j \leq t$ que descrevem cada recurso. A avaliação dessas restrições é realizada considerando os valores-alvo $\tau_k, \rho^q + 1 \leq k \leq \rho^q + \rho^e$ definidos para elas.

Adicionalmente, os recursos selecionados devem ser ordenados considerando as restrições classificatórias especificadas. A ordem será obtida de acordo com os operadores de classificação $\boxminus_k, \rho^q + \rho^e + 1 \leq k \leq \rho^q + \rho^e + \rho^r$ definidos para essas restrições.

CSDP pode ser mais formalmente expresso como:

$$
\begin{aligned}
&\text{Maximizar} \quad \varphi(\mathcal{F}) = \sum_{i=1}^{n} \sum_{j=1}^{t} \varphi_{ij} y_{ij}; \\
&\text{sujeito a} \quad \sum_{k=1}^{\rho} x_k^{\bullet} = \rho; \\
&\qquad\qquad \sum_{j=1}^{t} y_{ij} = 1, \qquad\qquad i \in \{1, \ldots, n\}; \\
&\qquad\qquad y_{ij} \in \{0, 1\}, \qquad\qquad i \in \{1, \ldots, n\}, \\
&\qquad\qquad\qquad\qquad\qquad\quad\; j \in \{1, \ldots, t\}.
\end{aligned}
$$

A variável $y_{ij}$ é igual a 0, implicando que o recurso $j$ não é selecionado para implantar o serviço $i$, ou igual a 1 implicando que o recurso $j$ é selecionado para o serviço $i$. A notação $x_k^{\bullet}$ é usada como uma generalização das variáveis definidas nas equações 4-1, 4-2 e 4-3, apresentadas no Capítulo 4.

CSDP é claramente um problema de otimização e argumentamos que não é possível encontrar em tempo polinomial uma solução ótima para uma instância deste problema. Para confirmar isso, demonstramos que a versão de decisão do CSDP pertence à classe NP-Completo de problemas.

### 6.1.1 NP-Completude de CSDP

**Teorema 6.1** *CSDP pertence à classe* NP*-Completo.*

*Prova.* Mesmo CSDP sendo um problema de otimização, por simplicidade, apresentamos a demonstração tratando este como um problema de decisão, isto é, há um limite $b$ para $\varphi(\mathcal{F})$. Esta equivalência é válida uma vez que, ao fazer uma busca binária ao

limite $b$, pode-se transformar uma solução de tempo polinomial para a versão de decisão em um algoritmo de tempo polinomial para o problema de otimização correspondente. A seguir é apresentada a definição do problema de decisão considerado, usando o formato proposto por Garey e Johnson [99]:

*INSTÂNCIA:* Um conjunto $\mathcal{S}$ de serviços; um conjunto $\mathcal{V}$ de recursos; para cada $v_j \in \mathcal{V}$, $1 \leq j \leq t$, um ganho $\varphi_{ij} \in \mathbb{R}^+$; um conjunto $\kappa$ de $\rho$ restrições não-funcionais; um valor $b \in \mathbb{R}^+$.

*PERGUNTA:* Há uma seleção de exatamente um recurso para cada serviço $s_i \in \mathcal{S}$, $1 \leq i \leq n$ tal que $\sum_{k=1}^{\rho} x_k^{\bullet} = \rho$ e $\sum_{i=1}^{n}\sum_{j=1}^{t} \varphi_{ij} y_{ij} \geq b$?

Em primeiro lugar, demonstramos que CSDP $\in$ NP. É evidente que, dado um certificado representado por um conjunto de $n$ pares $(s_i, v_j), s_i \in \mathcal{S}, 1 \leq i \leq n, v_j \in \mathcal{V}, 1 \leq j \leq t$ e um limite $b$, é simples apresentar um algoritmo que verifica se $\sum_{k=1}^{\rho} x_k^{\bullet} = \rho$ e $\sum_{i=1}^{n}\sum_{j=1}^{t} \varphi_{ij} y_{ij} \geq b$ em tempo $O(n\rho)$. Esse algoritmo deverá simplesmente avaliar todas as restrições, considerando cada um dos pares especificados.

A seguir, demonstramos que CSDP pertence à classe NP-Difícil através de uma redução do problema da mochila multidimensional de múltipla escolha (*Multiple-choice Multi-dimension Knapsack Problem* – MMKP) [187].

MMKP é uma das variantes do Problema da Mochila 0-1 [234]. Ele é um problema de otimização que pode ser definido como: dado um conjunto $H$ de itens divididos em $h$ categorias $Q_q$, onde cada categoria $Q_q$, $q = 1,\ldots,h$, possui $\kappa_q = |Q_q|$ itens tal que $\forall\ 1 \leq i, j \leq h$ e $i \neq j, Q_i \cap Q_j = \emptyset$ e $\cup_{i=1}^{h} Q_i = H$. Cada item $o, o = 1,\ldots,\kappa_q$, da categoria $Q_q$ oferece um valor não-negativo como lucro $\varrho_{qo}$, e possui dimensões dadas por um vetor de peso $W_{qo} = \{w_{qo}^1, w_{qo}^2, \ldots, w_{qo}^{\ell}\}$, onde cada elemento $w_{qo}^a, 1 \geq a \geq \ell$ também é um valor não negativo. As dimensões da mochila são dadas pelo vetor $\mathcal{A} = \{\mathcal{A}^1, \mathcal{A}^2, \ldots, \mathcal{A}^{\ell}\}$. O objetivo do MMKP é selecionar exatamente um item de cada categoria de forma a maximizar o lucro total, sujeito às dimensões da mochila.

Formalmente, o MMKP pode ser definido como:

$$
\begin{aligned}
\text{Maximizar} \quad & Z(Y) = \sum_{q=1}^{h}\sum_{o=1}^{\kappa_q} \varrho_{qo} y_{qo}; & \\
\text{sujeito a} \quad & \sum_{q=1}^{h}\sum_{o=1}^{\kappa_q} w_{qo}^a y_{qo} \leq \mathcal{A}^a, & a \in \{1,\ldots,\ell\}; \\
& \sum_{o=1}^{\kappa_q} y_{qo} = 1, & q \in \{1,\ldots,h\}; \\
& y_{qo} \in \{0,1\}, & q \in \{1,\ldots,h\}, \\
& & o \in \{1,\ldots,\kappa_q\}.
\end{aligned}
$$

A variável $y_{qo}$ é igual a 0, implicando que o item $o$ da categoria $Q_q$ não é selecionado, ou igual a 1 implicando que o item $o$ da categoria $Q_q$ é selecionado.

A versão de decisão para o problema MMKP é definida a seguir.

*INSTÂNCIA:* Um conjunto $H$ de itens divididos em $h$ categorias $Q_q$, onde cada categoria $Q_q$, $q = 1,\ldots,h$, possui $\kappa_q = |Q_q|$ itens tal que $\forall\ 1 \leq i,j \leq h$ e $i \neq j, Q_i \cap Q_j = \emptyset$ e $\cup_{i=1}^{h} Q_i = H$; para cada item $o \in Q_q$ um lucro $\varrho_{qo} \in \mathbb{R}^+$, e dimensões dadas por um vetor $W_{qo} = \{w_{qo}^1, w_{qo}^2, \ldots, w_{qo}^\ell\}$, onde cada elemento $w_{qo}^a \in \mathbb{R}^+, 1 \leq a \leq \ell$; as dimensões da mochila dadas pelo vetor $\mathcal{A} = \{\mathcal{A}^1, \mathcal{A}^2, \ldots, \mathcal{A}^\ell\}$, onde cada elemento $\mathcal{A}^a \in \mathbb{R}^+, 1 \leq a \leq \ell$; um valor $b \in \mathbb{R}^+$.

*PERGUNTA:* Há uma seleção de exatamente um elemento de cada categoria $Q_q$, $1 \leq q \leq h$ tal que $\sum_{q=1}^{h} \sum_{o=1}^{\kappa_q} w_{qo}^a y_{qo} \leq \mathcal{A}^a, 1 \leq a \leq \ell$ e $\sum_{q=1}^{h} \sum_{o=1}^{\kappa_q} \varrho_{qo} y_{qo} \geq b$?

Dado que MMKP pertence à classe NP-Completo [187], iremos demonstrar que MMKP $\preceq_{\mathsf{P}}$ CSDP.

Dada uma instância do MMKP, ou seja, $H$, $h$, $\{Q_1,\ldots,Q_q\}$, $\{\kappa_1,\ldots,\kappa_q\}$, $\{\varrho_{11},\ldots,\varrho_{1\kappa_1},\ldots,\varrho_{q1},\ldots,\varrho_{q\kappa_q}\}$, $\{W_{11},\ldots,W_{1\kappa_1},\ldots,W_{q1},\ldots,W_{q\kappa_q}\}$, $\mathcal{A}$, $b$, iremos construir uma instância do CSDP. Essa redução é ilustrada na Figura 6.2. Seja $n = h$. Embora no CSDP todos os elementos no conjunto de recursos são analisados para cada serviço, iremos assumir, sem perda de generalidade, que para cada serviço há um conjunto diferente de recursos disponíveis para sua implantação. Essa suposição apenas restringe ainda mais o problema, mas não altera sua definição. Ao fazer isso, é possível mapear $H$ para $\mathcal{V}$ e supor cada $Q_q$ como sendo o conjunto de recursos para um determinado serviço. Seja $W_{qo}$ a função de utilidade empregada na avaliação de uma restrição de QoS, de forma que $(w_{qo}^1, w_{qo}^2, \ldots, w_{qo}^\ell)$ é mapeado para $(\mathcal{U}_{ij1}, \mathcal{U}_{ij2}, \mathcal{U}_{ij\rho})$ e $\varrho_{qo}$ é mapeado para $\varphi_j$. Neste caso, assumimos que na instância criada para o CSDP, $\rho^e = 0$ e $\rho^r = 0$. Consequentemente, cada dimensão $\mathcal{A}^a$, $a = 1,\ldots,\ell$ é mapeada para o valor-alvo $\tau_k$ da restrição. Consideramos que o limite $b$ é o mesmo para as instâncias dos dois problemas. Claramente, essa construção pode ser feita em tempo polinomial. Em seguida, é necessário demonstrar que os itens selecionados na instância original do MMKP satisfazem todas as restrições e possuem lucro $\geq b$ se e somente se os recursos selecionados na instância criada do CSDP também satisfazem todas as restrições e oferecem um ganho $\geq b$.

Inicialmente, suponhamos que há uma seleção de um único recurso para cada serviço (considerando todas as coreografias) que satisfaça todas as restrições e que ofereça um ganho $\varphi(\mathcal{F}) = b$, ou seja, $\mathcal{U}_{11k} \oplus \ldots \oplus \mathcal{U}_{1tk} \oplus \mathcal{U}_{21k} \oplus \ldots \oplus \mathcal{U}_{2tk} \oplus \ldots \oplus \mathcal{U}_{ntk} \leq \tau_k, 1 \leq k \leq \rho$, onde $\mathcal{U}_{ijk}, 1 \leq i \leq n, 1 \leq j \leq t$ é o elemento neutro da métrica de QoS referente à restrição $k$ quando não existir par $s_i, v_j$; $\sum_{k=1}^{\rho} x_k^q = \rho$ e $\sum_{i=1}^{n} \sum_{j=1}^{t} \varphi_{ij} y_{ij} = b$. Pela nossa construção, isso significa que $\sum_{q=1}^{h} \sum_{o=1}^{\kappa_q} w_{qo}^a y_{qo} \leq \mathcal{A}^a, a \in \{1,\ldots,\ell\}$, ou

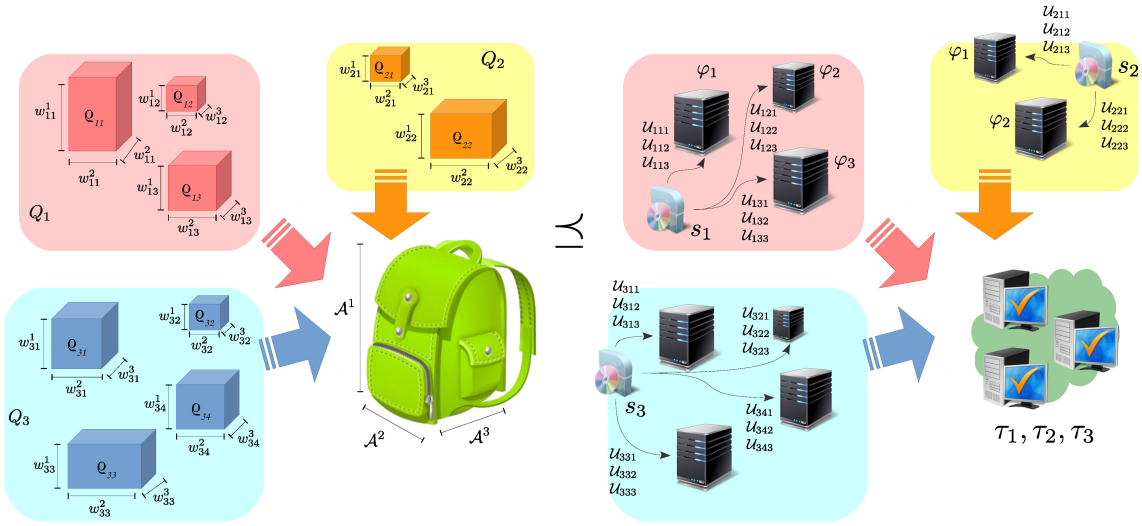


**Figura 6.2:** *Redução de uma instância do MMKP para uma instância do CSDP.*

seja, os itens equivalentes aos recursos selecionados não ultrapassarão os limites da mochila. Tendo a instância do CSDP solução, para a instância equivalente do MMKP, $\sum_{o=1}^{\kappa_q} y_{qo} = 1, q \in \{1,\ldots,h\}$, isto é, somente um item de cada categoria será selecionado. Como havíamos suposto que $\varphi(\mathcal{F}) = b$, para a instância do MMKP a relação $Z(Y) = b$ também será verdadeira.

Por outro lado, suponhamos que a instância do MMKP tem solução tal que $Z(Y) = b$. Dessa forma, $\sum_{q=1}^{h}\sum_{o=1}^{\kappa_q} w_{qo}^{a} y_{qo} \leq \mathcal{A}^{a}, 1 \leq a \leq \ell$. Similarmente, pela nossa construção, isso significa que $\mathcal{U}_{11k} \oplus \ldots \oplus \mathcal{U}_{1tk} \oplus \mathcal{U}_{21k} \oplus \ldots \oplus \mathcal{U}_{2tk} \oplus \ldots \oplus \mathcal{U}_{ntk} \leq \tau_k, 1 \leq k \leq \rho$, ou seja, todas as restrições são satisfeitas (considerando todas as coreografias): $\sum_{k=1}^{\rho} x_k^q = \rho$. Tendo a instância do MMKP solução, $\sum_{o=1}^{\kappa_q} y_{qo} = 1, 1 \leq q \leq h$, isto é, somente um item de cada categoria é selecionado, o que significa que $|\mathcal{F}| = n$, implicando que há um par com cada serviço que o mapeia a um recurso. Como havíamos suposto que $Z(Y) = b$, $\varphi(\mathcal{F}) = b$ e a instância do CSDP criada também possui solução. Inversamente, suponhamos que a instância do MMKP não tenha solução. Assim sendo, para satisfazer a condição $\sum_{o=1}^{\kappa_q} y_{qo} = 1, 1 \leq q \leq h$ tem-se que $\exists a \in \{1,\ldots,\ell\} | \sum_{q=1}^{h}\sum_{o=1}^{\kappa_q} w_{qo}^{a} y_{qo} > \mathcal{A}^{a}$. Pela nossa construção, a restrição correspondente $k, 1 \leq k \leq \rho$ sobre $a$ é também avaliada como $\mathcal{U}_{11k} \oplus \ldots \oplus \mathcal{U}_{1tk} \oplus \mathcal{U}_{21k} \oplus \ldots \oplus \mathcal{U}_{2tk} \oplus \ldots \oplus \mathcal{U}_{ntk} > \tau_k$, isto é, não é possível satisfazer essa restrição e, consequentemente, a instância do CSDP não tem solução.

Portanto, demonstramos que MMKP $\leq_{\mathsf{P}}$ CSDP. Assim, CSDP pertence à classe NP-difícil, e uma vez que demonstramos que CSDP pertence à classe NP, CSDP pertence à classe NP-completo. □

## 6.2 Estimativa de recursos

A estimativa de recursos está diretamente relacionada a propriedades não-funcionais esperadas para os serviços coreografados. Mais precisamente, essa atividade é guiada pelas restrições de QoS que devem ser satisfeitas, tornando a estimativa de QoS um dos aspectos mais importantes na definição da capacidade dos recursos.

Conforme discutido anteriormente, uma restrição de QoS pode ser especificada para um único serviço/operação ou pode restringir valores fim-a-fim na encenação de uma coreografia. No primeiro caso, basta analisar a função de utilidade especificada em cada restrição de QoS e verificar se sua estimativa está dentro do valor-alvo. Por outro lado, a avaliação de QoS fim-a-fim é influenciada pela topologia da coreografia e, neste caso, a estimativa de QoS é obtida de forma diferente para cada padrão de conexão.

Em nossa abordagem, a influência de cada padrão de conexão na QoS fim-a-fim é avaliada, de forma indireta, com base na representação conjunta das coreografias (grafo de dependências), que definimos no Capítulo 4. Usando esta estrutura é possível avaliar restrições fim-a-fim mesmo sem a representação explícita dos conectores das composições originais, uma vez que:

*i)* o padrão de sequência é explicitamente modelado através das arestas do grafo;

*ii)* quando há um conector de disjunção mutuamente exclusiva, a probabilidade de execução de cada opção no fluxo é refletida na carga quando o grafo de dependências é gerado. Dessa forma, quando a função de utilidade é calculada para serviços incluídos em fluxos com este padrão, a probabilidade é aplicada indiretamente; e

*iii)* quando há conectores de conjunção e disjunção, as informações sobre fluxos executados em paralelo são preservadas nas estruturas de árvore de paralelismo incluídas na representação dos grupos em restrição. Conforme será detalhado no próximo capítulo, consideramos a estimativa de QoS para cada um dos fluxos executados em paralelo e estabelecemos aquela mais restritiva como estimativa final. Ao adotar este procedimento, a estimativa de QoS obtida será a mesma que aquela resultante da avaliação do conector original.

A estimativa de recursos é realizada com base no modelo canônico de recursos. A utilização deste modelo em detrimento do uso do modelo concreto se justifica pelo fato do modelo canônico oferecer uma visão geral dos níveis de capacidade oferecidos nos tipos disponíveis, ao mesmo tempo que permite reduzir significativamente o espaço de busca. Conforme será discutido no Capítulo 8, demonstramos com alguns experimentos que usar essa estratégia permite obter a estimativa de recursos com precisão aceitável, sem a sobrecarga necessária para considerar todos os tipos disponíveis.

Um dos principais desafios para a estimativa de recursos no problema considerado é lidar com interferências sobre a satisfação de restrições de QoS devido ao compartilhamento de serviços. Para oferecer uma solução eficiente para a estimativa de recursos sem o custo associado a uma estratégia de busca exaustiva, utilizamos uma heurística para a solução do problema MMKP. Usando em ordem inversa a redução apresentada na Seção 6.1 para demonstrar a NP-completude de CSDP, construímos uma instância do problema MMKP a partir de uma instância do problema CSDP. Utilizamos a heurística WS-HEU [278], que resolve instâncias do MMKP e gera soluções que chegam a 96% do valor da solução ótima. Esta heurística possui complexidade de tempo no pior caso de $O(\rho^q n^2(\Psi-1)^2)$, em que $\rho^q$ = número de restrições de QoS, $n$ = número de serviços, e $\Psi$ = número de tipos canônicos.

WS-HEU é uma versão modificada da heurística HEU [150], baseada no conceito de consumo agregado ao escolher um item candidato em um grupo para resolver o MMKP. A heurística original encontra uma solução atualizando os itens selecionados de cada grupo. A heurística WS-HEU começa com um pré-processamento para encontrar uma solução viável inicial, isto é, uma combinação de itens que satisfaz todas as restrições, mas que não necessariamente é a melhor solução. Uma etapa de pós-processamento melhora o valor de utilidade total da solução, com uma atualização, seguida por um ou mais retrocessos.

Na versão original do problema MMKP e, consequentemente na heurística WS-HEU, as restrições são aplicadas a todos os grupos dos quais itens devem ser selecionados. Dessa forma, modificamos a heurística para que as restrições possam ser aplicadas apenas a um subconjunto dos grupos, que em nosso caso equivalem aos serviços. A necessidade dessa modificação se dá porque no nosso problema as restrições são especificadas para coreografias isoladas e, assim, não serão sempre aplicadas a todos os serviços. Apesar de não demonstrarmos formalmente, argumentamos que essa modificação não altera o problema original de forma a invalidar a heurística. Esse argumento é embasado no fato das restrições serem avaliadas de acordo com o valor agregado da contribuição dos serviços (grupos) a que elas se referem, sendo que informações sobre esses serviços não são usadas de forma direta na decisão. Com isso, considerar todos os serviços ou somente um subconjunto deles não interfere no funcionamento da heurística.

Em nossa implementação da estimativa de recursos, o procedimento começa pela avaliação das restrições de QoS especificadas para serviços isolados (restrições locais). Cada restrição com esse perfil é avaliada considerando os tipos canônicos de recurso disponíveis. Dessa forma, os tipos canônicos que não possuem capacidade suficiente para satisfazer alguma das restrições locais para um determinado serviço são descartados na subsequente avaliação de restrições fim-a-fim especificadas sobre

coreografias que contém este serviço. Caso todos os tipos canônicos sejam descartados nessa etapa para um determinado serviço, é adicionada uma nova instância deste serviço como forma de reduzir a demanda de recurso sobre instâncias individuais deste serviço. Em seguida, o procedimento é reiniciado. Esse processo é iterado até que haja ao menos um tipo canônico para cada serviço que possa satisfazer todas as restrições de QoS locais sobre ele.

A avaliação de restrições de QoS fim-a-fim inicia com o cálculo do peso de todos os pares serviço, recurso, usando a agregação da contribuição desse serviço em todas as restrições de QoS fim-a-fim especificadas sobre coreografias que o contêm. Para calcular essa agregação, os valores da contribuição em cada restrição de QoS são mapeados para um valor numérico que representa o peso. Para métricas cujo domínio já é um valor numérico (como latência) isso é direto. Contudo, para métricas cujo domínio é formado por valores nominais, deve ser definido um valor numérico equivalente a cada um dos valores nominais.

Para se obter a contribuição de um serviço em uma restrição é verificado se este serviço faz parte de alguma árvore de paralelismo relacionada à restrição. Nos casos em que o serviço não está inserido em uma árvore de paralelismo, a contribuição é estabelecida como o valor da função de utilidade estimado para a métrica. Quando o serviço está inserido em uma árvore de paralelismo, a contribuição deste serviço deve ser calculada levando em consideração a contribuição (nesta restrição) dos demais serviços inseridos na árvore. Neste caso, o procedimento para obter a contribuição consiste em verificar o ramo da árvore que oferece o pior valor de QoS. Caso o serviço esteja inserido neste ramo a contribuição é estabelecida como o valor da função de utilidade para este serviço. Caso contrário, sua contribuição é estabelecida como zero, uma vez que a contribuição que prevalece é aquela definida pelos serviços que formam o ramo com pior QoS, devendo então a contribuição de serviços fora deste ramo ser ignorada.

Após o cálculo da contribuição de QoS, os pares são ordenados pela razão entre o valor do recurso e o peso calculado. O valor do recurso, neste caso, refere-se ao ganho obtido ao selecionar este recurso. A estratégia imediata para estabelecer esse atributo seria considerar o inverso do custo monetário, de forma que recursos com menor custo devem ser privilegiados na seleção. Contudo, uma vez que não faz sentido incluir a representação do custo monetário como atributo nos tipos canônicos, pois eles representam uma generalização dos tipos concretos, o valor do recurso é definido como

$$\frac{1}{r} \tag{6-1}$$

em que $r$ é a agregação dos atributos de *hardware* normalizados. Como consequência,

os tipos canônicos com maior capacidade de *hardware* apresentam menor valor. Adotamos esta estratégia como uma heurística para influenciar a seleção de tipos concretos com menor custo, uma vez que nos provedores de nuvem pública o custo de um tipo concreto é diretamente proporcional à capacidade de *hardware*. Essa heurística não é ideal em todos os casos, visto que esse comportamento não é verificado para tipos alocados em nuvens privadas. Contudo, o estabelecimento de uma estratégia mais precisa para a definição do valor dos tipos canônicos é tido como trabalho futuro.

Tendo como entrada os pares ordenados, uma solução é construída escolhendo para cada serviço o par com maior razão. Após isso, as restrições fim-a-fim são verificadas e, se não forem satisfeitas, novas soluções são criadas considerando os pares subsequentes até que todas as restrições sejam satisfeitas. Quando, mesmo considerando todos os pares nesta etapa, nenhuma solução viável é encontrada, novas instâncias dos serviços devem ser incluídas. A estratégia implementada para realizar a adição de instâncias de serviços será discutida no próximo capítulo.

Após a inclusão de novas instâncias dos serviços, nos casos em que isso é necessário, o procedimento é reiniciado, refazendo a avaliação das restrições locais para os serviços que tiveram instâncias adicionadas. Em seguida, o peso das contribuições é novamente calculado e os pares de serviço e recurso novamente ordenados pela razão entre o valor do recurso e o peso calculado, sendo uma solução viável então buscada. Caso ainda assim nenhuma solução seja encontrada, todo o processo de estimativa é repetido novamente. Caso contrário, a heurística tenta melhorar a solução encontrada localmente para cada serviço usando a técnica *simulated annealing* [1].

*Simulated annealing* é uma meta-heurística para otimização que realiza busca local probabilística, e se fundamenta numa analogia com a termodinâmica. *Annealing* é o processo utilizado para fundir um metal, onde este é aquecido a uma temperatura elevada e em seguida é resfriado lentamente, de modo que o produto final seja uma massa homogênea. Neste contexto, o processo de otimização é realizado por níveis, simulando os níveis de temperatura no resfriamento. Em cada nível, dado um ponto no espaço de solução, vários pontos na vizinhança deste ponto são gerados e o correspondente valor da função sendo otimizada é calculado. Cada ponto gerado é aceito ou rejeitado de acordo com uma certa probabilidade. Esta probabilidade de aceitação decresce de acordo com o nível do processo, ou equivalentemente, de acordo com a temperatura [121].

A saída da heurística WS-HEU é um mapeamento de cada serviço na estrutura do grafo de dependências (e, consequentemente, suas prováveis instâncias adicionais) para o tipo canônico selecionado. Ao utilizar essa estratégia, a capacidade de *hardware* mais adequada é selecionada a partir dos tipos canônicos de recurso disponíveis, ao passo que a contribuição nas restrições fim-a-fim é balanceada entre os serviços

do conjunto de coreografias. A estimativa de recursos é, portanto, obtida como uma consequência do emparelhamento. Os tipos canônicos selecionados são usados na próxima etapa da síntese, que realiza a seleção de recursos.

## 6.3 Seleção de recursos

A segunda etapa da síntese de recursos consiste na seleção de recursos. Essa atividade é implementada em nossa abordagem através do mapeamento dos tipos canônicos selecionados na primeira etapa da síntese para tipos concretos de recurso. As restrições em relação a atributos do recurso são avaliadas nessa etapa, pois elas determinam os tipos concretos que são aceitáveis para cada serviço.

Na seleção de recursos, além dos tipos concretos inicialmente descobertos, consideramos a inclusão de novos tipos criados como uma aproximação da capacidade de *hardware* estabelecida em cada tipo canônico selecionado. Esses tipos são incluídos quando se considera o uso de uma nuvem privada com capacidade disponível ou quando a funcionalidade de criação de tipos de VM customizáveis é oferecida pelos provedores de nuvem pública considerados. A criação de tipos customizáveis é uma funcionalidade que passou a ser oferecida por alguns provedores (ex.: Google [114]) como uma alternativa nos cenários em que os tipos predefinidos não satisfazem de maneira eficiente as necessidades da aplicação. Contudo, a inclusão desses tipos em nossa abordagem não descarta a utilização de tipos concretos predefinidos, uma vez que:

*i)* atualmente é bem limitado o número de provedores de nuvem que permitem a criação de tipos de VM customizáveis. Além disso, não há total liberdade na configuração destes tipos, visto que alguns provedores impõem regras sobre sua criação, como por exemplo, limitar o número de núcleos de CPU a números pares ou o total de memória a múltiplos de uma determinada constante. Outro problema é que essas regras podem variar para cada região. Dessa forma, ao considerar apenas o uso de tipos de VM customizáveis seria difícil satisfazer todas as restrições não-funcionais requeridas. O uso dos tipos concretos preestabelecidos faz com que as opções na seleção de recursos sejam aumentadas significativamente, favorecendo o processo de implantação de serviços;

*ii)* os tipos concretos predefinidos são necessários para a geração dos tipos canônicos. Outra estratégia para a geração dos tipos canônicos poderia ser a geração de tipos com capacidade estabelecida de maneira empírica, ou seja, através da variação da capacidade considerando faixas de valores (como por exemplo, entre 1 GB e 100 GB para memória), ou usando uma estratégia equivalente. Contudo,

a alocação de tipos concretos considerando as capacidades estabelecidas seria um desafio, pelos motivos citados acima.

Após a inclusão dos novos tipos, a seleção de recursos é realizada pela filtragem e classificação dos tipos concretos representados por cada tipo canônico. Visando uma seleção de recursos ainda mais eficiente, investigamos a possibilidade de alocar um único recurso para implantar múltiplos serviços, ou seja, consolidar a seleção inicialmente realizada para um conjunto de serviços em um único recurso. Definimos essa etapa como ***consolidação de recursos***, que consiste na última etapa realizada na seleção de recursos. A seguir são discutidos maiores detalhes sobre cada etapa da seleção de recursos.

### 6.3.1 Filtragem e classificação de recursos

Em virtude da estratégia utilizada para geração dos tipos canônicos (descrita na Seção 5.2), pode-se garantir que todos os tipos concretos representados por um determinado tipo canônico têm, no mínimo, a mesma capacidade de *hardware* descrita por este tipo canônico. Dessa forma, essa etapa da seleção de recursos pode ser realizada com base apenas nas restrições em relação a atributos do recurso. Para tal, a filtragem e classificação de recursos pode ser aplicada para cada serviço separadamente, uma vez que as restrições em relação a atributos do recurso são especificadas para serviços isolados. Para permitir o processamento das restrições em relação a atributos do recurso, adicionamos na taxonomia originalmente apresentada na Figura 4.1 um nível de categorização dessas restrições, conforme pode ser visto na Figura 6.3.

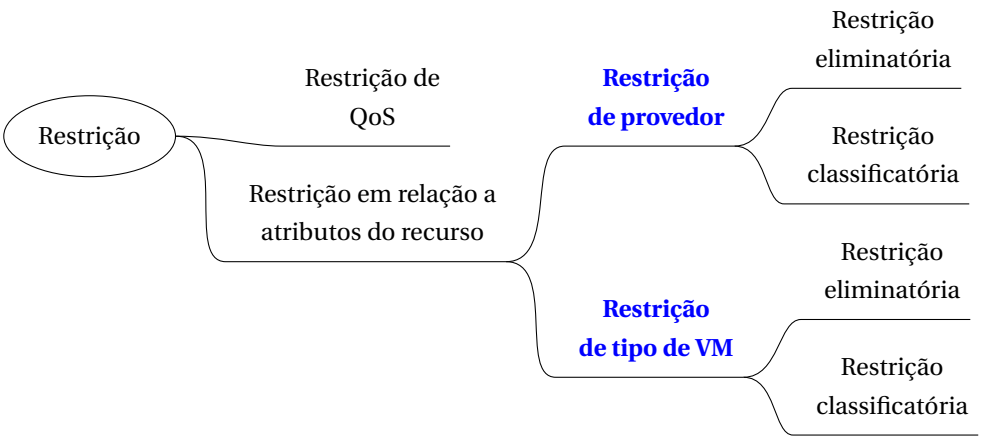



**Figura 6.3:** *Taxonomia de restrições atualizada.*

**Restrições de provedor** referem-se a atributos que caracterizam o provedor e, consequentemente, se aplicam a todos os tipos de VM disponibilizados por ele. Exemplos de restrições nessa categoria são: evitar provedores que requerem cobrança mí-

nima dos recursos utilizados (restrição eliminatória) e privilegiar a seleção de recursos de provedores com maior reputação (restrição classificatória).

Por outro lado, **restrições de tipos de VM** dizem respeito ao tipo concreto propriamente dito e não têm relação direta com atributos que caracterizam o provedor. Exemplos de restrições nessa categoria são: selecionar recursos que serão alocados em um determinado país (restrição eliminatória) e usar o custo de utilização como critério de escolha entre os tipos de VM disponíveis (restrição classificatória).

De acordo com a definição das restrições em relação a atributos do recurso, especificadas através da linguagem proposta na Seção 4.1.3, são gerados filtros e classificadores que são aplicados sobre os tipos concretos. A ordem de aplicação é definida de acordo com a taxonomia atualizada. Visando diminuir o tempo necessário para obter o resultado da seleção de recursos, restrições eliminatórias são avaliadas primeiro. Pelo mesmo motivo, restrições de provedor tem preferência sobre restrições de tipos de VM, uma vez que elas permitem uma redução mais imediata no espaço de busca. Com base nessas considerações, as restrições em relação a atributos do recurso são avaliadas na seguinte sequência:

1º) restrições eliminatórias de provedor;
2º) restrições eliminatórias de tipo de VM;
3º) restrições classificatórias de provedor; e
4º) restrições classificatórias de tipo de VM.

A ordem estabelecida dos tipos concretos de recurso depende da sequência com que as restrições classificatórias são avaliadas, uma vez que a ordem parcial estabelecida por uma restrição é mantida ao avaliar a restrição classificatória seguinte. Mais precisamente, a sequência de avaliação das restrições classificatórias é estabelecida privilegiando restrições classificatórias de provedor e, em seguida, restrições classificatórias de tipo de VM, conforme estabelecido acima. No caso de haver restrições em uma mesma categoria, a sequência é estabelecida usando a mesma sequência com que as restrições foram especificadas pelo usuário. Contudo, essa sequência de avaliação pode ser definida de forma diferente através da especificação (pelo usuário) da prioridade de cada restrição classificatória.

Tendo como base a sequência estabelecida, cada restrição classificatória na sequência só é avaliada nos casos em que não é possível estabelecer a ordem apenas considerando a restrição precedente, ou seja, quando os tipos de recurso são iguais de acordo com o critério usado na ordenação. Se esta estratégia não fosse adotada a única ordem garantida seria aquela da última restrição avaliada, sendo que a ordem estabelecida por restrições de maior prioridade poderia ser desfeita em alguns casos. Porém, é importante ressaltar que este procedimento desconsidera a avaliação de algumas

restrições classificatórias estabelecidas mas é justificável porque em alguns casos não é possível realizar a ordenação de um conjunto considerando muitos critérios.

Como forma de simplificar a implementação, os algoritmos propostos para a filtragem e classificação dos recursos são baseados na análise individual de cada tipo concreto disponível. Como trabalho futuro é sugerido o uso de alguma heurística para tornar essa etapa mais eficiente como, por exemplo, restringir o conjunto de tipos que devem ser analisados através de algum critério probabilístico.

Como resultado da filtragem e classificação dos recursos, tem-se um conjunto de tipos concretos aceitáveis para cada serviço, ordenados de acordo com as restrições classificatórias. Alternativamente, para alguns serviços pode-se ter um conjunto vazio, significando que não há nenhum tipo concreto que satisfaz todas as restrições em relação a atributos do recurso estabelecidas. Nesse caso, a estimativa de recursos deve ser refeita de forma que outros tipos canônicos de recurso sejam selecionados para os serviços para os quais ocorreu insucesso.

Quando o resultado da seleção de recursos obtém ao menos um tipo concreto para cada serviço, a próxima etapa consiste em verificar a possibilidade de consolidar os recursos selecionados.

### 6.3.2 Consolidação de recursos

O uso do modelo canônico de recursos permite que os tipos concretos sejam agregados em conjuntos representados pelos tipos canônicos. Porém, como cada tipo canônico é definido como a capacidade de *hardware* mais restrita no conjunto, pode ocorrer situações em que o tipo concreto selecionado tenha capacidade excedente àquela realmente necessária para o serviço. Além disso, mesmo com a aplicação das restrições em relação a atributos do recurso, o conjunto de tipos concretos aceitáveis para um determinado serviço pode ser grande.

Como forma de se beneficiar da variedade de tipos aceitáveis e aproveitar a provável capacidade excedente, evitando desperdícios na alocação de recursos, propomos uma estratégia de consolidação de recursos.

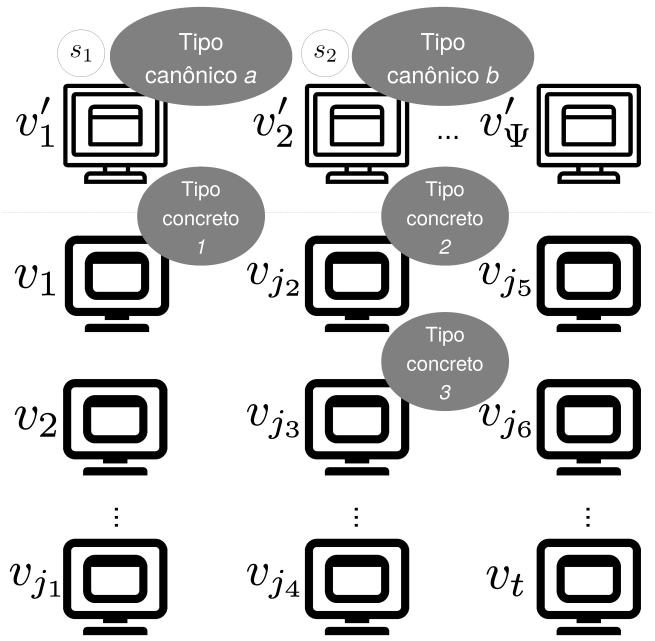
A consolidação de recursos consiste em implantar dois ou mais serviços em um único recurso em vez de alocar um recurso isoladamente para cada serviço. A consolidação pode envolver a troca de um tipo concreto por outro que também satisfaça as restrições estabelecidas e que permita a consolidação. Este problema pode ser considerado uma variação do problema de compartimento (*bin packing*) com compartimentos heterogêneos, ou de tamanho variável (*Variable Size Bin Packing Problem* – VSBPP) [68, 98, 143].

No VSBPP, dado um conjunto de itens, cada um com um certo peso, e um conjunto de compartimentos particionados em várias classes, em que cada classe

corresponde a compartimentos de uma dada capacidade e um certo custo, o objetivo é empacotar todos os itens nos compartimentos e minimizar o custo total associado aos compartimentos utilizados. Para tal, é assumido que cada classe tem um número ilimitado de compartimentos. Este problema pertence à classe NP-Completo [225], o que evidencia sua complexidade.

A estratégia de consolidação de recursos pode ser realizada quando há capacidade excedente que seja suficiente para satisfazer a demanda de todos os serviços envolvidos na consolidação e quando o custo dos recursos selecionados na consolidação é satisfatório. Neste caso, o custo é tido como satisfatório quando se mantém o mesmo que o custo obtido ao alocar cada serviço separadamente, ou apresenta um acréscimo adicional suportado, de acordo com a configuração do usuário.

A Figura 6.4 apresenta a situação em que a consolidação pode ocorrer, juntamente com as regras que devem ser satisfeitas para que ela ocorra.




(1) $\zeta_{\text{Tipo concreto } 3} \geq \zeta_{\text{Tipo concreto } 2}$

(2) $\zeta_{\text{Tipo concreto } 3} - \zeta_{\text{Tipo canônico } b} \geq \zeta_{\text{Tipo canônico } a}$

(3) $c_{\text{Tipo concreto } 3} \leq c_{\text{Tipo concreto } 1} + c_{\text{Tipo concreto } 2}$

**Figura 6.4:** *Regras para consolidação de recursos.*

Nesta figura, $s_1$ e $s_2$ são os serviços para os quais os recursos estão sendo analisados, $\zeta_v$ e $c_v$ especificam a capacidade e o custo de utilização, respectivamente, do recurso $v$, ao passo que $\{v'_1, v'_2, \ldots, v'_\Psi\}$ representam os tipos canônicos mapeados na estimativa de recursos, $\{v_1, v_2, \ldots, v_{j_1}, v_{j_2}, v_{j_3}, \ldots, v_t\}$ representam os tipos concretos re-

presentados pelos tipos canônicos considerados, e “Tipo canônico *a*”, “Tipo canônico *b*”, “Tipo concreto 1”, “Tipo concreto 2”, “Tipo concreto 3” são papéis assumidos por estes tipos nas regras apresentadas, durante uma busca por consolidação. Estas regras representam regras gerais que devem ser atendidas para que a consolidação de recursos ocorra. Elas são detalhadas a seguir:

(1) A capacidade do novo tipo concreto selecionado deve ser igual ou superior à capacidade do tipo concreto atualmente selecionado.
(2) A capacidade excedente do novo tipo concreto selecionado, isto é, a capacidade adicional à capacidade especificada pelo tipo canônico que representa este tipo concreto, deve ser igual ou superior à capacidade dos tipos canônicos selecionados para os demais serviços envolvidos na consolidação.
(3) O custo do novo tipo concreto selecionado deve ser igual ou inferior à soma dos tipos concretos atualmente selecionados para os serviços envolvidos na consolidação. Alternativamente, pode-se estabelecer um limite aceitável de gasto adicional.

Para verificar se estas regras realmente podem ser satisfeitas, analisamos os tipos concretos usados no exemplo apresentado na Seção 5.3. Consideramos todos os tipos concretos disponíveis nos provedores utilizados no exemplo, de forma que nenhuma restrição em relação a atributos do recurso foi aplicada sobre eles.

Para cada tipo concreto, verificamos se as regras (1), (2) e (3), descritas anteriormente, podem ser satisfeitas considerando tipos concretos em conjuntos representados por tipos canônicos com menor capacidade. Esta verificação foi realizada de forma isolada para cada par de conjuntos referentes aos tipos canônicos. Como exemplo, os tipos concretos referentes ao tipo canônico `SMALL` foram verificados inicialmente considerando os tipos concretos referentes ao tipo canônico `MICRO` e, em seguida, aos tipos concretos referentes ao tipo canônico `NANO`. Contabilizamos então para cada conjunto a quantidade de tipos para os quais as três regras foram satisfeitas para ao menos um tipo do conjunto comparado.

A Tabela 6.1 apresenta o resultado dessa análise. Cada linha/coluna da tabela equivale ao conjunto representado pelo tipo canônico descrito. Cada célula $[i][j]$ (onde $i$ = linha e $j$ = coluna) indica o percentual de tipos concretos no conjunto $i$ para os quais as três regras são satisfeitas ao comparar com tipos concretos do conjunto $j$. Como exemplo, para 52,53% dos tipos concretos no conjunto MICRO é possível consolidar ao menos um recurso do conjunto NANO.

Com exceção de tipos XLARGE, para os quais não há capacidade adicional uma vez que eles representam um tipo concreto específico, e de alguns casos em que tipos de um conjunto são comparados com tipos no conjunto diretamente inferior

| | NANO | MICRO | SMALL | MEDIUM | LARGE |
|---|---|---|---|---|---|
| NANO | | | | | |
| MICRO | 52,53% | | | | |
| SMALL | 31,46% | 0,00% | | | |
| MEDIUM | 37,46% | 5,28% | 0,00% | | |
| LARGE | 79,94% | 73,35% | 70,13% | 54,19% | |
| XLARGE | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |

**Tabela 6.1:** *Percentual de tipos concretos que satisfazem as regras de consolidação comparando-os com tipos em conjuntos mais restritos.*

(`SMALL` com `MICRO` e `MEDIUM` com `SMALL`), a consolidação de recursos seria possível para grande parte dos tipos concretos. Neste exemplo, essa quantidade chegou a quase 80% dos casos (ao se comparar tipos de `LARGE` com tipos de `NANO`).

Além de reduzir o número de VMs alocadas, a consolidação de recursos tem como objetivo reduzir o atraso de comunicação entre os serviços. Para isso, usamos a demanda de comunicação entre os serviços (representada pela carga $\lambda$ sobre as operações) como critério para guiar as decisões relacionadas à consolidação de recursos. Outro critério utilizado é o ganho obtido ao considerar um determinado tipo concreto. Esse ganho é estabelecido em função de uma variável que chamamos de *fator de comunicação*. Essa variável avalia a redução no atraso de comunicação ao considerar um serviço e o tipo concreto selecionado para ele, e os serviços adjacentes a ele, juntamente com os respectivos tipos concretos selecionados.

O fator de comunicação é calculado de acordo com a Equação 6-2 como sendo a média dos valores de ganho de comunicação ao considerar cada par formado pelo recurso selecionado para o serviço sendo analisado e os recursos selecionados para os serviços adjacentes a ele. Esse ganho, por sua vez, é estabelecido como a razão entre a carga de comunicação ($\lambda$) entre os serviços e a distância física entre os recursos selecionados, ou somente como a carga quando a distância é zero. Para reduzir a complexidade, não incluímos nessa equação o volume de dados trocados entre os serviços. Na equação, $n$ indica a quantidade de serviços ajacentes a um determinado serviço $a$. Usamos o fator de comunicação como um indicador do ganho ao selecionar um tipo concreto em detrimento de outro. Quanto maior o fator de comunicação, maior será a redução no atraso de comunicação.

$$\text{Fator de com.}_{a} = \frac{\sum_{i=1}^{n} \text{ganho na com.}_{(\text{tipo } a, \text{ tipo } i)}}{n}. \tag{6-2}$$

$$\text{Ganho na com.}_{(\text{tipo } a, \text{ tipo } b)} = \begin{cases} \frac{\lambda}{\text{distância}_{(\text{tipo } a, \text{ tipo } b)}}, & \text{se distância} > 0; \\ \lambda, & \text{caso contrário.} \end{cases} \tag{6-3}$$

O Algoritmo 6.1 descreve o procedimento para selecionar recursos a serem consolidados. Ele consiste em avaliar os tipos concretos remanescentes para cada serviço após a filtragem e classificação de recursos. Inicialmente, cada serviço é mapeado para o primeiro tipo concreto disponível. Os demais tipos são então avaliados (a partir da linha 8 no algoritmo) de forma a verificar a possibilidade de consolidação ou redução no atraso de comunicação pela troca por outros tipos concretos. Os serviços são avaliados em ordem decrescente do valor de corte do vértice equivalente no grafo de dependências. Definimos o corte de um vértice como sendo a soma da carga de comunicação do serviço, representada como parte do peso em cada aresta incidente ao vértice em questão, quando o vértice adjacente também representa um serviço.

A condição na linha 12 verifica a possibilidade de realizar a busca por consolidação, que só ocorre nos casos em que a capacidade excedente no tipo concreto corrente é suficiente para consolidar outros recursos. Para tal, ela deve ser igual ou superior à demanda do tipo canônico menos robusto. Além disso, essa capacidade deve ser superior à melhor capacidade excedente tida até então, acrescida da capacidade necessária para melhorar a melhor solução até então encontrada.

Outro fator que gera a possibilidade de busca por consolidação ocorre quando o tipo concreto analisado tem possibilidade de reduzir o atraso de comunicação, o que é determinado pelo fator de comunicação.

Apesar da quantidade de tipos concretos a serem avaliados ser potencialmente grande, a busca por consolidação não ocorre sempre, pois ela será realizada apenas nos casos em que realmente há possibilidade de haver algum ganho.

A busca por consolidação é implementada pelo Algoritmo 6.2, invocado na linha 13. Esse algoritmo recebe como entrada o serviço e o tipo concreto atualmente sendo avaliados. Seu resultado é o conjunto de serviços para os quais a consolidação nesse tipo concreto ocorreu, ou um conjunto unitário contendo apenas o serviço fornecido como entrada, significando que não é possível haver consolidação. Como resultado secundário, esse algoritmo indica a capacidade adicional de recurso necessária para obter um resultado melhor na consolidação, o qual é usado nas buscas seguintes, conforme descrito anteriormente.

A condição na linha 14 determina os casos em que há a troca do tipo concreto até então selecionado pelo atualmente em análise. Isso ocorre quando há redução no atraso de comunicação, ou seja, quando o número de serviços envolvidos na consolidação é aumentado e/ou quando o fator de comunicação do tipo em análise é maior que o do tipo até então selecionado.

A busca pela consolidação é interrompida para um determinado serviço caso todos os demais serviços sejam envolvidos na consolidação para um dos recursos considerados (linha 17), uma vez que não é possível obter resultado mais satisfatório.

**Algoritmo 6.1:** Consolidação de recursos.

**Entrada:** $\mathcal{G}$: grafo de dependências; $f : \mathcal{S} \rightarrow \mathcal{V}$: tipos canônicos selecionados; $\mathscr{V}[]$: um vetor que representa os tipos concretos remanescentes, em que $\mathscr{V}[j], 1 \leq j \leq \Psi$, especifica os tipos concretos remanescentes no conjunto representado pelo tipo canônico $j$.

**Saída:** $f' : \mathcal{S}' \subseteq \mathcal{S} \rightarrow \mathcal{V}$: tipos concretos selecionados.

**Dados:** $\mathcal{S}$: conjunto de serviços em $\mathcal{G}$, $t$: número de tipos canônicos.

```
1  Marque todos os serviços s ∈ S como não mapeados
2  enquanto existir serviço não mapeado faça
3      maxP ← índice do vértice p com maior corte em G | p ∈ S e p ainda não foi mapeado
4      tiposCandidatos ← V[maxP]
5      melhorTipo ← null, melhorRecursoAdic ← 0, melhorFatorCom ← 0
6      minDemandaRec ← capacidade do tipo canônico menos robusto
7      capacParaMelhorar ← 0
8      enquanto tiposCandidatos não está vazia faça
9          tipoCorrente ← tiposCandidatos.first()
10         recursoAdic ← Recurso excedente de tipoCorrente
11         fatorCom ← Fator de comunicação de tipoCorrente
12         se recursoAdic ≥ minDemandaRec e (recursoAdic ≥ melhorRecursoAdic +
             capacParaMelhorar ou fatorCom > melhorFatorCom) então
13             busca_consolidação(G, f, V, maxP, tipoCorrente)
14             se há redução no atraso de comunicação então
15                 melhorTipo ← tipoCorrente, melhorRecursoAdic ← recursoAdic,
                     melhorFatorCom ← fatorCom
16                 se número de serviços consolidados = |S| então
17                     Remove todos os tipos em tiposCandidatos
18                 senão
19                     Remove tipoCorrente de tiposCandidatos
20                 fim
21             fim
22         senão
23             Remove tipoCorrente de tiposCandidatos
24         fim
25         se tipoCorrente é o tipo mais robusto no conjunto então
26             Remove todos os tipos em tiposCandidatos
27         fim
28     fim
29     f'( serviços envolvidos na consolidação) ← melhorTipo
30     para cada serviço s ∈ grupo de consolidação faça
31         Marque s como mapeado
32     fim
33 fim
34 retorna f'
```

Outra condição para interromper a busca para um determinado serviço é quando o tipo concreto mais robusto para este serviço foi avaliado (linha 26), nova-

mente porque nenhuma solução envolvendo os outros tipos será mais satisfatória que a encontrada até então.

O resultado da consolidação é estabelecido como um mapeamento de um conjunto de serviços para o tipo concreto que possibilitou o melhor resultado na consolidação (linha 29). Esse conjunto contém necessariamente o serviço com o qual a busca foi iniciada, além dos demais serviços envolvidos na consolidação. Nos casos em que não é possível realizar consolidação, o conjunto conterá apenas o serviço com o qual a busca foi iniciada, sendo o tipo concreto estabelecido como aquele que obteve melhor fator de comunicação ao considerar os serviços adjacentes.

O fato do mapeamento ser estabelecido como resultado de uma busca não indica que os serviços permanecerão mapeados para o tipo selecionado, uma vez que grupos já mapeados também são considerados em buscas seguintes. Dessa forma, quando for possível, o grupo como um todo poderá ser envolvido em uma consolidação para outro tipo de recurso. Este processo ocorrerá até que todos os serviços sejam mapeados.

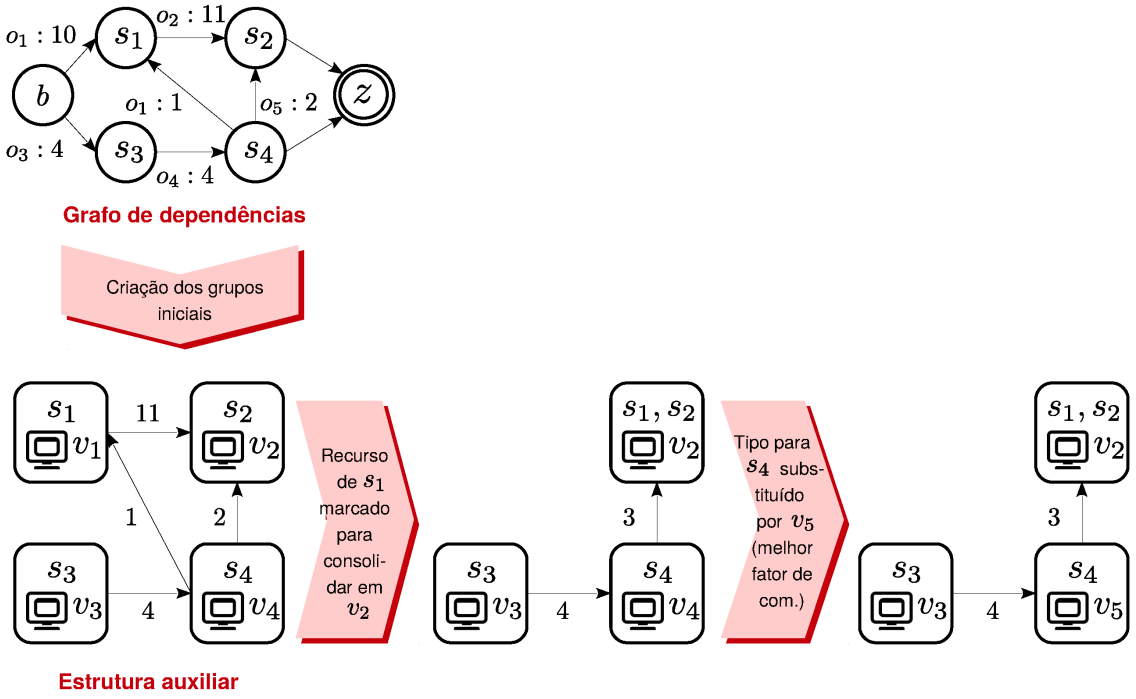



**Figura 6.5:** *Exemplo de criação e atualização da estrutura auxiliar utilizada na busca por consolidação.*

Para realizar a busca por consolidação definimos uma nova estrutura de grafo auxiliar, criada a partir do grafo de dependências e dos resultados de buscas anteriores. Nessa estrutura, os vértices representam grupos de serviços envolvidos em uma consolidação, mapeados para um determinado recurso no algoritmo principal. Inicialmente é criado um grupo para cada serviço, que contém apenas esse serviço mapeado para

o primeiro tipo concreto disponível para ele. Esses grupos podem então ser alterados após a realização de cada busca por consolidação. As arestas nessa estrutura representam a carga de comunicação entre os serviços que compõem cada grupo. A Figura 6.5 ilustra como essa estrutura é criada e atualizada. Por simplicidade, não apresentamos todo o modelo de recursos na figura.

Inicialmente é criado um grupo para cada um dos serviços ($s_1, s_2, s_3, s_4$) e o primeiro tipo concreto aceitável para estes serviços ($v_1, v_2, v_3$ e $v_4$, neste caso). Apenas as arestas que ligam pares de serviços são mantidas nessa estrutura. No exemplo, o recurso utilizado por $s_1$ foi marcado para consolidar com $s_2$. Dessa forma, o grupo original com $s_1$ deixa de existir, e esse serviço é adicionado ao grupo de $s_2$. O exemplo também ilustra outra transformação possível na estrutura, que ocorre quando não é possível haver consolidação mas um novo tipo concreto é selecionado por oferecer melhor fator de comunicação. Neste caso, o tipo concreto para o qual o serviço $s_4$ será mapeado é trocado de $v_4$ para $v_5$.

O procedimento de busca por consolidação, descrito no Algoritmo 6.2, se baseia em uma busca em largura (*Breadth-First Search* – BFS) [264] no grafo de dependências. A busca é realizada enquanto houver grupos não visitados e recurso excedente (linha 6). A ordem de avaliação de grupos é estabelecida de acordo com a carga agregada dos serviços neles contidos, visando consolidar aqueles que têm maior demanda de comunicação.

Cada grupo adjacente ao grupo atualmente avaliado é inicialmente consultado para ver a possibilidade dos serviços desse grupo serem implantados no tipo concreto fornecido como parâmetro. Essa verificação é realizada na linha 10 através da aplicação das restrições em relação a atributos do recurso referentes aos serviços contidos no grupo. Caso algum serviço do grupo não possa ser implantado no tipo concreto em questão, o grupo é desconsiderado. Caso contrário (e se o grupo ainda não tiver sido visitado), a demanda de recurso desse grupo é somada usando o tipo canônico selecionado para cada serviço no grupo (linha 14). Essa demanda é então comparada com a capacidade excedente no recurso usado na busca para checar se a consolidação é possível (linha 15).

Outra verificação realizada é se o custo adicional (caso ele exista) é aceitável (linha 16), o que é avaliado de acordo com a configuração de custo adicional admissível, estabelecida pelo usuário. Apenas se ambas as condições forem satisfeitas é que o grupo é marcado em uma consolidação. Neste caso a busca é interrompida se não há mais capacidade excedente (linha 19). Nos casos em que não é possível consolidar porque a capacidade excedente não é suficiente, a capacidade necessária para melhorar a solução é atualizada (linha 22) como sendo a capacidade adicional que seria necessária para permitir a consolidação deste grupo, ou seja, a diferença entre a demanda de

recurso para todos os serviços no grupo e a capacidade adicional nesta iteração.

**Algoritmo 6.2:** *busca_consolidação*

**Entrada:** $\mathcal{G}$: grafo de dependências; $f : \mathcal{S} \rightarrow \mathcal{V}$: tipos canônicos selecionados; $\mathcal{V}[]$: um vetor que representa os tipos concretos remanescentes, em que $\mathcal{V}[j], 1 \leq j \leq \Psi$ especifica os tipos concretos remanescentes no conjunto representado pelo tipo canônico $j$; *maxP*: vértice com corte máximo em $\mathcal{G}$; *tipoCorrente*: tipo concreto a ser analisado

**Saída:** Grupo de serviços a ser consolidados e recurso necessário para melhorar essa solução

**Dados:** $\mathcal{S}$: conjunto de serviços em $\mathcal{G}$, $t$: número de tipos canônicos

```
1  recursoAdic ← Recurso excedente do tipo corrente
2  grupoInicial ← grupo contendo maxP
3  queue ← grupoInicial
4  Marque todos os grupos como não visitados
5  Marque grupoInicial como visitado
6  BFS: enquanto queue não estiver vazia e recursoAdic > 0 faça
7      grupoCorrente ← queue.remove()
8      gruposAdj ← Grupos adjacentes a grupoCorrente ordenados pela carga agregada
9      para cada grupoAdj ∈ gruposAdj faça
10         se tipoCorrente não satisfaz todas as restrições para algum serviço em grupoAdj então
             continue
11         se grupoAdj não foi visitado então
12             queue.add(grupoAdj)
13             Marque grupoAdj como visitado
14             recursoGrupoAdj ←demanda de recurso de grupoAdj
15             se recursoGrupoAdj ≤ recursoAdic então
16                 se o custo adicional é aceitável então
17                     Marque grupoAdj para ser consolidado
18                     recursoAdic ← recursoAdic - recursoGrupoAdj
19                     se recursoAdic = 0 então break BFS
20                 fim
21             senão
22                 recursoParaMelhorar ← recursoGrupoAdj − recursoAdic
23             fim
24         fim
25     fim
26 fim
```

A Figura 6.6 ilustra parte da execução do algoritmo de consolidação. À esquerda é apresentada a entrada, que consiste no grafo de dependências e o mapeamento de cada serviço para um tipo canônico, juntamente com os tipos concretos remanescentes após a aplicação das restrições em relação a atributos do recurso. À esquerda de cada tipo é apresentado seu identificador, ao passo que à sua direita é apresentado um valor numérico que indica sua capacidade. Para os tipos concretos também é apresentado na figura o custo.

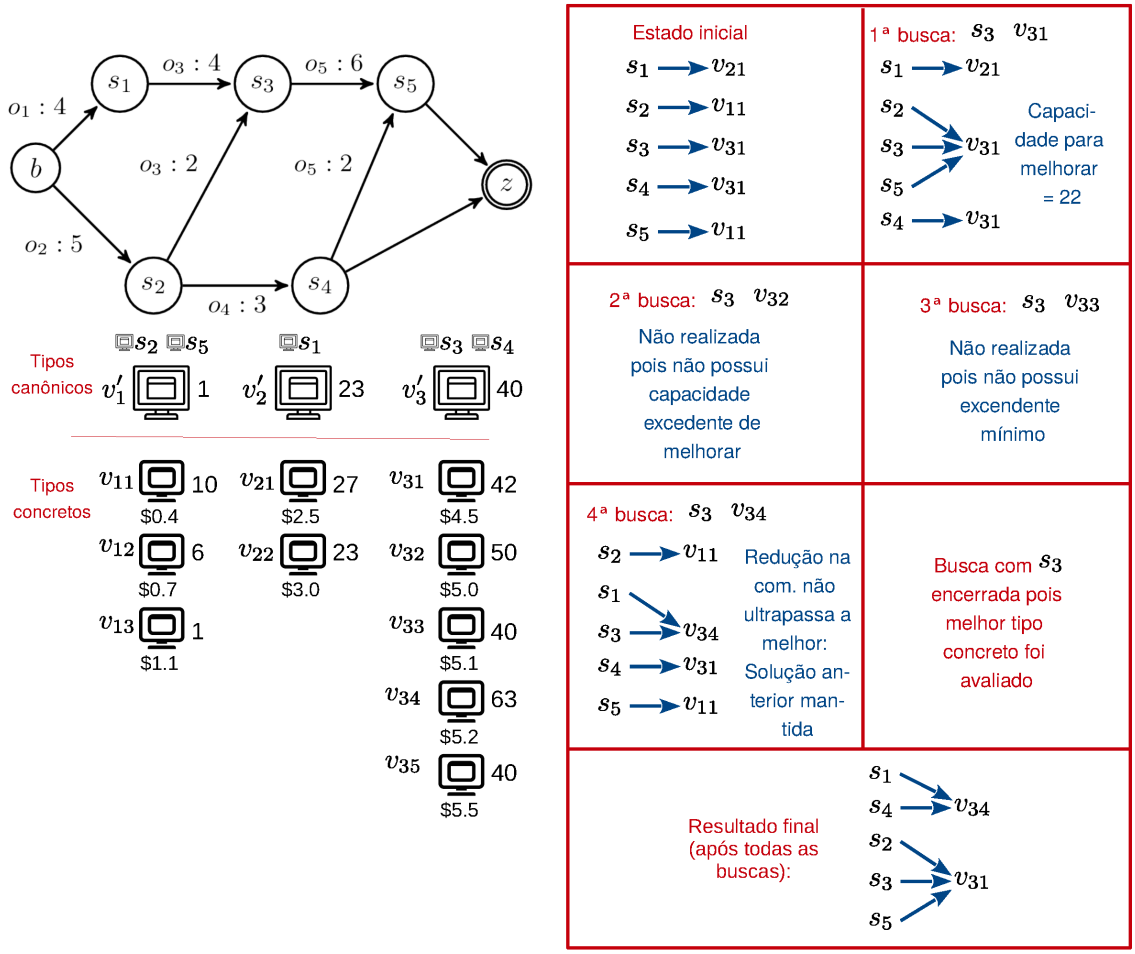



**Figura 6.6:** *Exemplo de execução do algoritmo de consolidação.*

Nesta entrada, por simplicidade, assumimos que o resultado da filtragem e classificação dos tipos concretos é igual quando utilizado o mesmo tipo canônico. O quadro na parte direita apresenta parte do processo, considerando as buscas a partir do serviço $s_3$, que equivale ao vértice com maior corte, ou seja, o vértice que possui precedência na avaliação dos tipos concretos no Algoritmo 6.1 (linha 3). A partir deste vértice as buscas serão realizadas no Algoritmo 6.2 considerando a ordem definida pela carga agregada. Por questão de espaço desconsideramos a análise do fator de comunicação neste exemplo, e usamos apenas a demanda expressa pela carga.

O algoritmo inicia com cada serviço sendo mapeado para o primeiro tipo concreto disponível. No exemplo dado, a primeira chamada ao procedimento de busca é realizada com o tipo $v_{31}$. O primeiro grupo adjacente a ser avaliado é aquele que contém $s_5$ devido à demanda maior de comunicação. O recurso para o qual esse serviço foi mapeado é então marcado para consolidar, pois sua demanda de recursos não ultrapassa a capacidade adicional de $v_{31}$ e não há custo adicional. Como ainda há recurso adicional, a busca avalia o próximo serviço $s_1$ mas o recurso que esse serviço foi mapeado não pode ser consolidado, uma vez que sua demanda é maior que a

capacidade remanescente: sua demanda é 23 e a capacidade remanescente é 1. Nesse ponto, a capacidade necessária para melhorar a solução é estabelecida então como 22, por ser a capacidade adicional necessária para permitir a consolidação do recurso para o qual $s_1$ foi mapeado. Em seguida, $s_2$ é avaliado, sendo seu recurso também marcado para consolidação. Como não há mais capacidade excedente após a avaliação de $s_2$, a busca é encerrada com os recursos de $s_5$ e $s_2$ marcados para consolidar no recurso de $s_3$, e $s_1$ e $s_4$ mantidos nos recursos originais.

O próximo tipo, $v_{32}$, não é avaliado pois não possui capacidade excedente suficiente para melhorar a solução previamente encontrada. Para ser possível melhorar a solução, a capacidade deveria ser maior ou igual à soma da capacidade excedente do melhor tipo até então encontrado ($v_{31} = 2$) com a capacidade não disponível na busca anterior (22), ou seja, deveria ser no mínimo 24 (mas é somente 10). O mesmo acontece para o próximo tipo $v_{33}$, mas neste caso ele não é avaliado porque não possui nem mesmo a capacidade excedente mínima para permitir que o recurso seja consolidado, que é igual à demanda do tipo canônico menos robusto (1 em nosso exemplo).

Continuando o procedimento, a busca usando o tipo $v_{34}$ também é realizada de acordo com a ordem da demanda de comunicação. Ao avaliar o serviço $s_5$, seu recurso não é marcado para consolidar porque o custo adicional não é aceitável, supondo neste caso que não há tolerância quanto ao custo adicional. Isso acontece porque a soma do custo dos recursos inicialmente utilizados por $s_3$ = \$4.5 e $s_5$ = \$0.4 é menor que o custo de $v_{34}$ = \$5.2, não justificando a troca, pois o custo será menor ao alocar os recursos sem a consolidação. Em virtude disso, a busca prossegue com $s_1$ e finaliza com o recurso deste serviço sendo consolidado pois não há mais capacidade excedente. O resultado dessa busca é então comparado com o melhor resultado até então obtido (que era usando $v_{31}$). Como a quantidade de serviços cujo recursos foram consolidados, e consequentemente a redução na demanda de comunicação, anteriormente obtida é maior, o resultado anterior é mantido. Nesse momento, as buscas partindo do serviço $s_3$ são encerradas, pois o tipo de recurso concreto mais robusto foi avaliado. Somente nesse instante os grupos envolvendo serviços para os quais houve mudança no mapeamento são atualizados na estrutura auxiliar, sendo os serviços $s_3$, $s_5$ e $s_2$ marcados como mapeados. O procedimento se repete novamente considerando os serviços que ainda não foram mapeados, $s_1$ e $s_4$. O resultado final da consolidação é apresentado no final do quadro, onde, apesar de não terem comunicação entre si, o recurso de $s_1$ foi marcado para consolidar no recurso de $s_4$ devido à capacidade excedente, diminuindo o gasto com os recursos selecionados.

O exemplo apresentado ilustra os dois tipos de redução de custos proporcionados pela técnica de consolidação: redução no atraso de comunicação e no custo dos recursos alocados.

## 6.4 Casos de insucesso da síntese de recursos

A síntese de recursos está sujeita a alguns casos em que não é possível obter os resultados das atividades de estimativa e seleção de recursos. Como a estimativa de recursos é realizada considerando apenas as restrições de QoS sem, portanto, avaliar as restrições em relação a atributos do recurso, pode ocorrer que nenhum tipo concreto representado pelo tipo canônico selecionado satisfaça todas as restrições impostas sobre o serviço em questão. Com isso, ocorrerá insucesso na seleção pois não será possível implantar o serviço em um recurso com capacidade estabelecida pelo tipo canônico selecionado. Esse tipo de problema ocorre principalmente porque não é possível garantir que os tipos concretos representados por um determinado tipo canônico possuam alguma propriedade além daquela relacionada à sua capacidade. Ou seja, não é possível garantir, por exemplo, que haverá tipos concretos que são disponibilizados por um determinado provedor, ou que haverá tipos concretos que podem ser instanciados em uma certa localidade.

Para a seleção de recursos possa ser realizada de forma satisfatória, a estimativa de recursos deve desconsiderar os tipos canônicos para os quais não haverá na seleção ao menos um tipo concreto que satisfaça todas as restrições em relação a atributos do recurso. Uma forma de garantir que isso ocorra é avaliar essas restrições antes de realizar a estimativa de recursos, como forma de desconsiderar para um serviço os tipos canônicos para os quais todos os tipos concretos foram eliminados. Ao fazer isso, a seleção de recursos passaria a ser imediata pois o primeiro tipo concreto disponível no conjunto representado pelo tipo canônico seria considerado factível. Contudo, para implementar essa funcionalidade é preciso avaliar as restrições eliminatórias para todos os pares formados por serviços e tipos concretos disponíveis, o que não é uma solução eficiente, dado que a complexidade desta tarefa é $O(\rho^e n t)$, onde $\rho^e$ é o número de restrições eliminatórias, $n$ é o número de serviços e $t$ é o número de tipos concretos de recurso. Apesar de ser uma tarefa com complexidade polinomial, o número de tipos concretos pode ser alto se considerarmos múltiplos provedores. Como exemplo, considerando os provedores utilizados na Seção 5.3 para demonstrar a geração dos tipos canônicos, foram utilizados 827 tipos concretos.

Em virtude disso, propomos outra abordagem mais simples, que consiste em detectar insucesso na seleção de recursos e refazer a etapa de estimativa, desconsiderando os tipos canônicos para os serviços para os quais insucesso foi detectado. Apesar de parecer ineficiente, essa abordagem obtém um índice menor de tempo de processamento que a proposta anterior porque a estimativa se baseia nos tipos canônicos, cuja quantidade é necessariamente ordens de magnitude menor que a quantidade de tipos concretos. Dessa forma, mesmo que a estimativa tenha que ser refeita (o que não

acontecerá com tanta frequência) o resultado será obtido em menos tempo do que usando a solução anterior.

Outro cenário de insucesso possível na síntese é não haver tipos canônicos com capacidade suficiente para satisfazer todas as restrições de QoS. Isso ocorre quando a carga é muito elevada e/ou quando a demanda de recursos para processar determinadas operações é muito elevada. Nestes casos, mesmo selecionando o tipo canônico mais robusto para todos os serviços não é possível satisfazer as restrições impostas. Isso acontece porque todos os tipos concretos representados pelo tipo canônico mais robusto têm exatamente a capacidade de hardware expressa neste tipo, o que significa que não há disponível nenhum tipo cuja capacidade seja superior.

Para lidar com esse tipo de problema propomos a adição de instâncias dos serviços. Para tal, assumimos que há um balanceador de carga com sobrecarga negligenciável, que escalona as requisições recebidas de maneira igual entre as instâncias de um determinado serviço. Uma vez que essas instâncias não são entidades de primeira classe na modelagem dos serviços e visando reduzir o tempo de processamento da síntese, as instâncias adicionais não são explicitamente representadas no grafo de dependências. Em vez disso, apenas a carga de entrada é alterada como forma de simular a inclusão de novas instâncias.

A Figura 6.7 mostra como o uso de novas instâncias é considerado no modelo, onde $s_i$ é o serviço sobrecarregado para o qual uma nova instância foi adicionada. Apesar das novas instâncias existirem logicamente, como mostrado no esquema apresentado no lado superior direito da figura, no grafo de dependências apenas a carga sobre o serviço que teve novas instâncias criadas é alterada. Isso faz com que a síntese de recursos seja realizada de forma mais eficiente, pois não é preciso estimar e selecionar recursos para cada instância separadamente. Essas tarefas são realizadas apenas uma vez e o resultado é então reproduzido para cada instância do serviço. A informação sobre a quantidade de instâncias é mantida no descritor do serviço.

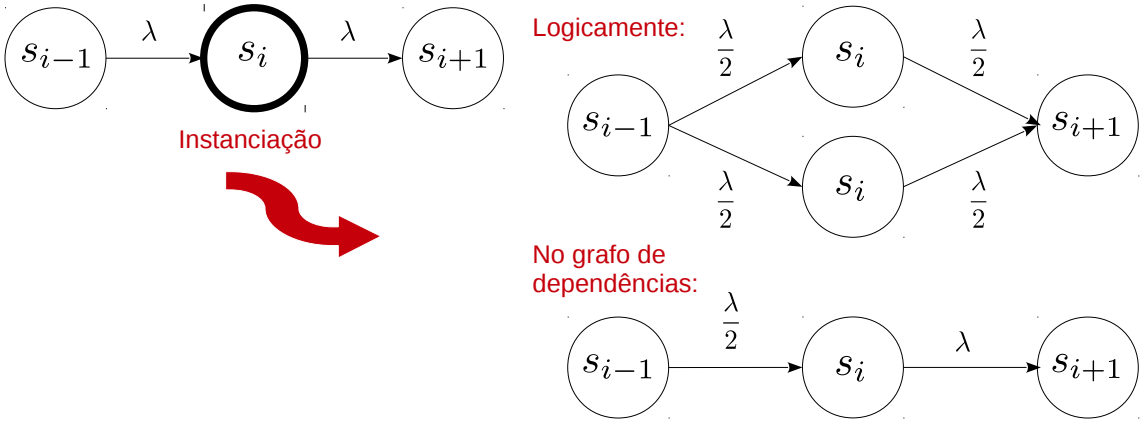



**Figura 6.7:** *Forma de inclusão de novas instâncias na modelagem de serviços.*

A estratégia usada para identificar os serviços que devem ter instâncias adicionadas é discutida no próximo capítulo, onde apresentamos detalhes sobre a implementação das estratégias aqui discutidas.

## 6.5 Considerações finais

A síntese de recursos consiste em determinar a capacidade de recurso necessária para implantar um conjunto de serviços coreografados e selecionar o tipo de recurso mais apropriado para oferecer essa capacidade. Neste capítulo, descrevemos nossa estratégia para solucionar este problema.

A Figura 6.8 descreve as etapas da nossa estratégia de síntese de recursos. Para cada etapa, são apresentadas as entradas (acima) e as saídas obtidas (abaixo).

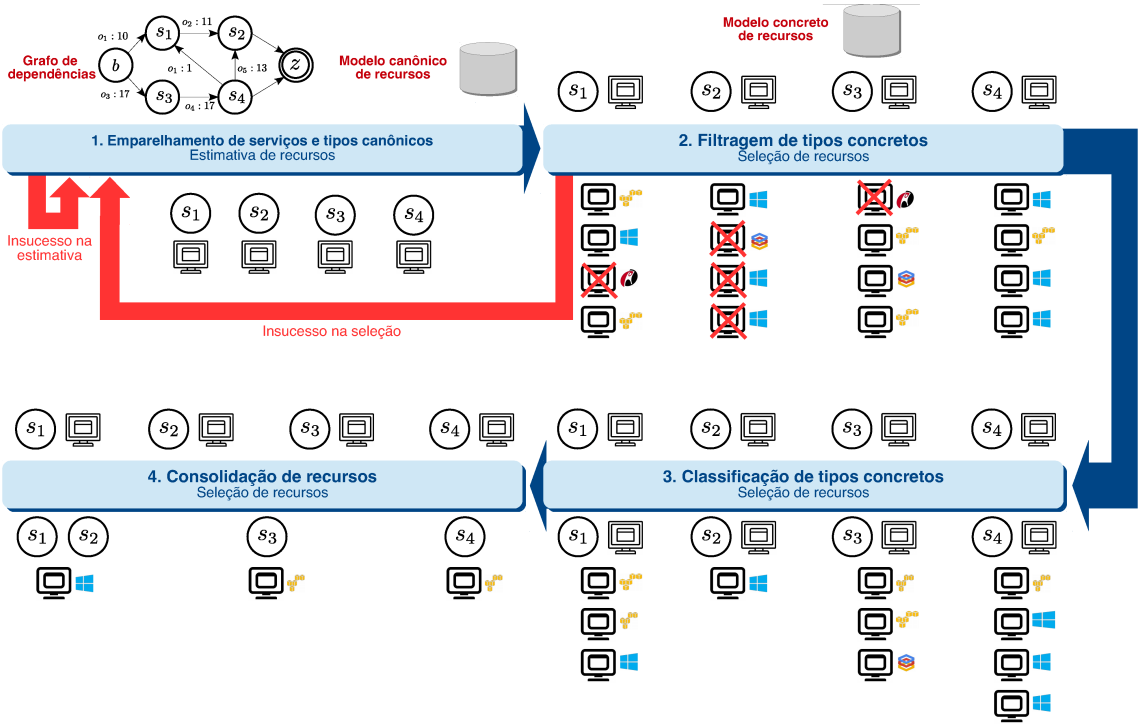



**Figura 6.8:** *Etapas da síntese de recursos.*

A primeira etapa realizada é a estimativa de recursos, que consiste em determinar, com base na representação do grafo de dependências e no modelo canônico de recursos, a capacidade de recurso necessária para satisfazer as restrições de QoS. Propomos que essa atividade seja realizada através do emparelhamento dos serviços com tipos canônicos, usando uma estratégia baseada no problema MMKP. Nessa etapa também é avaliado se instâncias únicas dos serviços serão suficientes para atender a demanda e, caso não seja (o que é caracterizado como insucesso), novas instâncias são criadas para os serviços. A saída da estimativa de recursos é o mapeamento de cada serviço (juntamente com possíveis instâncias adicionais) para um tipo canônico.

A próxima etapa da síntese consiste na primeira tarefa da seleção de recursos, que é responsável por filtrar os tipos concretos disponíveis para cada tipo canônico. Essa filtragem é realizada de acordo com as restrições eliminatórias estabelecidas sobre os serviços. De maneira semelhante, as restrições classificatórias são avaliadas para classificar esses tipos na etapa seguinte. A saída de ambas as etapas é o mapeamento de cada serviço para um conjunto de tipos concretos ou insucesso na síntese por não haver tipos concretos que satisfaçam todas as restrições eliminatórias. No caso de insucesso, a estimativa de recursos é refeita ignorando o tipo canônico que causou o insucesso para o serviço.

Após a avaliação das restrições em relação a atributos do recurso, a última etapa da síntese consiste em verificar a possibilidade de realizar a consolidação de recursos e avaliar a redução no atraso de comunicação proporcionada pela troca de tipos concretos. A saída dessa etapa, e consequentemente da síntese de recursos, é o mapeamento de um serviço (ou conjunto de serviços se houve consolidação) para o tipo concreto que será usado na implantação.

A síntese de recursos consiste na principal contribuição desta tese. Ela permite que um conjunto de coreografias seja implantado de maneira eficiente, satisfazendo um conjunto de restrições não-funcionais e levando em consideração um vasto conjunto de tipos de recursos. Para agregar as técnicas apresentadas neste capítulo (e nos capítulos anteriores) propomos uma arquitetura para implantação de coreografias, que será descrita no próximo capítulo.

CAPÍTULO 7

# Arquitetura e implementação

Anteriormente, discutimos os problemas tratados nesta tese e apresentamos as técnicas que propomos para resolvê-los. Mais precisamente, propomos estratégias para a representação de coreografias de serviços e restrições não-funcionais associadas e para a modelagem de recursos disponibilizados em um ambiente com múltiplos provedores de nuvem. Com base nessa representação, propomos abordagens para a estimativa e a seleção de recursos, visando à implantação de um conjunto de coreografias, com provável compartilhamento de serviços entre elas. Contudo, para que os objetivos estabelecidos nesta tese sejam de fato alcançados, é preciso que o uso das abordagens propostas seja feito de maneira conjunta e coordenada. Diante disso, apresentamos neste capítulo uma arquitetura proposta com este objetivo.

A arquitetura considera o gerenciamento de recursos tanto em tempo de implantação quanto em tempo de execução dos serviços que compõem as coreografias. Dessa forma, ela pode ser usada para estimar e selecionar os recursos que serão usados na implantação dos serviços e também para garantir que as restrições não-funcionais sejam satisfeitas durante a encenação das coreografias. Para ilustrar o uso da arquitetura discutimos cenários em que ela pode ser aplicada.

Nesse capítulo, também discutimos a implementação de um protótipo de parte da arquitetura proposta. Este protótipo foi implementado para demonstrar e avaliar a abordagem proposta.

## 7.1 Arquitetura

Propomos uma arquitetura que fornece abstração no gerenciamento de recursos para múltiplas coreografias de serviços. Esta arquitetura se baseia no processamento de modelos por camadas de funcionalidades que transformam os elementos do modelo até que comandos sejam gerados sobre a infraestrutura subjacente. Desta forma, como ilustrado na Figura 7.1, a arquitetura proposta é estruturada em três camadas: síntese de coreografias, síntese de recursos e *broker* de recursos.

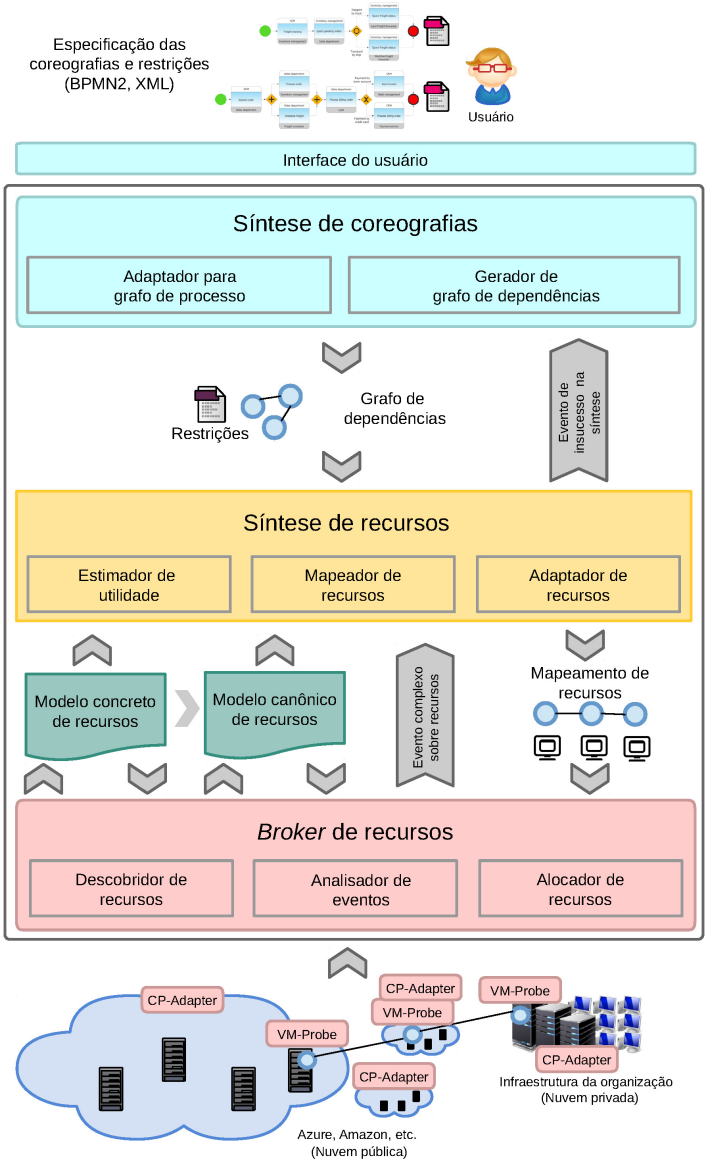



**Figura 7.1:** *Arquitetura geral proposta.*

A primeira camada da arquitetura – ***síntese de coreografias*** – constitui o ponto de interação com o usuário, através do componente ***Interface do usuário***. A especificação da(s) coreografia(s) de serviços, juntamente com restrições não-funcionais associadas, é submetida pelo usuário através desse componente. Nossa abordagem é melhor aplicada se as coreografias forem submetidas em lote, já que o compartilhamento de serviços pode ser melhor analisado neste caso. No entanto, a implantação de coreografias também pode ser realizada com a submissão de cada coreografia isoladamente, mas possivelmente exigindo adaptação em serviços previamente implantados.

Propomos o uso de adaptadores para a representação interna. Dessa forma, desde que o adaptador equivalente esteja implementado como parte da arquitetura (componente ***Adaptador para grafo de processo***), a especificação das coreografias pode ser feita em qualquer linguagem usada para este propósito (como BPMN 2.0, por

exemplo). Dado que a seleção de serviços não faz parte do escopo considerado neste trabalho, assumimos que a modelagem das coreografias é feita indicando implementações concretas dos serviços.

Além de interagir com o usuário, a camada de síntese de coreografias também é responsável por gerar a estrutura do grafo de dependências, que consiste na representação conjunta das coreografias e das restrições não-funcionais associadas e submetidas como entrada. Essa tarefa é realizada pelo componente ***Gerador de grafo de dependências***. A estrutura do grafo de dependências é usada na próxima camada para guiar as atividades de estimativa e seleção de recursos.

A ***síntese de recursos*** é a principal camada da arquitetura. Ela é responsável por estimar e selecionar recursos usando as abordagens discutidas no capítulo anterior. Para isso, além do grafo de dependências ela utiliza os modelos de recurso (concreto e canônico). O principal componente na síntese de recursos é o ***Mapeador de recursos***, que implementa tanto a estimativa quanto a seleção de recursos. A primeira tarefa é realizada usando as estimativas de contribuição de QoS de acordo com os tipos canônicos de recurso. A estimativa da função de utilidade associada às métricas de QoS é abstraída no componente ***Estimador de utilidade***. Assumimos que esta tarefa pode ser implementada independentemente do mapeamento de recursos. Dessa forma, isolamos o problema da estimativa de QoS e focamos nas especificidades do mapeamento de recursos.

A saída da síntese de recursos é o mapeamento de conjuntos de serviços coreografados para o tipo concreto selecionado para cada conjunto. O tamanho de cada conjunto dependerá da possibilidade de realizar consolidação de recursos. Nos casos em que isso não é possível, os conjuntos corresponderão a conjuntos unitários contendo serviços isolados.

Nos casos em que não é possível satisfazer todas as restrições não-funcionais com os recursos disponíveis, um evento de insucesso é gerado para a camada superior que notifica o usuário. Neste caso, ele/ela deve selecionar outra implementação de serviço ou relaxar as restrições especificadas para o serviço ou coreografia para a qual ocorreu insucesso.

Outra tarefa executada na camada de síntese de recursos, através do componente ***Adaptador de recursos***, é a adaptação da alocação de recursos frente às mudanças em tempo de execução, como aumento ou diminuição da carga de trabalho.

A implantação dos serviços é realizada pela camada inferior – ***Broker de recursos*** –, através do componente ***Alocador de recursos***, usando como entrada o modelo de recursos gerado pela síntese. Essa camada gerencia a infraestrutura privada e interage com provedores de nuvem pública através do envio de comandos por adaptadores. Outra tarefa implementada nesta camada é a geração dos modelos concreto e

canônico de recursos. Para tal, o componente ***Descobridor de recursos*** obtém a capacidade disponível em máquinas físicas, no caso de nuvens privadas, e os tipos de VM disponibilizados por provedores de nuvem pública. O gerenciamento de recursos em nuvens públicas é realizado agregando todos os provedores em um mesmo canal por meio de um *endpoint* gerenciável (que interage com adaptadores para cada provedor – ***CP-Adapter***). Somente são considerados provedores previamente registrados nesta camada e para os quais há um adaptador.

Apesar de não termos implementado no protótipo desenvolvido, a arquitetura prevê que durante a encenação das coreografias, o componente ***VM-Probe*** execute, em cada VM alocada, o monitoramento contínuo para detectar possíveis violações da satisfação nas restrições não-funcionais especificadas. Dessa forma, quando ocorresse mudanças, como aumento no número de requisições, o modelo de recursos seria verificado pelo componente ***Analisador de eventos*** para avaliar se a satisfação de restrições ainda é possível, dado o novo cenário. Propomos que esta verificação se baseie no uso de Processamento de Eventos Complexos (*Complex Event Processing* – CEP) [169], uma técnica para monitoramento e execução de tarefas reativas em sistemas distribuídos, através da análise de correlação entre eventos. A partir dos dados monitorados, seria realizada uma análise lógico-temporal para correlacionar eventos capturados, gerando eventos complexos que seriam usados para inferir causas e possíveis soluções frente a violações das restrições. A implementação destes mecanismos é tida como trabalho futuro.

Caso as mudanças verificadas sejam admissíveis com a atual configuração de recursos, a encenação da coreografia não é interrompida. Senão, quando não for possível satisfazer todas as restrições para o novo cenário (por exemplo, quando os recursos selecionados são insuficientes), uma adaptação na síntese de recursos é acionada com o envio de um evento complexo sobre recursos. Com base neste evento, o componente ***Adaptador de recursos*** decide qual a melhor estratégia para solucionar as violações detectadas, o que provavelmente resultará na execução de uma nova síntese de recursos, com o redimensionamento ou aumento/diminuição no número de VMs previamente alocadas; ou, alternativamente, no envio de um evento de insucesso na síntese, para os casos em que não é possível reparar as violações (como por exemplo, quando não há recursos que satisfaçam todas as restrições).

## 7.2 Cenário de uso da arquitetura

Conforme ilustrado na Figura 7.2, a arquitetura proposta pode ser classificada como uma solução de plataforma como serviço (PaaS) para o gerenciamento de recursos, que interage com provedores na categoria de infraestrutura como serviço (IaaS)

para permitir a implantação de composições de serviços gerenciadas por provedores de *software* como serviço (SaaS).

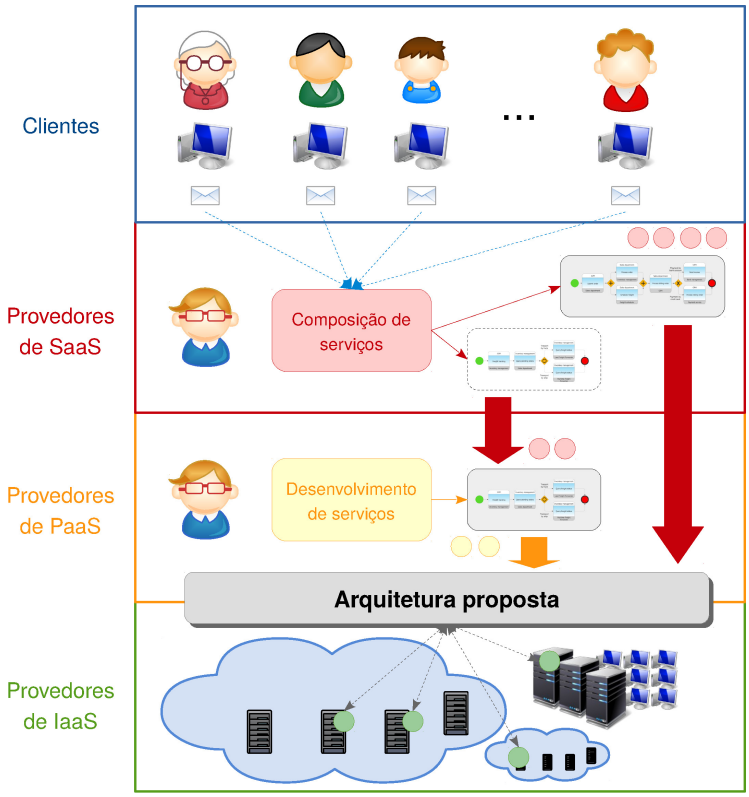



**Figura 7.2:** *Cenário de uso da arquitetura.*

Adicionalmente, os provedores de serviço podem desenvolver suas soluções usando funcionalidades oferecidas por provedores de PaaS que, dessa forma, podem utilizar a arquitetura proposta para implantar os serviços. Porém, é importante destacar que a classificação da arquitetura como uma solução de PaaS não inclui todos os elementos tipicamente encontrados em soluções desta categoria [157]. As facilidades oferecidas pela arquitetura como plataforma são limitadas ao gerenciamento de recursos, não tendo relação com as outras atividades relacionadas ao desenvolvimento das aplicações, como, por exemplo, codificação e teste.

Nessa figura, as três camadas (SaaS, PaaS e IaaS) podem constituir um provedor único ou cada uma delas representar um provedor diferente. Os provedores de SaaS desenvolvem aplicações direcionadas a um conjunto de clientes. Apesar desse desenvolvimento envolver também a criação de aplicações usando serviços isolados, consideramos os casos que as aplicações constituem composições de serviços. Nos casos em que há separação entre as categorias de provedores, o desenvolvimento das composições de serviços geralmente é realizado tendo como base facilidades oferecidas por provedores de PaaS, que proveem não somente o ambiente de desenvolvimento do *software*, mas também um conjunto de serviços previamente desenvolvidos

e que podem ser integrados à composição. A composição de serviços precisa então ser implantada em um conjunto de VMs na camada de IaaS antes que elas possam atender as requisições dos clientes.

Para provedores de SaaS que interagem diretamente com provedores de IaaS, a implantação dos serviços deve ser realizada por eles mesmos. Por outro lado, se provedores de SaaS desenvolvem suas aplicações usando plataformas de provedores de PaaS, o provedor de PaaS é o responsável pela implantação dos serviços. Em ambos os casos, a arquitetura proposta pode ser utilizada para facilitar as atividades relacionadas ao gerenciamento de recursos. Dessa forma, ela constitui uma interface entre a camada de IaaS e as demais camadas, podendo o usuário neste cenário ser tanto provedores de SaaS quanto de PaaS. Nas duas situações, as restrições não-funcionais estabelecidas são guiadas pelas necessidades dos clientes que utilizam os serviços, havendo a preocupação de que essas restrições sejam atendidas de forma a reduzir o custo associado à utilização dos recursos.

Apesar das atividades automatizadas pela arquitetura terem como alvo a experiência oferecida aos clientes, eles não interagem diretamente com ela. Apenas o usuário (em nível SaaS ou PaaS) faz submissões a ela e tem influência sobre as decisões tomadas durante sua utilização.

Da maneira que foi proposta, a arquitetura pode ser implementada como um sistema dedicado, que é utilizado em contexto intrainstitucional, ou como uma plataforma compartilhada, que é utilizada, por exemplo, através de uma API de acesso remoto.

## 7.3 Implementação da arquitetura

Para demonstrar e validar as abordagens propostas, implementamos um protótipo de parte da arquitetura apresentada. A implementação foi realizada usando as linguagens Java 8 e XML, além de um conjunto de arcabouços e bibliotecas auxiliares. Foram empregados diversos princípios de engenharia de *software*, como por exemplo padrões de projeto, para tornar o código modular e de fácil extensão, visando sua reutilização em trabalhos futuros.

### 7.3.1 Implementação da síntese de coreografias

Como primeiro elemento na camada de síntese de coreografias implementamos um mecanismo para a representação dos grafos de processo e das restrições não-funcionais.

Em nossa implementação consideramos que dados sobre os serviços estão disponíveis em um catálogo implementado usando o padrão *Registry* [97]. Com isso, usando um identificador do serviço é possível obter os dados disponíveis para este serviço. De maneira semelhante, consideramos que existe um catálogo para cada um dos modelos de recurso gerados, cujo conteúdo também é acessado usando um identificador.

A especificação de restrições de QoS é realizada através da implementação das classes abstratas apresentadas no diagrama de classes da Figura 7.3. Consequentemente, para incluir uma restrição de QoS associada a uma coreografia de serviços, o usuário deve especificar uma implementação a ser usada para cada uma dessas classes.

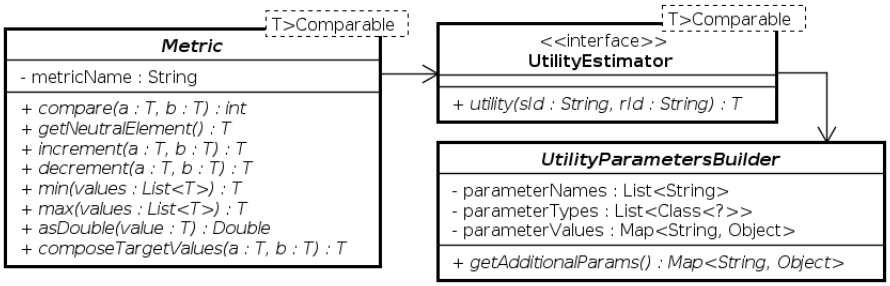



**Figura 7.3:** *Classes que definem uma restrição de QoS no arcabouço proposto.*

Como pode ser visto, a representação de métricas de QoS (definida pela classe `Metric`) segue a Definição 4.1 proposta anteriormente. Os valores da função de utilidade são obtidos pela implementação da classe `UtilityEstimator`, usando como parâmetros os identificadores do serviço e do recurso, assim como possíveis parâmetros adicionais providos pelo construtor de atributos da métrica equivalente, definido pela implementação da classe `UtilityParametersBuilder`.

Como o objetivo da implementação deste protótipo era demonstrar a abordagem proposta, assumimos que a especificação da coreografia é dada diretamente na notação do grafo de processo, ou seja, não implementamos um adaptador que traduz coreografias especificadas em BMPN 2.0 (ou outra linguagem). Dessa forma, a implementação do componente ***Adaptador para grafo de processo*** foi estabelecida como trabalho futuro.

Visando permitir a submissão de coreografias usando protocolos padrão da *Web*, a especificação de cada grafo de processo é feita usando um arquivo XML cujo esquema é apresentado no Apêndice B. Apesar desta decisão, por simplicidade, a interface do usuário foi disponibilizada na forma de uma biblioteca na linguagem Java.

A implementação de outras alternativas, mais adequadas ao cenário de uso da arquitetura, como o oferecimento de suas funcionalidades através de uma API

REST [91], é tida como trabalho futuro.

Na especificação de restrições de QoS, além de indicar a implementação das classes abstratas que permitem a estimativa dos valores de utilidade da métrica de QoS, o usuário deve especificar as operações que são restringidas pela métrica (ou omitir essa informação nos casos em que a restrição se aplica a toda a coreografia), o operador relacional considerado e o valor alvo da restrição.

Para distinguirmos restrições eliminatórias e classificatórias que devem ser aplicadas a todos os serviços que compõem a coreografia, usamos uma *tag* XML própria. Quando a restrição diz respeito a serviços específicos, o usuário deve especificar quais são esses serviços. A descrição dos demais componentes dessas restrições é feita usando a linguagem integrante do arcabouço descrito na Seção 4.1.3.

Para ilustrar a representação proposta, o Código 7.1 apresenta o arquivo XML de especificação da coreografia de rastreamento de frete e as restrições não-funcionais associadas, apresentadas na Seção 2.5. Por simplicidade, apresentamos apenas a representação dos serviços Gerenciador de Inventário (`GEREN_INVENT`) e Departamento de Vendas (`DEP_VENDAS`), uma vez que a especificação dos demais é semelhante. As anotações incluídas indicam os componentes principais da representação.

**Código 7.1:** *Especificação do grafo de processo e das restrições não-funcionais referentes à coreografia para rastreamento de frete.*

```
<?xml version="1.0" encoding="UTF-8"?>
<processGraph>
  <!-- Vértices do grafo -->
  <vertices>
    <!-- Vértice inicial -->
    <vertex id="START" type="START" />
    <vertex id="GEREN_INVENT" type="SERVICE">
      <!-- Descrição de um serviço -->
      <serviceDescriptor>
        <name>GEREN_INVENT</name>
        <dependencies>
          <dependency serviceSpecRole="consultarPedPen" />
          <dependency serviceSpecRole="consultarFrete" />
          <dependency serviceSpecRole="consultarFrete" />
        </dependencies>
        <logicalOperations>
          <!-- Descrição de uma operação -->
          <logicalOperation>
            <name>rastrearFrete</name>
            <numberOfInstructions>3</numberOfInstructions>
            <uri>/rastrearFrete/{id}</uri>
            <roles>rastrearFrete</roles>
          </logicalOperation>
        </logicalOperations>
        <serviceType>SOAP</serviceType>
        <roles>rastrearFrete</roles>
        <category>NEW_SERVICE</category>
        <packageURI>http://www.inf.ufg.br/~raphael/services/inventManager.jar</packageURI>
        <packageType>COMMAND_LINE</packageType>
        <endpointName>inventManager</endpointName>
        <port>1234</port>
        <version>0.1</version>
      </serviceDescriptor>
    </vertex>
    <vertex id="DEP_VENDAS" type="SERVICE">
```

```
36        <serviceDescriptor>
37          <name>DEP_VENDAS</name>
38          <logicalOperations>
39            <logicalOperation>
40              <name>consultarPedPen</name>
41              <numberOfInstructions>29</numberOfInstructions>
42              <uri>/consultarPedPen/{id}</uri>
43              <roles>consultarPedPen</roles>
44            </logicalOperation>
45          </logicalOperations>
46          <serviceType>SOAP</serviceType>
47          <roles>consultarPedPen</roles>
48          <category>NEW_SERVICE</category>
49          <packageURI>http://www.inf.ufg.br/~raphael/services/salesDep.jar</packageURI>
50          <packageType>COMMAND_LINE</packageType>
51          <endpointName>salesDep</endpointName>
52          <port>1234</port>
53          <version>0.1</version>
54        </serviceDescriptor>
55      </vertex>
56      ...
57      <!-- Conectores entre serviços -->
58      <vertex id="ORS1" type="ORS" />
59      <vertex id="ORJ1" type="ORJ" />
60      <!-- Vértice final -->
61      <vertex id="END" type="END" />
62    </vertices>
63    <!-- Arestas do grafo -->
64    <edges>
65      <!-- Requisição a uma operação -->
66      <edge vertexFrom="START" vertexTo="GEREN_INVENT" op="rastrearFrete" />
67      <edge vertexFrom="GEREN_INVENT" vertexTo="DEP_VENDAS" op="consultarPedPen" />
68      <edge vertexFrom="GEREN_INVENT" vertexTo="ORS1" />
69      <!-- Requisição a uma operação com probabilidade -->
70      <edge vertexFrom="ORS1" vertexTo="FRETE_TERR" op="consultarFrete">
71        <p>0.5</p>
72        <pConj id="12">0.4</pConj>
73      </edge>
74      <edge vertexFrom="ORS1" vertexTo="FRETE_MAR" op="consultarFrete">
75        <p>0.1</p>
76        <pConj id="12">0.4</pConj>
77      </edge>
78      <edge vertexFrom="FRETE_TERR" vertexTo="ORJ1" />
79      <edge vertexFrom="FRETE_MAR" vertexTo="ORJ1" />
80      <edge vertexFrom="ORJ1" vertexTo="END" />
81    </edges>
82    <!-- Carga esperada sobre a coreografia -->
83    <load>80</load>
84    <constraints>
85      <!-- Especificação de restrições de QoS -->
86      <qosConstraints>
87        <qosConstraint>
88          <targetOps>
89            <targetOp service="GEREN_INVENT" op="rastrearFrete" />
90            <targetOp service="DEP_VENDAS" op="consultarPedPen" />
91            <targetOp service="FRETE_TERR" op="consultarFrete" />
92            <targetOp service="FRETE_MAR" op="consultarFrete" />
93          </targetOps>
94          <metricClass>br.ufg.inf.utility.ThroughputMetric</metricClass>
95          <utilityEstimator>br.ufg.inf.utility.ThroughputEstimator</utilityEstimator>
96          <utilityParamBuilder>br.ufg.inf.utility.ThroughputParamBuilder</
                utilityParamBuilder>
97          <relationalOp>>=</relationalOp>
98          <targetValue>10.0</targetValue>
99        </qosConstraint>
100     </qosConstraints>
101     <!-- Especificação de restrições em relação a atributos do recurso -->
102     <resourceConstraints>
103       <generalConstraintsDescriptor>(MINIMUM_RESOURCE_USE = 0) & (ASCENDENT COST) & (
              DESCENDENT CLOUD_PROVIDER_REPUTATION)</generalConstraintsDescriptor>
104       <!-- Apenas 12 horas de indisponibilidade por ano -->
105       <serviceConstraintDescriptor service="GEREN_INVENT">(AVAILABILITY > 99.9987443)</
              serviceConstraintDescriptor>
```

```
106       </resourceConstraints>
107     </constraints>
108   </processGraph>
```

Como pode ser visto, a representação proposta inclui atributos relacionados à descrição da coreografia (como papéis e conectores) e aos serviços que a compõem.

A topologia da coreografia é especificada através dos vértices e das arestas do grafo. Cada vértice é definido por um tipo que estabelece os atributos que obrigatoriamente devem ser especificados. Os tipos possíveis são:

- `START`: define o vértice inicial do grafo;
- `END`: define um vértice final do grafo;
- `SERVICE`: define um vértice que representa um serviço da coreografia;
- `ANDS`: define um vértice que representa um conector *fork* de conjunção;
- `ANDJ`: define um vértice que representa um conector *join* de conjunção;
- `ORS`: define um vértice que representa um conector *fork* de disjunção;
- `ORJ`: define um vértice que representa um conector *join* de disjunção;
- `XORS`: define um vértice que representa um conector *fork* de disjunção mutuamente exclusiva; e
- `XORJ`: define um vértice que representa um conector *join* de disjunção mutuamente exclusiva.

Para cada serviço, há a descrição dos papéis que ele implementa assim como dos papéis dos quais ele depende. Esses papéis devem coincidir com operações providas e requisitadas pelos serviços que são conectados por arestas no grafo de processo. Por sua vez, para cada operação são descritas informações que podem ser usadas para estimar a demanda de recursos para o processamento desta operação. Conforme será discutido adiante, para as métricas de QoS que implementamos neste protótipo é suficiente restringir estas informações ao número médio de instruções usadas para processar cada requisição da operação.

Além dos atributos que são usados na estimativa e seleção de recursos, a representação dos serviços também considera outros atributos que são essenciais para a sua implantação. São eles:

- URI das operações;
- tipo de serviço: `SOAP` ou `REST`. Este atributo define o processo de invocação do serviço, que é usado para configurar as dependências entre eles;
- categoria do serviço: `NEW_SERVICE`, para serviços implantáveis, e `LEGACY_SERVICE`, para serviços legados;
- URI de acesso ao pacote de implementação do serviço;

- tipo do pacote de implementação do serviço: `COMMAND_LINE` ou `TOMCAT`. Este atributo define o processo de implantação e execução do serviço. Por simplicidade nos limitados a estes dois tipos.
  Serviços cujo tipo de pacote é definido como `COMMAND_LINE` devem ser associados a um pacote `JAR` que contém todas as dependências embutidas nelas. Por outro lado, serviços cujo tipo de pacote é definido como `TOMCAT` devem ser associados a um pacote `WAR`. Em ambos os casos, as dependências devem estar embutidas nos pacotes associados;
- *endpoint* do serviço;
- porta de execução do serviço; e
- versão do serviço.

Na definição das arestas, sempre que o vértice destino for um serviço, a operação utilizada deve ser especificada, sendo essa informação nula nos casos em que o vértice destino constitui um conector ou um vértice final. Quando conectores de disjunção são utilizados, a probabilidade de execução do subgrafo deve ser fornecida como um dos atributos da aresta.

As restrições não-funcionais são especificadas usando as definições e considerações apresentadas no Capítulo 4. No caso de restrições em relação a atributos do recurso, a linguagem utilizada é interpretada usando um analisador descendente recursivo, implementado como uma adaptação do analisador proposto por Malizia [171]. O código equivalente à restrição é então automaticamente gerado usando Javassist [58], uma biblioteca que permite manipular *bytecodes* Java.

O componente da arquitetura ***Gerador de grafo de dependências*** foi implementado usando o procedimento descrito no Capítulo 4. A geração do grafo de dependências é realizada usando a descrição das coreografias de entrada, especificadas isoladamente em cada arquivo XML. A representação gerada é mantida apenas internamente como objetos Java.

Além da métrica de vazão utilizada no código apresentado como exemplo, implementamos também a métrica latência, que consiste no tempo decorrido entre a chegada da requisição no serviço e seu processamento. Maiores detalhes sobre a implementação dessas métricas serão discutidos na próxima seção.

Quanto a restrições em relação a atributos do recurso, implementamos apenas a criação de restrições considerando um conjunto de atributos relacionados aos recursos. Estes atributos são listados a seguir:

- `MINIMUM_RESOURCE_USE`: indica a cobrança mínima tolerada. Nos casos em que provedores que estipulam cobrança mínima devem ser evitados, a restrição deve ser estabelecida indicando esse atributo como zero, conforme apresentado no exemplo.

- `LOCATION`: indica o país (juntamente com a região, opcionalmente) onde o recurso deve ser alocado.
- `COST`: indica o custo aceitável para alocação do recurso.
- `CLOUD_PROVIDER_REPUTATION`: um valor inteiro que indica a reputação do provedor. Quanto maior esse valor, maior a reputação do provedor. Em nossos experimentos, usando dados do último relatório da Gartner sobre IaaS [163], estabelecemos os seguintes valores para reputação dos provedores considerados: AWS = 4, Azure = 3, GCE = 2, Rackspace = 1.
- `CLOUD_PROVIDER`: indica qual provedor de nuvem deve ser usado na seleção de recursos. Neste caso, os demais provedores serão ignorados.
- `PUBLIC_CLOUD`: um valor lógico indicando se ambientes de nuvem pública devem ser usados.
- `AVAILABILITY`: Disponibilidade é a probabilidade do sistema estar em funcionamento. Apesar deste atributo ser considerado uma propriedade de QoS, em nossa abordagem o tratamos como uma restrição em relação a atributos do recurso. Essa decisão foi tomada porque em nosso protótipo ele é estimado usando unicamente propriedades dos tipos concretos de recurso que não tem relação com a capacidade e, portanto, não podem ser refletidas na representação dos tipos canônicos.
  Para estimar a disponibilidade, nos baseamos em dados analíticos de tempo e de inatividade de provedores de nuvem públicos fornecidos pela CloudHarmony [65], uma companhia que provê dados sobre desempenho de serviços em nuvem. A metodologia empregada pela CloudHarmony não permite uma análise detalhada das estatísticas sobre inatividade de provedores de nuvem. Como exemplo, CloudHarmony monitora uma região elegendo uma única zona de disponibilidade nesta região, de forma que interrupções que ocorrem em zonas de disponibilidade não monitoradas não serão registradas. Apesar dessa limitação, argumentamos que as informações providas são suficientes para ter uma visão geral sobre a disponibilidade de provedores de nuvem pública.

O protótipo desenvolvido é disponibilizado como uma biblioteca Java. Dessa forma, a interface com o usuário é realizada através de chamadas à API provida por essa biblioteca.

### 7.3.2 Implementação da síntese de recursos

O primeiro componente da camada de síntese de recursos – ***Estimador de utilidade*** – é responsável por estimar os valores da função de utilidade $\mathcal{U}$ para as res-

trições de QoS. Em nosso protótipo, implementamos esse componente considerando duas métricas de QoS: latência e vazão.

No caso de latência, a função de utilidade é calculada usando o tempo de processamento da requisição, $T_p$, que se refere à duração total do tempo que essa requisição permanece na fila para uso do recurso e em processamento. Utilizamos Teoria de Filas [158] para estimar este valor.

Com base em trabalhos anteriores [116, 257, 270], modelamos cada recurso como um centro de serviço com múltiplos servidores (modelo *M/M/c*). Ao fazer isso, assumimos que:

- existem $c$ servidores, o que corresponde à quantidade de núcleos da CPU no recurso;
- chegadas de requisições ocorrem a uma taxa $\lambda = \lambda_1 + \lambda_2 + \ldots + \lambda_{|\mathcal{O}_{s_i}|}$, de acordo com um processo de *Poisson*, em que $\lambda_j (1 \leq j \leq |\mathcal{O}_{s_i}|)$ é a taxa média de chegadas para uma certa operação no serviço sendo implantado. A carga sobre cada operação é representada no grafo de dependências, sendo estabelecida como uma propagação da carga estimada sobre a coreografia;
- o tempo de serviço tem uma distribuição exponencial com taxa média $\mu = \frac{1}{E[\text{tempo de serviço}]}$, onde $E[X]$ denota o valor esperado de $X$. Assumimos uma estratégia de escalonamento *round-robin* no processador. Dessa forma, $\mu$ pode ser calculado usando o tempo médio de serviço das operações utilizadas. O tempo de serviço para cada operação, por sua vez, é calculado em função do número de instruções de programa necessárias para implementar essa função e da capacidade do processador do recurso utilizado.
- o *buffer* possui capacidade infinita, de forma que não há limite no número de requisições que ele contém.

Tendo como base essas considerações, de acordo com Barbeau e Kranakis [26] $T_p$ é dado por:

$$T_p = \frac{1}{\mu} + \frac{C(c,\rho)}{c\mu - \lambda} \tag{7-1}$$

Em que $\rho = \frac{\lambda}{\mu}$ e $C(c,\rho)$ é a probabilidade de que uma requisição que chega seja forçada a esperar na fila (porque todos os servidores estão ocupados), dada pela fórmula de Erlang [253]:

$$C(c,\rho) = \frac{\left(\frac{(c\rho)^c}{c!}\right)\left(\frac{1}{1-\rho}\right)}{\sum_{k=0}^{c-1}\frac{(c\rho)^k}{k!} + \left(\frac{(c\rho)^c}{c!}\right)\left(\frac{1}{1-\rho}\right)} \tag{7-2}$$

A métrica vazão indica o número de requisições atendidas em um determinado período de tempo. Por simplicidade, assumimos que $\mathcal{U}$ pode ser estimada para esta métrica usando:

$$\frac{1}{T_p} \tag{7-3}$$

A Tabela 7.1 apresenta a definição completa das métricas de QoS implementadas, de acordo com as Definições 4.1 e 4.10, apresentadas no Capítulo 4.

| **Nome** | $\mathbb{D}$ | $\leq$ | $\oplus$ | $\bigwedge$ | $\bigvee$ | $\mathcal{U}$ | $\odot$ |
|---|---|---|---|---|---|---|---|
| Latência | $\mathbb{R}^+$ | $\leq$ | $+$ | min | max | $T_p$ | $\bigwedge$ |
| Vazão | $\mathbb{R}^+$ | $\geq$ | $+$ | max | min | $\frac{1}{T_p}$ | $+$ |

**Tabela 7.1:** *Definição das métricas de QoS implementadas.*

Com relação ao componente ***Mapeador de recursos***, conforme discutido no Capítulo 6, a estimativa de recursos é implementada usando a heurística WS-HEU [278]. Contudo, essa heurística só permite obter uma solução nos casos em que os tipos de recurso disponíveis possuem capacidade suficiente para satisfazer as restrições de QoS, dada a carga imposta sobre os serviços. Nos casos em que não é possível processar esta carga dentro dos limites estabelecidos pelas restrições, propomos como solução a criação de novas instâncias dos serviços.

Uma das maiores dificuldades em adicionar instâncias nesse caso é identificar quais serviços estão de fato sobrecarregados. A solução ideal seria estabelecer uma estratégia para determinar a capacidade de admissão da coreografia e, com base nisso, definir os serviços que precisam ter instâncias adicionadas para estender essa capacidade de forma a permitir cargas mais elevadas. Contudo, o desenvolvimento desta estratégia deve lidar com a definição do perfil de cada serviço isoladamente e da coreografia como um todo, de forma a determinar padrões de uso dos recursos. Estes padrões são influenciados por uma série de fatores que envolvem incerteza, como tipos de cliente e intervalo de tempo das requisições, o que torna o desenvolvimento da estratégia complexo. Em virtude disso, consideramos duas alternativas mais imediatas. A primeira, identificada como Estratégia I, consiste em identificar o principal gargalo no conjunto de serviços com relação à estimativa de recursos. Para tal, estabelecemos um indicador calculado como a carga multiplicada pela demanda de recursos para atender cada requisição. Uma nova instância é então criada para o serviço que apresenta maior valor para esse indicador. A segunda alternativa considerada, identificada como Estratégia II, consiste em utilizar os pares que apresentam a maior razão entre o valor do recurso e o peso calculado, sendo criadas novas instâncias para os serviços que constituem esses pares. Em ambas as estratégias, caso os serviços selecionados para

ter instâncias adicionadas sejam serviços legados, os serviços seguintes na sequência estabelecida são então considerados.

Enquanto na primeira alternativa apenas uma única instância do serviço é adicionada por vez, a segunda permite que um número maior de instâncias seja criado cada vez que insucesso for detectado, o que teoricamente faria com que sucesso na síntese seja obtido mais rapidamente, dado que as restrições afetadas são aquelas fim-a-fim. Para verificar se isso realmente ocorre, executamos a síntese de recursos com cargas variadas usando as coreografias do exemplo introduzido no Capítulo 2 e os recursos disponíveis nos provedores descritos no Capítulo 5.

De acordo com nossa análise, para os casos que ocorre insucesso na estimativa, o uso da Estratégia I (que adiciona instâncias apenas do serviço que constitui o gargalo) faz com que o resultado seja obtido muito mais rapidamente. Nas execuções realizadas, a Estratégia II (que considera os pares com pior razão entre valor e peso) fez com que a estimativa tenha que ser refeita, em média, 9 vezes a mais que a outra alternativa. Isso acontece porque os objetos que apresentam a pior razão entre valor e peso não necessariamente constituem o motivo das restrições não serem satisfeitas. Dessa forma, criar novas instâncias desses serviços não faz com que sucesso seja obtido.

Com isso, mesmo adicionando apenas uma única instância a cada vez que insucesso for detectado, a alternativa que usa o gargalo constitui uma melhor opção. Uma estratégia para reduzir ainda mais o tempo de processamento seria redimensionar o número de instâncias adicionais a cada vez que insucesso na estimativa fosse detectado novamente para uma mesma entrada. Como exemplo, poderia ser utilizada uma progressão que considera, a princípio, a adição de uma única instância; na segunda iteração, adicionam-se mais duas instâncias; na terceira iteração, mais quatro instâncias e assim sucessivamente. Contudo, como pode ser visto nos resultados do experimento, a correção da insucesso ocorre com re-execuções da estimativa de forma linear, fazendo com que essa estratégia leve à criação de instâncias desnecessárias.

Após a inclusão de instâncias e prováveis reexecuções da heurística WS-HEU, o resultado da estimativa de recursos é obtido como sendo o mapeamento de cada serviço na estrutura do grafo de dependências (e suas prováveis instâncias adicionais) para o tipo canônico selecionado. Este mapeamento é então usado como entrada para a seleção de recursos.

A seleção de recursos consiste em aplicar as restrições em relação a atributos do recurso estabelecidas. O conjunto dessas restrições é ordenado de acordo com as regras apresentadas na Seção 6.3.1. Para os casos de restrições da mesma categoria, a ordem é estabelecida aleatoriamente. Após a ordenação, filtros e classificadores, implementados de acordo com as restrições, são aplicados sobre os tipos concretos referentes a cada tipo canônico selecionado na primeira etapa da síntese. Esse pro-

cesso é realizado separadamente para cada serviço, levando em consideração apenas as restrições impostas sobre ele. As restrições são avaliadas na sequência estabelecida pela ordenação, sendo que o resultado da aplicação de uma restrição é usado como entrada na restrição seguinte. O processo finaliza quando todas as restrições sobre o serviço tiverem sido avaliadas ou quando o resultado de uma restrição for um conjunto vazio, o que caracteriza um caso de insucesso na seleção de recursos, devendo a estimativa de recursos ser refeita, conforme discutido no Capítulo 6. O tipo concreto para cada serviço é então parcialmente selecionado como sendo o primeiro da lista de tipos concretos remanescentes.

Caso sejam adicionadas instâncias de serviços na estimativa de recursos, elas são ignoradas na filtragem e classificação de recursos. Isso acontece porque as instâncias adicionais estão sujeitas às mesmas restrições, bastando considerar apenas o serviço original. Contudo, essas instâncias são consideradas na consolidação de recursos, uma vez que poderão permitir a consolidação de conjuntos diferentes de serviços. Isso é implementado inicialmente avaliando apenas a demanda do serviço original e o tipo canônico selecionado para este. Caso nenhum serviço possa ser consolidado nos tipos concretos disponíveis para o serviço original, as instâncias adicionais não serão avaliadas. Contudo, se ocorrer a consolidação, o processo se repete considerando a busca a partir de uma instância adicional até que não seja possível consolidar novos recursos.

Na implementação do algoritmo de consolidação de recursos, para estimar a distância entre dois recursos, necessária para calcular o fator de comunicação usado no algoritmo, nos baseamos na fórmula de Vincenty [259], que calcula a distância entre duas localizações geográficas especificadas por suas coordenadas. Essas coordenadas são obtidas através de consultas à API Google Maps Geocoding [115] usando como parâmetros o código ISO 3166-1 alpha-2 [134] do país e demais atributos do endereço (quando eles existem) da região onde os recursos serão implantados.

Os aspectos dinâmicos da abordagem, em parte coordenados pelo componente ***Adaptador de recursos***, não foram implementados no protótipo desenvolvido. Conforme será discutido no Capítulo 10, consideramos diversas mudanças possíveis que podem impactar a encenação das coreografias e, consequentemente, o gerenciamento de recursos, e propomos uma solução inicial para parte delas. Contudo, o tratamento de forma ampla das mudanças ocasionadas pela dinamicidade do cenário considerado foi mantido como trabalho futuro.

### 7.3.3 Implementação do *broker* de recursos

Na implementação do componente ***Descobridor de recursos*** da camada de *broker* de recursos, assumimos que os dados que descrevem os tipos concretos são for-

necidos em arquivos CSV específicos de cada provedor de nuvem. A descoberta automática de recursos é tratada como trabalho futuro. Ela pode ser implementada usando ferramentas como HtmlUnit [103] para consultas às páginas *Web* dos provedores, em que os atributos dos recursos são descritos.

Para a implementação do componente ***Alocador de recursos***, bem como dos adaptadores que interagem com provedores de nuvem (***CP-Adapter***) e dos *probes* de monitoramento nas VMs instanciadas (***VM-Probe***), realizamos uma adaptação do *Enactment Engine* (EE) [159, 161, 59], um dos componentes do projeto CHOReOS [60], desenvolvido pelo Instituto de Matemática e Estatística (IME) da Universidade de São Paulo (USP) em parceria com instituições europeias. A escolha de adaptar essa plataforma em vez de realizar a implementação por completo foi tomada visando evitar que tarefas puramente de implementação relacionadas à alocação de recursos tivessem que ser refeitas e assim, pudéssemos focar na principal contribuição da tese que é a síntese de recursos.

Conforme ilustrado na Figura 7.4, o CHOReOS EE permite a implantação de uma coreografia de serviços dada a sua descrição em alto nível. Esta plataforma não considera o uso de linguagens de modelagem de coreografias, sendo a especificação feita usando a API disponibilizada. A implantação da coreografia inclui a instalação e configuração de toda a pilha de *software* necessária para a execução do serviço, o que inclui o sistema operacional e um servidor *Web* no caso de serviços *Web*. Os números na figura indicam a ordem com que os componentes de *software* são instalados. A direção das setas vai do componente que coordena a instalação para o componente sendo instalado. A plataforma CHOReOS EE também realiza a configuração das dependências entre os serviços, de forma que a coreografia possa posteriormente ser encenada.

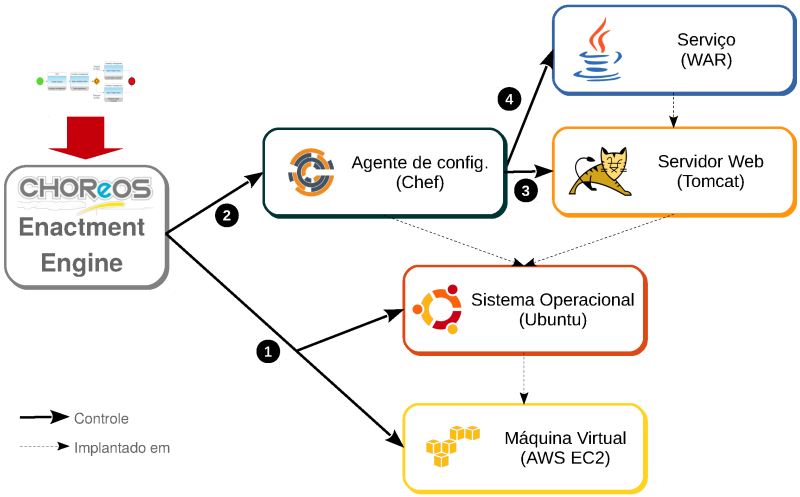



**Figura 7.4:** *Princípio geral do funcionamento do CHOReOS Enactment Engine.*

O funcionamento do CHOReOS EE se baseia no uso da ferramenta Chef [57],

que consiste em um agente responsável pela configuração de uma VM, com base em *scripts* gerados de acordo com as características do serviço sendo implantado. Embora o componente utilize implementações preestabelecidas para o *software* utilizado (Ubuntu [52] como sistema operacional e Tomcat [18] como servidor *Web*), a utilização de Chef como gerenciador de configuração facilita a eventual utilização de diferentes versões ou produtos alternativos, uma vez que ele abstrai as peculiaridades do *software* utilizado.

Apesar da seleção de recursos ter sido proposta no CHOReOS EE, seus desenvolvedores não implementaram essa funcionalidade, sendo que um único tipo preestabelecido é usado na implantação de todos os serviços. Dessa forma, grande parte da adaptação realizada por nós, na implementação deste componente, consistiu em incluir, no processo de implantação de coreografias realizado pelo CHOReOS EE, a síntese de recursos por nós desenvolvida. As demais modificações feitas na implementação original são listadas abaixo:

- Atualização das bibliotecas utilizadas para versões mais recentes, visando melhoria no desempenho.
- Expansão dos atributos usados na descrição de recurso, de forma a incluir os atributos que propusemos em nossa abordagem, entre eles, custo e localização.
- Inclusão do suporte a múltiplas regiões de um mesmo provedor, de forma que a localização do recurso possa ser levada em consideração na alocação.
- Inclusão de um adaptador para uma nuvem pública simulada. Este adaptador foi implementado usando CloudSim [47, 51], um *toolkit* extensível que permite a modelagem e simulação de sistemas de computação em nuvem e políticas de provisionamento de recursos nesse tipo de ambiente. Utilizamos esse adaptador para simular chamadas a um provedor real e assim, realizar testes do protótipo.
- Inclusão de um mecanismo de redirecionamento dos eventos gerados pelos *probes* para o componente analisador em nossa arquitetura.
- Implementação de adaptadores para as entidades de domínio comuns nas duas implementações, que dizem respeito à representação de serviços, de coreografias e de recursos.
- Correção de pequenos *bugs* no código.

Para tornar transparente a inclusão de nossa abordagem de síntese de recursos, permitindo seu uso em soluções baseadas no CHOReOS EE, não alteramos significativamente o processo de implantação de coreografias implementado por ele, sendo a síntese de recursos adicionada como uma etapa adicional nesse processo.

Em nosso protótipo, implementamos um adaptador da notação do grafo de dependências para a representação usada no CHOReOS EE. Dessa forma, do ponto de

vista desse componente, o conjunto de coreografias sendo implantadas consiste em uma única composição. Essa visão não tem impacto sobre o resultado esperado, uma vez que as dependências são configuradas da mesma forma.

Pelos mesmos motivos apresentados na seção anterior com relação aos aspectos dinâmicos da abordagem, a implementação do componente ***Analisador de eventos*** foi estabelecida como trabalho futuro.

## 7.4 Considerações finais

A principal limitação das propostas encontrados na literatura para implantação de coreografias é que elas geralmente não consideram todas as atividades relacionadas ao gerenciamento de recursos. Diante disso, neste capítulo apresentamos uma arquitetura para a execução coordenada da abordagem proposta para a modelagem de coreografias, restrições e recursos e para a descoberta, estimativa e seleção de recursos.

A arquitetura foi discutida com base em seu cenário de uso. Em seguida, apresentamos a implementação de um protótipo de parte dessa arquitetura. Conforme apresentado no próximo capítulo, o protótipo implementado foi usado para demonstrar a abordagem proposta e avaliá-la por meio de experimentos, através dos quais analisamos a efetividade, o tempo de processamento e a estabilidade da estratégia de síntese de recursos apresentada nesta tese.

A implementação desenvolvida incorporou apenas os componentes diretamente relacionados às atividades efetuadas em tempo de implantação das coreografias. A implementação dos demais componentes, que são relacionados aos aspectos dinâmicos, isto é, que ocorrem em tempo de encenação das coreografias, foram mantidos como trabalho futuro. Contudo, é importante ressaltar que os componentes implementados são suficientes para validar e avaliar a proposta apresentada nesta tese, uma vez que o principal problema nela discutido diz respeito à satisfação de restrições não-funcionais nas atividades relacionadas à implantação dos serviços.

# Parte IV

# Validação e conclusão

CAPÍTULO 8

# Avaliação experimental

No Capítulo 6, descrevemos a abordagem proposta para a síntese de recursos para um conjunto de coreografias de serviços sujeitas a restrições não-funcionais. Essa abordagem de síntese foi incluída em uma arquitetura que considera as atividades de gerenciamento de recursos durante a implantação e encenação das coreografias. O funcionamento da síntese de recursos se baseia nos tratamentos sugeridos para a representação de recursos e para a modelagem conjunta de coreografias, e constitui a principal contribuição desta tese. Diante disso, com o objetivo de avaliar os resultados obtidos pela abordagem desenvolvida e comprovar a factibilidade de seu uso realizamos uma série de experimentos que são descritos neste capítulo.

Com relação às atividades realizadas durante a implantação das coreografias, a maior preocupação é garantir que todas as restrições não-funcionais especificadas são atendidas, ao mesmo tempo que a estimativa e seleção de recursos são realizadas de forma que a alocação resultante seja eficiente, reduzindo os custos associados à utilização dos recursos e à comunicação entre os serviços. Assim sendo, inicialmente avaliamos a efetividade da síntese de recursos com relação a esses aspectos. Nesses experimentos buscamos verificar o impacto do uso da abordagem proposta, considerando diferentes cenários de utilização. De maneira mais precisa, realizamos dois experimentos visando avaliar a efetividade da síntese, nos quais inicialmente avaliamos a redução no atraso de comunicação entre os serviços, considerando o impacto de duas técnicas utilizadas na síntese de recursos. Em seguida, avaliamos a redução no custo dos recursos selecionados considerando variações no modelo canônico de recursos.

De maneira complementar, é preciso garantir que os resultados da síntese de recursos sejam obtidos rapidamente, não causando sobrecarga excessiva sobre o processo de implantação dos serviços. Associado a isso, é fundamental que as decisões tomadas não gerem a necessidade de mudanças significativas na alocação de recursos para serviços previamente implantados. Com base nesses critérios, realizamos experimentos para avaliar o tempo de processamento da síntese de recursos e para medir a estabilidade nos resultados oferecidos. Neste contexto, a estabilidade se refere a quão significativas são as adaptações, necessárias de maneira global, ao considerar mudan-

ças pontuais nos serviços previamente implantados.

Uma vez que o objetivo ao realizar esses experimentos não foi avaliar o processo de implantação das coreografias como um todo, mas apenas a abordagem de síntese propriamente dita, nenhum experimento foi realizado diretamente nos provedores de nuvem considerados. Apenas foram utilizadas informações sobre os recursos disponibilizados nestes provedores para a criação do modelo de recursos utilizado na síntese. Essa estratégia foi adotada como forma de eliminar gastos associados à execução dos experimentos, dada a sua finalidade.

Os experimentos foram projetados e descritos seguindo a estrutura proposta por Wohlin *et al.* [265] para a apresentação de experimentos: definição do experimento, planejamento do experimento, operação do experimento e análise e interpretação dos resultados.

## 8.1 Efetividade da síntese de recursos

A efetividade da síntese de recursos está relacionada ao custo de utilização dos recursos. De maneira mais precisa, buscamos avaliar se a abordagem desenvolvida é realmente eficiente, de forma que os resultados obtidos representam redução no custo de utilização dos recursos selecionados e no atraso de comunicação entre os serviços.

Para avaliar a efetividade da síntese de recursos, consideramos dois experimentos, nos quais avaliamos o atraso de comunicação entre os serviços (Experimento I), considerando os recursos selecionados e o custo de utilização desses recursos (Experimento II). A seguir discutimos os experimentos realizados.

### 8.1.1 Definição dos experimentos

O objeto de estudo do Experimento I é o impacto causado pela abordagem de síntese proposta sobre o atraso de comunicação entre os serviços a serem implantados.

O atraso de comunicação refere-se à sobrecarga imposta na encenação das coreografias pela troca de mensagens entre os serviços. Esta sobrecarga é medida principalmente em função da quantidade de mensagens trocadas e pelo tempo entre o envio e o recebimento destas mensagens. Apesar do gerenciamento de recursos não ter impacto sobre a quantidade de mensagens trocadas entre os serviços, uma vez que este aspecto é uma característica inerente da lógica do modelo de negócio representado pela coreografia, ele pode influenciar o atraso de comunicação. Isso se dá porque a seleção de recursos pode fazer com que a sobrecarga de tempo da comunicação seja reduzida, ao implantar serviços em um mesmo recurso ou em recursos com localidade

próxima. Esta é a razão pela qual consideramos o atraso de comunicação como uma métrica de efetividade da síntese de recursos.

A redução no atraso de comunicação entre os serviços é obtida na abordagem de síntese proposta através de duas técnicas: consolidação de recursos e troca de tipos usando como critério a localidade.

A consolidação de recursos consiste em alocar múltiplos serviços compartilhando o mesmo recurso. Essa técnica promove a redução no atraso de comunicação porque o principal critério utilizado para decidir quais serviços devem ser alocados em um mesmo recurso é a demanda de comunicação entre eles.

A troca de tipos usando como critério a localidade é uma técnica auxiliar utilizada durante a busca por consolidação de recursos, que consiste em analisar a redução no atraso de comunicação proporcionado pela seleção de um tipo concreto de recurso com melhor localidade em detrimento de outro previamente selecionado. Essa troca de tipos considera um custo (financeiro) adicional tolerável para a utilização dos recursos. Este custo adicional é configurado pelo usuário como um percentual do custo original do recurso.

O propósito de realizar este experimento é avaliar as vantagens e desvantagens da aplicação destas técnicas. Para tal, avaliamos também a sobrecarga causada pelo seu uso, no tempo de processamento da síntese. Uma vez que a consolidação de recursos tem impacto sobre o custo de utilização dos recursos selecionados, também avaliamos este aspecto nesse experimento.

De maneira complementar, o Experimento II tem como escopo o custo financeiro de utilização dos recursos selecionados, uma vez que um dos principais objetivos da abordagem de síntese desenvolvida é reduzir este custo. Esta redução é obtida pela seleção de uma quantidade menor de instâncias de tipos concretos de recurso (obtida pela consolidação de recursos) e, evidentemente, pela seleção de tipos concretos que possuem menor custo de utilização. Desta forma, o propósito da realização deste experimento é avaliar como o uso de diferentes modelos de recurso como entrada na síntese interfere neste aspecto.

Obter resultados que reduzem o custo de utilização dos recursos, ao mesmo tempo em que satisfazem todas as restrições não-funcionais estabelecidas é particularmente importante para garantir que os serviços implantados sejam oferecidos sem uma alta tarifação associada e, assim, lidar com a competitividade crescente em soluções baseadas em nuvem. Diante disso, na abordagem proposta o custo de utilização dos recursos tem grande impacto sobre os resultados por ser usado como critério de decisão na estimativa e seleção de recursos.

### 8.1.2 Planejamento dos experimentos

A principal variável utilizada para medir a efetividade com relação aos resultados obtidos pela síntese é o **custo de utilização dos recursos selecionados**. Usamos como unidade de medida o custo em dólar ($) por uma hora de utilização do recurso. Esse custo ($c_v$ em nossa notação) é atribuído de acordo com a tarifa estabelecida pelo provedor de nuvem, no caso de ser um tipo concreto selecionado em uma nuvem pública, ou como uma proporção do custo para manter em execução a máquina física, no caso do tipo concreto ser instanciado em uma nuvem privada. Neste último caso, o custo é calculado de acordo com a tarifa de energia elétrica usando o consumo da máquina física (medido em *watt*) e o custo de referência de $0.14 por Kilowatt-hora (KWH)[1]. Para cada resultado da síntese de recursos, realizamos a soma do custo de todos os tipos concretos selecionados. Apesar dessa variável estar relacionada ao Experimento II, também a analisamos no Experimento I, uma vez que a consolidação de recursos tem influência sobre o custo de utilização dos recursos.

Para quantificar a redução do ***atraso de comunicação*** no Experimento I, consideramos o uso de HTTP e TCP/IP na comunicação entre os serviços. Por simplicidade, ignoramos informações como a topologia da rede e a quantidade de *hops* usados para transmitir dados entre dois serviços. Pelo mesmo motivo, em nossa análise desconsideramos o tempo de processamento na rede, assim como o tempo de espera. Essas simplificações não têm impacto significativo na solução e constitui apenas uma forma de operacionalizar o experimento.

Em nosso experimento, não é correto assumir que a largura de banda disponível para o recurso é completamente utilizada. Isso é especialmente verdade no cenário com múltiplos provedores de nuvem, onde os recursos são alocados em regiões geográficas que podem ser distantes umas das outras. Nesse caso, o funcionamento do protocolo TCP faz com que a distância tenha um impacto relevante no atraso de comunicação [20]. A principal razão é que a largura de banda realmente obtida passa a ser limitada a um fator calculado de acordo com o tamanho da janela de transmissão do TCP (*TCP Window*) e do tempo de propagação dos dados.

Para estabelecer um resultado de referência a ser usado na análise do atraso de comunicação, além do cenário esperado no escopo do problema (ambiente multi-nuvem), a síntese de recursos também foi realizada considerando apenas recursos disponibilizados por um único provedor em uma região específica.

Analisamos também em ambos os experimentos a relação entre a quantidade de instâncias de serviço e a quantidade de instâncias de recurso alocadas para hospedar os serviços. Essa variável é particularmente importante nos cenários em que é

[1] Valor em dólar do custo por KWH em Goiânia - GO, Brasil; em agosto de 2016.

possível realizar consolidação de recursos, pois ela indica a quantidade de recursos que foram consolidados. Essa relação é definida pelo ***percentual de instâncias de VM por instâncias de serviço***, calculada de acordo com a equação a seguir:

$$\text{Percentual de instâncias de VM} = \frac{\text{Quantidade de instâncias de VM}}{\text{Quantidade de instâncias de serviço}} \times 100. \quad (8\text{-}1)$$

Outra variável analisada em ambos os experimentos é o **tempo de processamento da síntese**, que consiste no tempo necessário para obter o resultado em cada iteração. Ele é medido em segundos (*s*).

Os experimentos foram realizados usando como entrada as coreografias de serviço e as restrições não-funcionais apresentadas no exemplo de cenário discutido no Capítulo 2, além dos tipos concretos considerados no exemplo na Seção 5.3. Dessa forma, a escala considerada foi de 8 serviços que deveriam ser mapeados para 827 tipos concretos de recurso. A escolha dessa entrada foi baseada em amostragem por quotas (*quota sampling*), em que o objetivo era utilizar uma entrada que contivesse todos os atributos considerados no escopo do problema. Mais especificamente, foi selecionada uma entrada que utilizasse diferentes tipos de conectores na topologia da coreografia, que consistisse em múltiplas coreografias com compartilhamento de serviços entre elas, e que apresentasse diferentes categorias de restrições não-funcionais.

Para ser possível a implantação de serviços que possuem como restrição a alocação de recursos em um ambiente privado, simulamos o uso de uma nuvem deste tipo. A nuvem privada simulada contém uma única máquina física com a seguinte configuração: servidor IBM® x3550 (2 × [Xeon® X5670 2933 MHz, 6 núcleos], 12GB). Esta configuração de hardware foi estabelecida visando tornar possível a implantação dos serviços para os quais a restrição existe, dada a demanda de recursos esperada.

No Experimento I, para ser possível analisar o impacto das duas técnicas propostas para redução no atraso de comunicação, realizamos a síntese de recursos para a entrada considerada levando em consideração os seguintes modos de utilização:

- **CON+LOC**: a síntese é realizada considerando recursos de múltiplos provedores em múltiplas regiões. Ambas as técnicas (consolidação de recursos e troca de tipos usando como critério a localidade) são utilizadas.
- **CON**: a síntese é realizada considerando recursos de múltiplos provedores em múltiplas regiões. Apenas a técnica de consolidação de recursos é utilizada.
- **LOC**: a síntese é realizada considerando recursos de múltiplos provedores em múltiplas regiões. Apenas a técnica de troca de tipos usando como critério a localidade é utilizada.

- **NEN**: a síntese é realizada considerando recursos de múltiplos provedores em múltiplas regiões. Nenhuma das técnicas propostas é utilizada.
- **Azure-Brasil**: a síntese é realizada considerando apenas recursos disponibilizados pelo provedor Microsoft Azure na região localizada no Brasil. Ambas as técnicas propostas são utilizadas.

O último modo é usado para obter resultados de referência para serem usados na análise dos outros modos, uma vez que o atraso de comunicação nestes resultados será o menor possível, visto que todos os recursos são alocados em uma mesma localidade. Adotamos o Microsoft Azure como provedor por ele possuir uma quantidade maior de tipos concretos. A escolha do Brasil como região é visando satisfazer todas as restrições não-funcionais, e por ser um dos países que contém uma única região do provedor considerado. Para que todos os recursos fossem alocados na mesma localidade, simulamos a localização da nuvem privada como sendo na mesma região geográfica, e consideramos que o tempo de transmissão entre a nuvem privada e a nuvem pública nesse caso é nulo.

Como forma de isolar a redução que é possível obter no atraso de comunicação sem que seja necessário aumento no custo dos recursos selecionados, não foi considerado custo financeiro adicional aceitável no processamento dessas técnicas. Dessa forma, o custo de utilização dos recursos selecionados com o uso das técnicas é igual ou inferior ao custo que seria obtido sem a utilização delas. Ao fazer essa afirmação, assumimos que o resultado da estimativa de recursos é idêntico para os dois resultados (com e sem uso das técnicas). Conforme será discutido adiante, a estratégia adotada na estimativa de recursos pode gerar resultados diferentes para uma mesma entrada, o que, consequentemente, levará a custos diferentes.

Com relação ao Experimento II, a efetividade da síntese de recursos é uma consequência direta dos tipos disponíveis no modelo canônico de recursos, uma vez que a distribuição das contribuições de cada serviço em restrições de QoS fim-a-fim é resultado do mapeamento realizado entre os serviços e os tipos considerados neste modelo. Dessa forma, o uso de um conjunto incipiente de tipos canônicos pode fazer com que o custo dos recursos selecionados seja elevado. Isso acontece nos casos que a real capacidade de recurso que um determinado serviço precisa não é representada por nenhum tipo canônico, fazendo com que o mapeamento seja realizado para tipos com maior capacidade que a necessária.

Diante disso, neste experimento avaliamos os resultados obtidos com a síntese de recursos com relação ao custo de utilização dos recursos selecionados, considerando diferentes modelos de recurso como entrada. Mais precisamente, variamos a quantidade de tipos canônicos usados na estimativa de recursos em 6, 12, 24, 48 e 96 tipos. O primeiro valor refere-se à quantidade estabelecida pelo algoritmo $x$-means,

que corresponde ao valor adotado nos demais experimentos. Na entrada considerada para o experimento existem 133 tipos concretos (considerando o segundo nível da representação proposta). Com isso, devido à intersecção entre esses tipos, ao utilizar 96 tipos garantimos que, na prática, todos os tipos concretos são usados na estimativa de recursos.

Em ambos os experimentos, realizamos a síntese de recursos para a entrada considerada, variando a carga sobre as coreografias de 10 a 100.000 requisições por segundo (req/s), com incrementos de 10 requisições em cada iteração do experimento. Para tal, assim como nos demais experimentos apresentados nesse capítulo, o número de instruções em cada operação provida pelos serviços é definido de maneira sintética. Para gerar esses valores realizamos uma projeção sobre o número de instruções tipicamente encontrado em métodos Java. Para tal, nos baseamos na análise fornecida por Collberg *et al* [66]. O tempo de serviço referente ao processamento de uma requisição é então estimado usando o número de instruções da operação requisitada e padronizando que a cada iteração do *clock* do processador 4 instruções são processadas. Esse número foi estabelecido por ser a quantidade ideal de instruções por ciclo [138].

### 8.1.3 Operação dos experimentos

As execuções da síntese em cada cenário dos experimentos foram realizadas em uma máquina com a seguinte configuração: CPU Intel® Core™ i5 2.67GHz, 4GB RAM, Ubuntu 14.04.

Para permitir a execução dos experimentos, foi implementado um programa auxiliar que gera a estrutura do grafo de dependências a partir da representação dos grafos de processo. Adicionalmente, o código que implementa a seleção de recursos foi instrumentado para permitir habilitar ou desabilitar o uso das técnicas sendo avaliadas. Também criamos *scripts* responsáveis por realizar chamadas à síntese de recursos em cada um dos cenários estabelecidos.

O grafo de dependências foi gerado a cada iteração de forma a garantir a configuração da carga sobre as coreografias.

Para permitir maior confiabilidade, cada iteração em ambos os experimentos foi repetida 30 vezes e os resultados foram gerados com intervalo de confiança de 95%. Como os erros foram desprezíveis, eles não foram considerados na análise e interpretação dos resultados.

### 8.1.4 Análise e interpretação dos resultados

**Experimento I**

De maneira geral, para cargas maiores o resultado foi semelhante entre as iterações. Em virtude disso, para facilitar a visualização dividimos os resultados para cada variável em quatro gráficos, considerando faixas da carga submetida, com diferentes escalas.

Uma vez que as demais variáveis são consequência direta da quantidade de instâncias criadas, primeiramente analisamos os resultados considerando esta variável. A Figura 8.1 apresenta os resultados, considerando a quantidade de instâncias de VM como o percentual da quantidade de instâncias de serviço, em função da carga sobre as coreografias. Quanto menor o valor deste percentual, melhor o resultado, pois significa que mais recursos foram consolidados.

Nesta figura (e nas demais que apresentam os resultados) os erros obtidos pelo intervalo de confiança não foram plotados para torná-la mais legível, uma vez que estes erros foram desprezíveis.

Para interpretar esses resultados é importante ressaltar que a quantidade de instâncias de VM será no máximo igual à quantidade de instâncias de serviço, podendo ser menor quando se utiliza a técnica de consolidação de recursos.

Como pode ser visto na figura, a consolidação de recursos acontece com mais frequência para cargas menores (até 2.000 req/s no cenário considerado).

O comportamento sobre consolidação para cargas menores acontece porque com o aumento da carga é preciso que recursos com maior capacidade sejam selecionados para satisfazer as restrições de QoS. Com isso, a tendência é que tipos mais robustos sejam preferidos na estimativa de recursos. Contudo, esses tipos têm menor capacidade excedente, em virtude dos tipos concretos consideramos no experimento. Em particular, no caso do tipo mais robusto do modelo, a capacidade excedente sempre será inexistente devido ao procedimento utilizado para gerar o modelo canônico de recursos.

Nos cenários considerados neste experimento, a consolidação de recursos chega a quase 40% da quantidade total de recursos quando se considera o uso de ambas as técnicas. Isso pode ser visto principalmente para cargas de até 100 req/s (gráfico no canto superior esquerdo). Além disso, consolidação ocorre em uma taxa menor quando não se considera a troca de tipos usando como critério a localidade. Isso se deve ao fato de algumas buscas por consolidação não serem realizadas se ambas as técnicas não forem utilizadas, o que faz com que o uso da técnica de consolidação isolada consiga consolidar recursos neste experimento apenas para cargas de até 100 req/s.

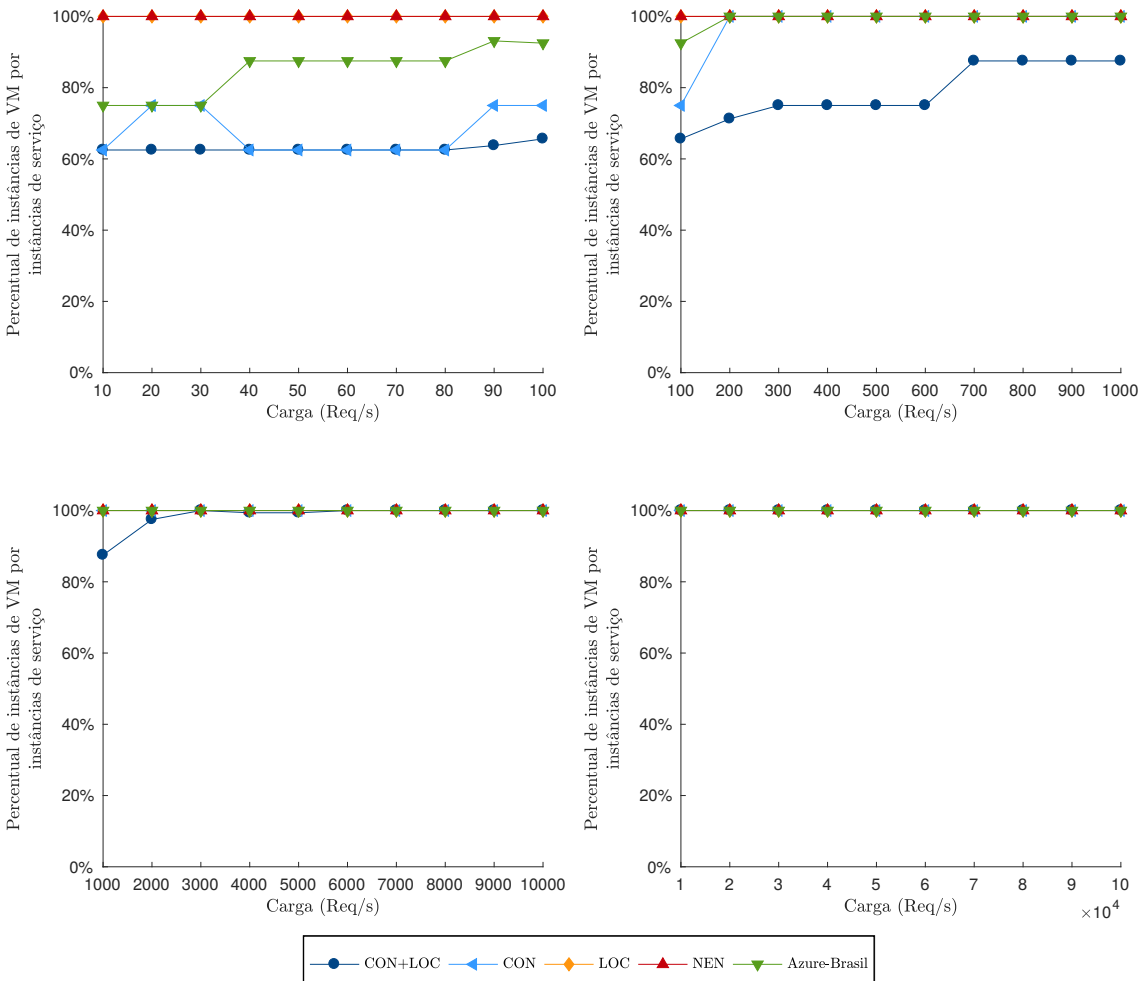



**Figura 8.1:** *Percentual de instâncias de VM por instâncias de serviço, de acordo com a carga em cada modos de utilização da síntese para o Experimento I.*

Para cargas mais elevadas (2.000 req/s no experimento) não é possível mais realizar a consolidação de recursos, significando que cada serviço é implantado isoladamente em um recurso. A razão deste comportamento é que nestes casos a demanda sobre todos os serviços é muito alta, não permitindo que os recursos sejam compartilhados.

Conforme esperado, considerar mais tipos concretos de recurso, ou seja, diferentes regiões de múltiplos provedores, favorece a consolidação de recursos, uma vez que o espaço de busca passa a ser maior. Isso justifica resultados piores quando se considera apenas recursos do Azure no Brasil.

Esses resultados evidenciam a vantagem de se considerar o uso de ambas as técnicas, pois a quantidade de instâncias de recurso alocadas é refletida no custo de utilização dos recursos. Esta afirmação pode ser comprovada nos resultados apresentados na Figura 8.2, que mostra o custo dos recursos selecionados em cada um dos cenários considerados no experimento.

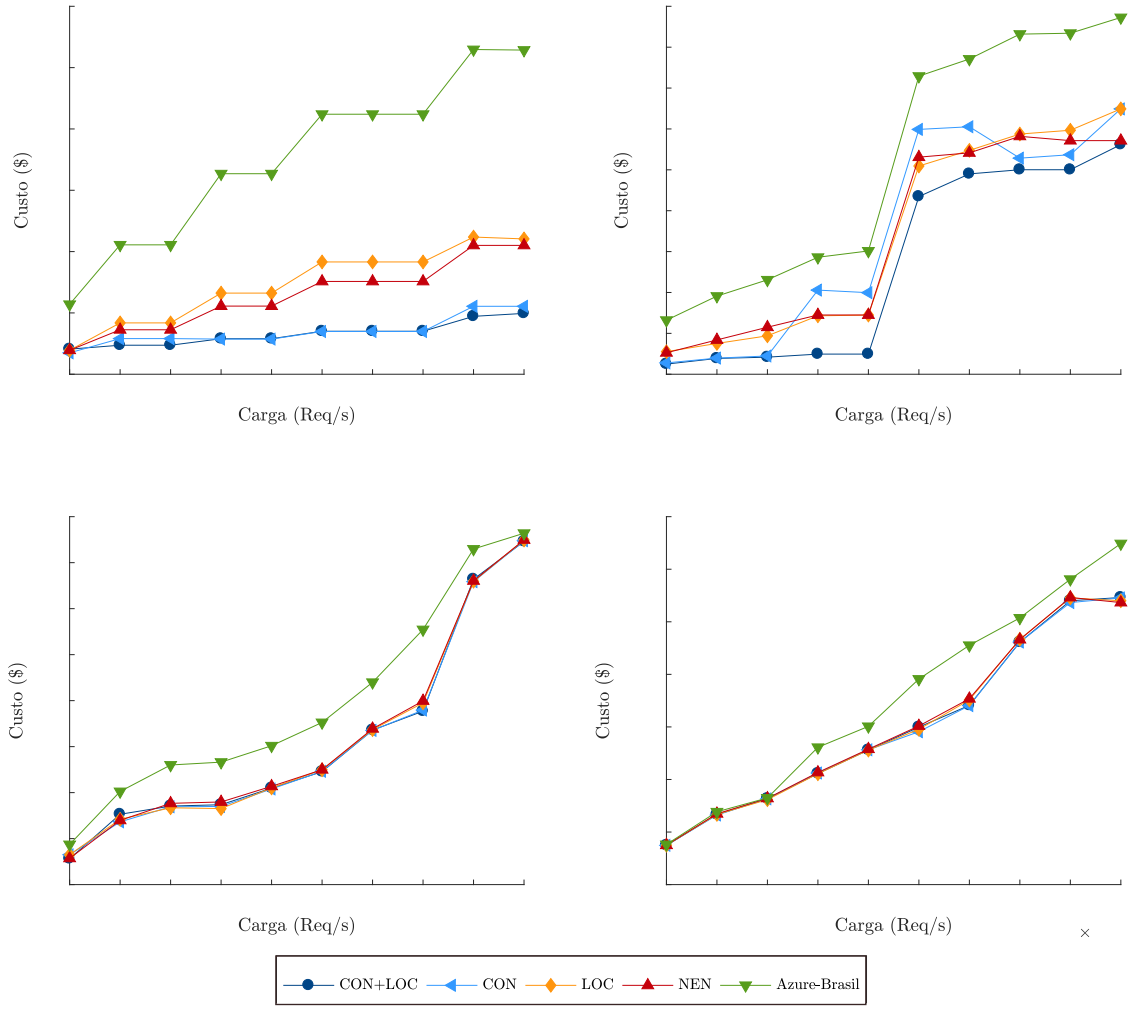



**Figura 8.2:** *Custo dos recursos selecionados de acordo com a carga em cada modo de utilização da síntese.*

Por se basear em um conjunto bem mais limitado de opções, o custo dos recursos selecionados ao considerar apenas um único provedor é mais elevado. Isso não é uma regra, uma vez que o único provedor considerado pode ter os recursos com menor custo. Contudo, para a entrada considerada, esse modo de utilização sempre apresentou o maior custo. Dessa forma, a adoção de um único provedor ou região só deve acontecer nos casos em que isso é uma restrição associada a todos os serviços sendo implantados.

Uma vez que a consolidação de recursos faz com que uma quantidade menor de instâncias de recurso sejam alocadas, o custo nos casos que isso ocorre é menor.

Nos casos em que não ocorreu consolidação de recursos, o custo é semelhante quando se utiliza múltiplos provedores (independente do uso ou não das técnicas avaliadas). Isso se deve ao fato de não terem sido considerados custos adicionais admissíveis na troca de tipos. A diferença no resultado em alguns casos, quando se utiliza múltiplos provedores, acontece em virtude da variação na estimativa de recursos, pois

a heurística utilizada garante somente o quão próximo da solução ótima o resultado está, sem necessariamente garantir que este resultado será o mesmo sempre para uma determinada entrada.

Tendo como base esses resultados, podemos analisar a redução no atraso de comunicação proporcionada pelo uso das técnicas propostas. A Figura 8.3 apresenta as medidas estimadas para esta variável, de acordo com a carga. Uma vez que consideramos também o tempo de transmissão das mensagens, o atraso varia com a carga.

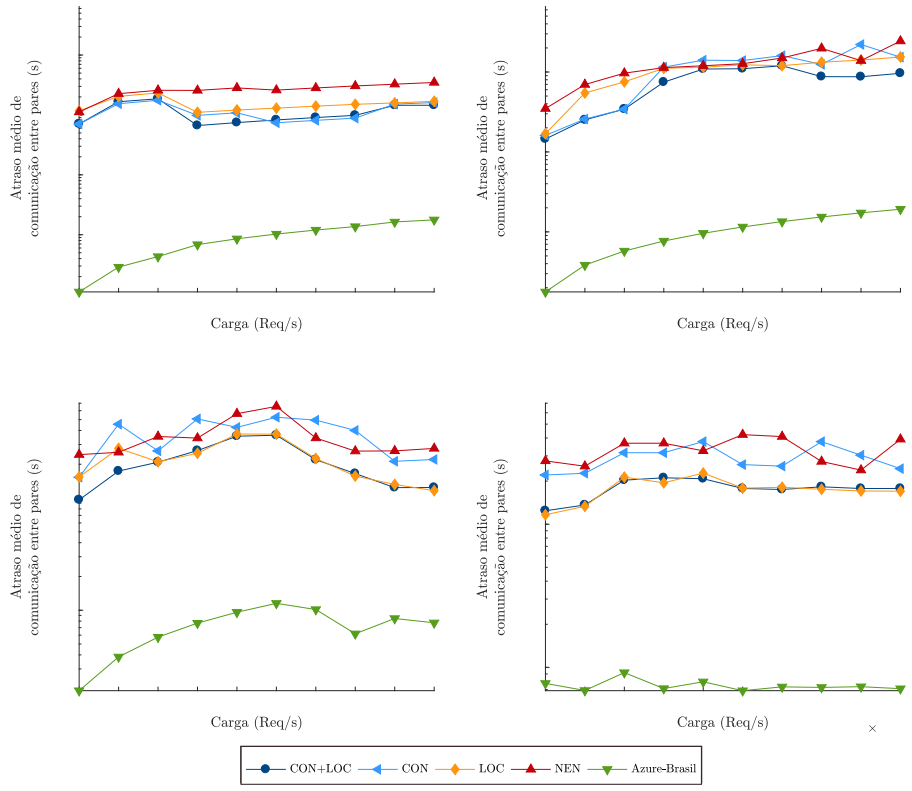
Por considerar uma única localidade, quando há um único provedor o atraso de comunicação é próximo de zero. Devido a nossas considerações, este constitui o melhor resultado possível na alocação de recursos, sendo então usado como um referencial na avaliação dos resultados para os outros casos.

Ao considerar o atraso médio de comunicação entre pares de serviços (Figura 8.3(a)) a consolidação de recursos aliada à troca usando como critério a localidade obtém melhores resultados, sendo que o uso isolado de consolidação reduz mais o atraso médio de comunicação do que o uso isolado da troca usando como critério a localidade. Isso era esperado, uma vez que ao consolidar recursos, o tempo de propagação das mensagens trocadas entre os serviços para eles mapeados passa a ser zero.

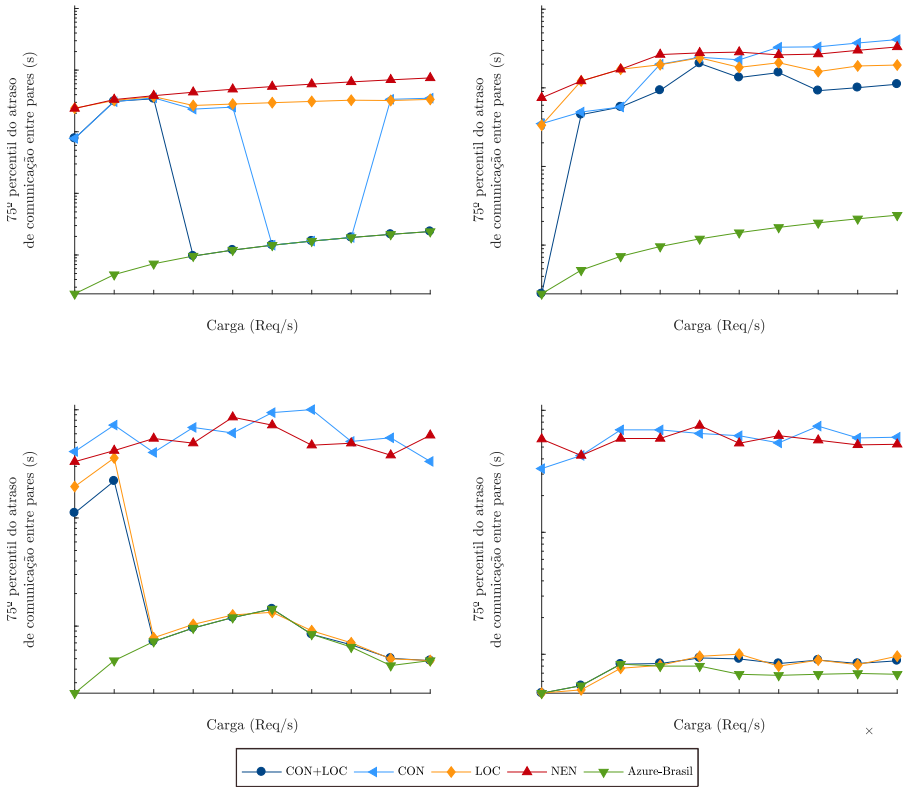
Ao analisar cargas mais elevadas (a partir de 3.000 req/s no experimento), os valores quando se utiliza a técnica de troca usando como critério a localidade são semelhantes, visto que a consolidação de recursos não é mais possível. Nesses casos o resultado depende do mapeamento feito na estimativa de recursos.

Apesar da redução no atraso de comunicação ser aparentemente não significativo ao considerar a média, a maioria dos valores de atraso é bem baixo quando se utiliza as técnicas propostas. Isso pode ser visto na Figura 8.3(b), que apresenta o 75º percentil para os valores dessa variável.

Como pode ser visto, na maioria dos casos em que as técnicas são utilizadas, a redução é relevante, sendo semelhante aos valores usados como referência. Como é possível destacar, o uso da técnica de troca usando como critério a localidade faz com que esse comportamento aconteça para quase todas as cargas consideradas. No experimento, as únicas exceções são para cargas entre 200 e 1.000 req/s, sendo que os valores altos de atraso de comunicação para este intervalo ocorrem devido à distância geográfica entre os tipos concretos selecionados para os serviços quando essa carga é esperada.




(a)




(b)

**Figura 8.3:** *Atraso de comunicação entre pares de serviço de acordo com a carga em cada modo de utilização da síntese: (a) média. (b) 75º percentil.*

Esses resultados comprovam a hipótese de que o uso das técnicas propostas reduz o atraso de comunicação entre os serviços. Contudo, eles adicionam uma sobrecarga no processamento da síntese de recursos. A Tabela 8.1 apresenta o tempo médio necessário para obter o resultado da síntese em cada um dos modos de utilização considerados. É importante destacar que o tempo de síntese não é influenciado pela carga.

| **Modo de utilização** | **Tempo (s)** |
|---|---|
| CON+LOC | 0,094 |
| CON | 0,073 |
| LOC | 0,038 |
| NEN | 0,026 |
| Azure-Brasil | 0,019 |

**Tabela 8.1:** *Tempo de processamento da síntese em cada modo de utilização considerado.*

Apesar do modo de utilização com apenas um provedor utilizar ambas as técnicas, o tempo de síntese é menor que os demais em virtude da quantidade bem menor de tipos concretos que precisam ser analisados. Quando se utiliza consolidação de recursos, o tempo de síntese é maior porque buscas sucessivas são realizadas sobre o grafo de dependências. Em contrapartida, a troca usando como critério a localidade adiciona uma pequena sobrecarga pela análise da localidade dos tipos concretos.

A análise do tempo de síntese serve apenas como um indicador inicial da sobrecarga causada pelas técnicas. Contudo, esse experimento não pode ser usado para prover uma análise mais conclusiva sobre esse aspecto, dado que outras entradas devem ser consideradas para avaliá-lo, como por exemplo o aumento no custo adicional admissível.

### Experimento II

Da mesma forma que no experimento anterior, iniciamos nossa análise sobre o número de instâncias. A Figura 8.4 apresenta a quantidade de instâncias de VM criadas como o percentual da quantidade de instâncias de serviço, de acordo com o aumento na carga. Não apresentamos os resultados com 96 tipos canônicos de recurso porque eles foram idênticos aos resultados com 48 tipos.

O mesmo comportamento comprovado no experimento anterior se repete, ou seja, quanto maior a carga sobre os serviços menor a possibilidade de consolidação de recursos. Em complemento a esse resultado, o experimento comprova outro fator que dificulta a possibilidade de consolidação de recursos: um número elevado de tipos canônicos. Como pode ser visto na figura, usando 48 (ou 96) tipos canônicos neste experimento não ocorreu consolidação de recursos. Isso acontece porque, quando há muitos tipos canônicos a estimativa de recursos é realizada de maneira mais precisa,

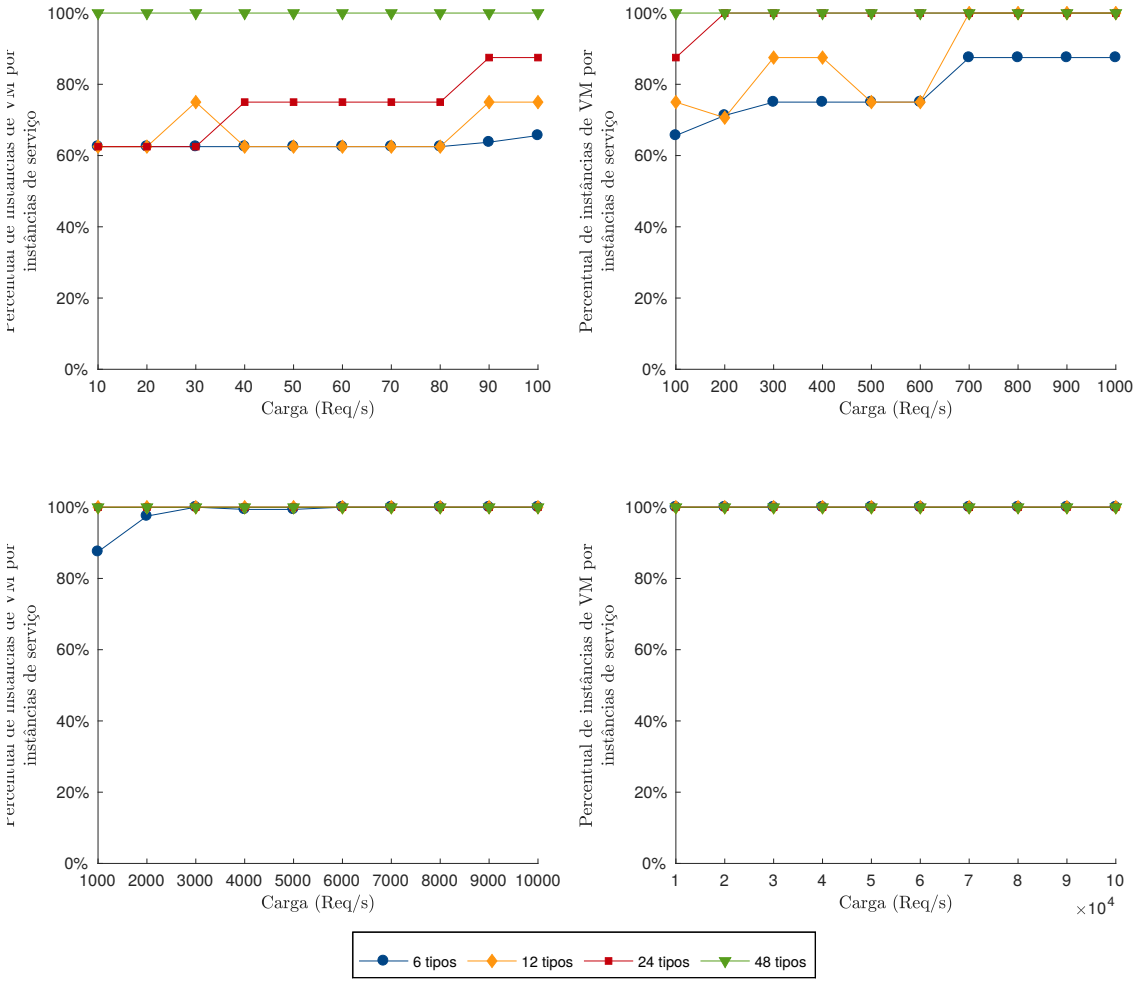



**Figura 8.4:** *Percentual de instâncias de VM por instâncias de serviço, de acordo com a carga em cada modo de utilização da síntese para o Experimento II.*

não havendo, portanto, capacidade excedente nos recursos concretos representados pelos tipos canônicos mapeados. Esse comportamento pode ser verificado nos casos em que ocorreu consolidação: quanto menor a quantidade de tipos canônicos considerados maior o percentual de recursos consolidados.

A estimativa de recursos mais precisa, proporcionada pelo uso de uma quantidade maior de tipos canônicos, tem a vantagem de permitir que os recursos sejam selecionados tendo como base capacidades mais próximas da real necessidade de cada serviço. Contudo, tal vantagem nem sempre é traduzida como uma redução no custo de utilização dos recursos uma vez que o custo dos recursos não é considerado na estimativa. Outra razão é que a possibilidade de consolidação quando uma quantidade menor de tipos canônicos é usada pode fazer com que o custo dos recursos selecionados seja inferior, mesmo a estimativa ocorrendo de maneira mais imprecisa.

Conforme já discutido, a consolidação de recursos tem impacto no custo dos recursos selecionados por permitir que uma quantidade menor de instâncias seja

alocada. A Figura 8.5 apresenta o custo dos recursos selecionados para cada um dos modos de utilização considerados no experimento, de acordo com o aumento na carga.

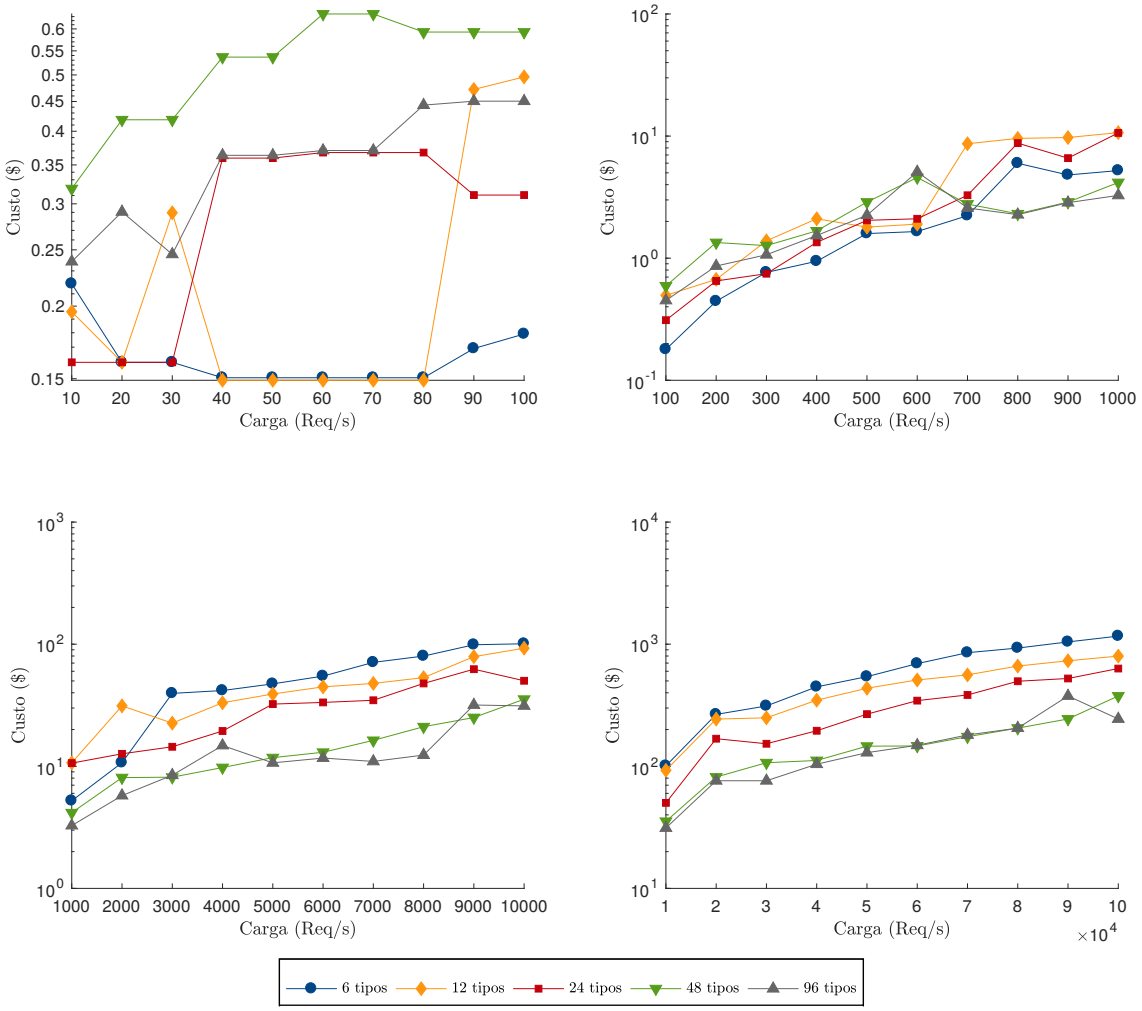



**Figura 8.5:** *Custo dos recursos selecionados, de acordo com a carga e a quantidade de tipos canônicos.*

Apesar do custo não ser um atributo incluído na representação dos tipos canônicos, a decisão de privilegiar a escolha de tipos com menor capacidade faz com haja redução no custo dos recursos. Como pode ser visto, quanto maior a quantidade de tipos canônicos considerados, menor o custo dos recursos selecionados. Isso pode ser claramente visto nos casos em que não ocorreu consolidação de recursos, ou seja, para cargas acima de 2.000 req/s no experimento. Por outro lado, a imprecisão na estimativa de recursos resultante do uso de uma quantidade menor de tipos canônicos é compensada pela possibilidade de realizar consolidação de recursos. Como foi ilustrado no gráfico da Figura 8.4, para cargas menores (predominantemente até 100 req/s no experimento) a consolidação ocorre com mais frequência, sendo o custo dos recursos selecionados (apresentado na Figura 8.5) nos casos que ocorreu consolidação

inferior ao custo quando se considera uma quantidade maior de tipos canônicos.

Neste experimento, quando não ocorre consolidação (cargas superiores a 2.000 req/s), a relação entre o aumento na quantidade de tipos canônicos e a redução no custo de utilização dos recursos selecionados é observada para 94,47% dos casos.

A relação descrita acima não foi verifica em todos os casos em virtude do procedimento adotado para a geração dos tipos canônicos: uma vez que o custo de utilização dos tipos concretos não é considerado na geração do modelo canônico, não é possível garantir que a capacidade representada por um tipo canônico possua relação com o custo dos tipos concretos por ele representados. Isto reforça a necessidade de levar em consideração o custo dos tipos concretos como um dos parâmetros a serem usados na geração do modelo canônico. Essa alteração na estimativa de recursos é necessária para garantir que os resultados obtidos representem o menor custo em todos os casos.

Outro critério relevante para analisar a efetividade da síntese quanto ao custo de utilização dos recursos selecionados seria comparar a proximidade dos resultados obtidos aos resultados ótimos para o problema. Contudo, obter soluções ótimas neste caso demanda alto processamento, pois o espaço de busca é muito grande: $t^n$, em que $t$ é o número de tipos concretos no primeiro nível da representação e $n$ é o número de serviços (827 e 9, respectivamente em nosso exemplo). Cada tupla que representa um resultado possível deve ser avaliada de acordo com todas as restrições não-funcionais.

Implementamos uma solução de busca exaustiva melhorada, na qual, usando os identificadores dos tipos concretos, geramos as combinações possíveis de recurso em ordem lexicográfica usando o algoritmo M (*Mixed-radix generation*) proposto por Knuth [152]. Tendo como base as tuplas geradas como possíveis resultados, as restrições são então avaliadas. Quando é identificada a violação de alguma das restrições para uma determinada tupla, as tuplas com mesmo prefixo, ou seja, que possuem os mesmos identificadores de tipos concretos até o ponto em que a violação foi detectada, não precisam ser avaliadas. Contudo, mesmo considerando coreografias como a de rastreamento de frete, composta por apenas 4 serviços, o tempo de processamento dessa solução foi superior a 22 horas para cada iteração, o que tornou inviável gerar esses valores em nossa análise.

Com relação ao tempo de processamento da síntese, a Tabela 8.2 apresenta os resultados para essa variável. É importante reforçar porém que o tempo de processamento da síntese não é influenciado pela carga. Como era esperado, quanto maior a quantidade de tipos canônicos utilizados maior o tempo médio de processamento da síntese. A razão disso é que a estimativa de recursos deve ser realizada considerando mais alternativas, o que causa maior sobrecarga para se chegar a resultados próximos da solução ótima como garantido pela heurística utilizada.

| Quantidade de tipos canônicos | Tempo (s) |
|---|---|
| 6 | 0,038 |
| 12 | 0,149 |
| 24 | 0,358 |
| 48 | 1,591 |
| 96 | 6,022 |

**Tabela 8.2:** *Tempo de processamento da síntese em cada modo de utilização considerado.*

Estes resultados, juntamente com os demais discutidos nesta seção, indicam a necessidade de haver um balanceamento da quantidade de tipos canônicos a serem considerados em cada caso da síntese, em vez de adotar uma quantidade padrão para todas as entradas. Isso ocorre porque o uso de uma quantidade maior de tipos canônicos pode tornar o resultado mais efetivo com relação ao custo de utilização dos recursos, em casos onde a carga é elevada. Por outro lado, utilizar um conjunto amplo de tipos canônicos tem a desvantagem de inibir a possibilidade de consolidação de recursos para casos com cargas menores, além de aumentar o tempo de processamento da síntese.

A análise do tempo de processamento da síntese em ambos os experimentos discutidos nesta seção teve como objetivo complementar os demais resultados obtidos para cada um dos modos de utilização considerados. Contudo, para obtermos resultados mais conclusivos sobre este aspecto realizamos uma análise mais profunda em outro experimento que será discutido na próxima seção.

## 8.2 Tempo da síntese de recursos

O tempo da síntese de recursos consiste no intervalo de tempo entre o início de seu processamento e a obtenção do resultado. A eficiência com relação ao tempo de síntese não é essencial nas atividades realizadas durante a implantação das coreografias, visto que são atividades realizadas antes de sua encenação, possivelmente em equipamentos dedicados. Contudo, o oferecimento de um nível satisfatório de desempenho passa a ser um requisito se a síntese for realizada em tempo de execução dos serviços, para realizar adaptações na alocação de recursos. Em outras palavras, o tempo de síntese precisa ser reduzido para que seja possível a reexecução da síntese como estratégia adaptativa nos casos em que as restrições não-funcionais são violadas, sobretudo devido a mudanças na carga.

Como forma de avaliar esse aspecto e comprovar a factibilidade de uso da abordagem proposta também em tempo de execução, realizamos o experimento descrito nessa seção.

### 8.2.1 Definição do experimento

O objeto de estudo do experimento é o tempo necessário para obter o resultado da síntese de recursos. Essa característica é particularmente importante porque, além dos casos em que ocorre a submissão de novas coreografias, o uso da síntese em tempo de execução será realizado em situações em que as restrições não-funcionais são violadas. Dessa forma, quanto mais tempo for necessário para solucionar a violação, maior será o período em que os serviços são utilizados com qualidade aquém da necessária.

Para analisar este aspecto, realizamos uma análise mais profunda acerca deste aspecto, considerando os piores casos na entrada da síntese. Dessa forma, o foco do experimento é verificar se o tempo necessário para obter a seleção de recursos é aceitável, considerando os contextos em que a síntese de recursos é utilizada.

### 8.2.2 Planejamento do experimento

A principal variável analisada nesse experimento é o **tempo de processamento da síntese**, medido em segundos (*s*). Na medição dessa variável, consideramos apenas o tempo necessário para realizar a estimativa e seleção de recursos, ignorando o tempo necessário para geração do modelo de recursos, visto que essa tarefa é realizada uma única vez.

Apesar da sobrecarga causada pela síntese de recursos na máquina em que ela é executada não ser um aspecto incluído no problema considerado, essa informação é relevante para decidir se o ambiente em que a abordagem será implantada é apropriado. Dessa forma, também avaliamos essa característica coletando o **uso de CPU e memória**. Essas variáveis são medidas como percentual da capacidade total da máquina utilizada no experimento. O experimento foi realizado considerando a execução da abordagem proposta em uma estação de trabalho convencional.

Na entrada utilizada no experimento, buscamos garantir que a topologia do grafo de dependências gerado, a carga sobre os serviços e as restrições especificadas fizessem com que o tempo de processamento fosse aumentado progressivamente. Este aumento ocorre principalmente devido à lógica de avaliação das restrições e a sucessivos insucessos na estimativa e na seleção de recursos. Nosso objetivo em adotar esta estratégia foi realizar a análise considerando os piores casos do problema. Diante disso, o experimento considera coreografias sintéticas cujo grafo de dependência segue o padrão ilustrado na Figura 8.6, em que cada vértice representa um serviço distinto e cada árvore representa um subconjunto de serviços conectados pelo padrão de conjunção.

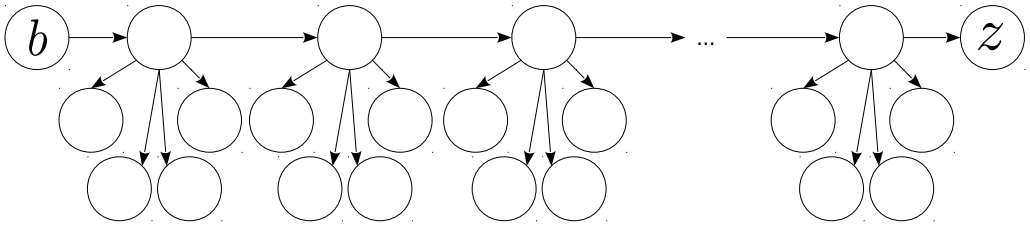



**Figura 8.6:** *Topologia das composições consideradas no experimento de tempo de processamento da síntese.*

Adotamos essa topologia porque o padrão de conjunção causa uma sobrecarga adicional na síntese, pois neste caso para obter a contribuição de um serviço é preciso avaliar a contribuição dos demais serviços que são executados em paralelo com ele. Ademais, essa estrutura tem a vantagem de representar um processo de negócio típico, em que ocorrem chamadas frequentes a outros serviços e a junção das respostas a estas chamadas, e possui um padrão repetitivo que pode ser usado para aumentar o tamanho da entrada na síntese.

Consideramos que as coreografias estão sujeitas a um conjunto de restrições não-funcionais, também gerado de maneira sintética. Para requerer maior tempo de processamento, consideramos apenas restrições de QoS fim-a-fim e apenas restrições classificatórias envolvendo todos os serviços.

Como forma de garantir o pior caso no procedimento de síntese, consideramos uma carga elevada sobre as coreografias ($10^4$ req/s no experimento), de forma que insucessos na estimativa de recursos sejam frequentes, fazendo com essa atividade tenha que ser refeita reiteradamente. Além disso, forçamos a ocorrência de insucesso na seleção de recursos ao estabelecer como restrição que somente tipos concretos referentes ao tipo canônico com menor capacidade seriam aceitáveis. Com isso, qualquer resultado envolvendo outro tipo canônico causaria insucesso na seleção de recursos, gerando a necessidade de reiniciar a estimativa de recursos.

O tempo de processamento da síntese de recursos é obtido em função de quatro parâmetros: quantidade de serviços na síntese, quantidade de tipos de recurso (concretos e canônicos) disponíveis, quantidade de restrições de QoS que devem ser avaliadas na estimativa de recursos e quantidade de restrições em relação a atributos do recurso que devem ser avaliadas na seleção de recursos. O modelo de recursos foi considerado constante porque variações não ocorrerão com tanta frequência para um mesmo contexto de uso da síntese, visto que o mesmo modelo provavelmente será utilizado em sucessivas execuções da síntese.

A síntese de recursos foi realizada variando a quantidade de serviços de 5 a 1.000 serviços, com incremento de 5 serviços a cada nova iteração. A inclusão de novos serviços segue o padrão de topologia adotado para a entrada.

Quanto às restrições estabelecidas para a entrada, utilizamos cenários com

1, 5, 10 e 15 restrições de cada categoria considerada (QoS e classificatória). Não consideramos valores mais elevados porque dificilmente haverá cenários em que será necessário implantar um conjunto de coreografias com um número muito superior de restrições a estas quantidades consideradas. Para as restrições de QoS, adotamos a latência como métrica em todas as restrições. Para restrições classificatórias também consideramos um único critério em todas as restrições, que consiste em selecionar tipos de acordo com a reputação do provedor.

Para que o processamento na classificação dos tipos concretos não fique concentrado na primeira restrição em relação a atributos do recurso, estabelecemos nesse experimento que todos os provedores têm mesma reputação, o que gera a necessidade de avaliação de todas as restrições. Isso acontece porque, no procedimento adotado na etapa de filtragem e classificação de tipos concretos, quando há o encadeamento de restrições classificatórias, a restrição seguinte na cadeia só é avaliada quando o critério de ordenação referente à restrição atualmente sendo avaliada indicar que os dois tipos em comparação são iguais. Isso é feito para preservar a ordem parcial estabelecida por cada restrição.

O processo adotado na geração das restrições não reduz o tempo de processamento da síntese, uma vez que nesse experimento as restrições são incluídas após a geração do grafo de dependências, não ocorrendo, portanto, a junção de restrições equivalentes.

O monitoramento de uso da CPU e da memória no experimento foi realizado com o utilitário `sysstat`.

### 8.2.3 Operação do experimento

Para permitir a execução do experimento foi implementado um programa auxiliar que gera a estrutura do grafo de dependências na topologia estabelecida, dado o número de serviços e restrições. Além disso, foi implementado um *script* responsável por realizar chamadas à síntese de recursos considerando as variações na entrada. Os resultados sobre uso de CPU e memória foram coletados usando outro *script* e armazenados em arquivos de texto.

Durante o experimento, os dados sobre uso de CPU e memória foram coletados a uma taxa de 1 Hz. O *script* de coleta dessas variáveis era inicializado no início do processamento da síntese e interrompido quando o resultado era obtido.

Para a execução deste experimento foi utilizada a mesma máquina descrita na Seção 8.1.3.

Para permitir maior confiabilidade, cada iteração em ambos os experimentos foi repetida 30 vezes e os resultados foram gerados com intervalo de confiança de

95%. Como os erros foram desprezíveis, eles não foram considerados na análise e interpretação dos resultados.

Para as variáveis uso de CPU e uso de memória, o volume de dados coletados foi muito grande, tornando inviável a apresentação conjunta de todos os resultados. Dessa forma, para a apresentação dos resultados foi selecionado o intervalo de 300 medições (que corresponde a 5 minutos do experimento), que contém maior média para cada uma destas variáveis. Adotamos esta estratégia porque o objetivo foi analisar o pior caso na execução da abordagem de síntese.

### 8.2.4 Análise e interpretação dos resultados

A Figura 8.7 apresenta os resultados sobre o tempo de processamento da síntese. Os eixos representam os parâmetros variados no experimento e a cor de cada ponto indica o resultado propriamente dito, cuja escala é apresentada (em segundos) na barra de cores ao lado.

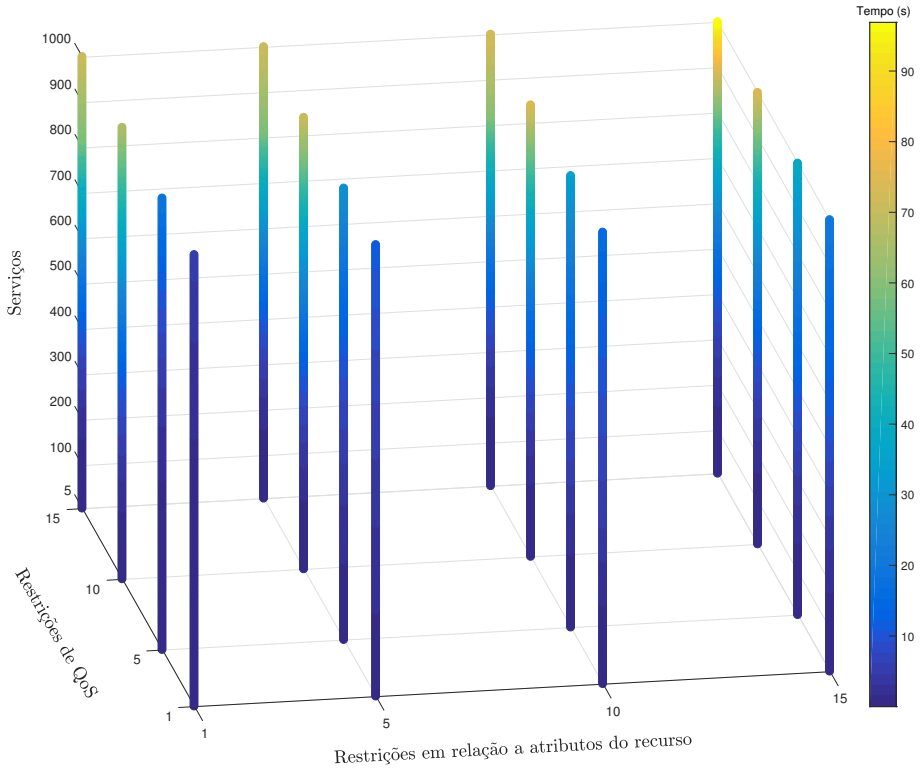



**Figura 8.7:** *Tempo de processamento da síntese de recursos, de acordo com a quantidade de serviços e restrições.*

Uma vez os pontos no gráfico representam cenários com parâmetros diferentes, é preciso analisar como o tempo de processamento da síntese se comporta de acordo com o tamanho da entrada. A Tabela 8.3 apresenta algumas medidas estatísticas realizadas sobre os resultados do experimento. Como é possível notar, o tempo de processamento se concentra nos valores mais reduzidos, o que pode ser confirmado

| **Medida** | **Valor (*s*)** |
|---|---|
| Média da diferença do tempo de processamento entre amostras suscetivas conforme o aumento na quantidade de serviços | 0,24 |
| Média da diferença do tempo de processamento entre amostras suscetivas conforme o aumento na quantidade de restrições em relação a atributos do recurso | 1,61 |
| Média da diferença do tempo de processamento entre amostras suscetivas conforme o aumento na quantidade de restrições de QoS | 7,05 |
| 75º percentil da diferença do tempo de processamento entre amostras suscetivas conforme o aumento na quantidade de serviços | 0,35 |
| 75º percentil da diferença do tempo de processamento entre amostras suscetivas conforme o aumento na quantidade de restrições em relação a atributos do recurso | 3,65 |
| 75º percentil da diferença do tempo de processamento entre amostras suscetivas conforme o aumento na quantidade de restrições de QoS | 15,40 |

**Tabela 8.3:** *Medidas estatísticas dos resultados do experimento sobre tempo de processamento da síntese*

pela predominância das cores frias no gráfico. Além disso, a quantidade de restrições de QoS tem maior influência sobre o tempo de processamento do que a quantidade de restrições em relação a atributos do recurso e a quantidade de serviços. Comparando a diferença para cada categoria separadamente e fixando os demais parâmetros, percebemos que o aumento na quantidade de serviços causa uma sobrecarga média de $0,24s$, ao passo que o mesmo aumento na quantidade de restrições em relação a atributos do recurso causa uma sobrecarga média de $1,61s$; e na quantidade de restrições de QoS uma sobrecarga média de $7,05s$. Essa diferença é reforçada pelos valores do 75º percentil para cada uma destas variações: $0,35s$, $3,65s$ e $15,40s$, respectivamente. Isso acontece porque as contribuições de QoS precisam ser balanceadas para cada restrição e elas têm interferência umas sobre as outras. Por outro lado, para avaliar restrições classificatórias é necessário realizar a ordenação dos tipos concretos, o que é feito em tempo linear.

Apesar do tempo de processamento da síntese parecer elevado para ser usado em tempo de execução, ele deve ser comparado com o tempo de processamento das demais atividades envolvidas na implantação da coreografia, para avaliar se é viável sua utilização. Com relação a isso, uma vez que não consideramos a utilização de VMs previamente instanciadas, a alocação de recursos é a atividade que constitui o principal gargalo. Isso ocorre porque as VMs têm que ser instanciadas e configuradas, o que envolve o tempo de comunicação com o provedor, de carregamento da VM e de instalação dos componentes de *software*. Mesmo não tendo realizado em nosso trabalho nenhum experimento a esse respeito, há uma série de resultados na literatura que indicam que o tempo de instanciação de uma VM em provedores públicos varia entre

44,2 e 810,2$s$ [124, 132, 174]. Esses resultados consideram apenas o tempo de inicialização da VM, ou seja, o tempo decorrido desde o envio da solicitação ao provedor até que seja possível realizar a primeira comunicação com a VM (por exemplo, via conexão `ssh` em máquinas Linux). Ao considerar as outras tarefas necessárias para que a coreografia possa ser considerada apta para receber requisições, esse tempo pode ser bem mais elevado. Como exemplo, a avaliação do componente CHOReOS Enactment Engine, descrita em [161], demonstra que o tempo de implantação de um conjunto de coreografias com um total de 100 serviços varia entre 433,1 e 623,3$s$, ao passo que para um conjunto maior, formado por 1.000 serviços, o tempo varia entre 1.296,1 e 1.614,3$s$. Em nosso experimento, o tempo de processamento da síntese para entradas com essas quantidades de serviços variou de 0,08 a 0,67$s$, e de 5,24 a 96,88$s$, respectivamente. Dessa forma, o tempo de processamento da síntese é aceitável, pois, mesmo considerando piores casos na entrada, adiciona uma sobrecarga proporcionalmente pequena na duração de todo o processo.

Com relação ao uso de CPU e memória no processamento da síntese de recursos, a Figura 8.8 apresenta os resultados para um intervalo de 5 minutos do experimento. Os valores de uso da CPU acima de 100% se devem ao uso simultâneo de múltiplos núcleos da CPU.

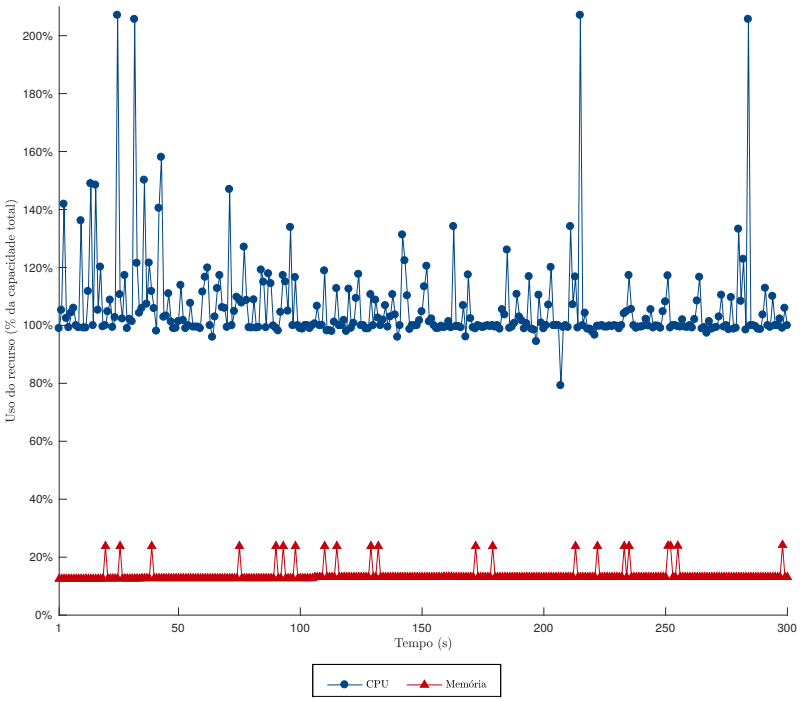



**Figura 8.8:** *Uso de CPU e memória no processamento da síntese de recursos.*

A alta utilização da CPU se deve ao fato da implementação ter sido projetada visando explorar toda a capacidade da máquina, visando reduzir o tempo de processamento da síntese. Para isso, nos baseamos em funcionalidades oferecidas pela bibli-

oteca *Java Concurrency API* [200] para criar múltiplas *threads* no processamento dos pontos mais críticos da síntese, de forma que todos os núcleos da CPU fossem utilizados.

Não visualizamos esse resultado como um problema da solução implementada, pois pode-se assumir, sem nenhum prejuízo, que a máquina onde a síntese é processada constitui um recurso dedicado para essa tarefa. Ademais, a síntese não será processada durante todo o tempo, mas apenas quando novas coreografias tiverem que ser implantadas ou quando houver mudanças significativas na carga. Contudo, para não impor essa sobrecarga como algo inevitável no uso da solução, definimos um parâmetro de configuração em que é possível desabilitar o uso de múltiplas *threads*, fazendo com que o processo utilize apenas um núcleo da CPU. Contudo, é importante destacar que o tempo de processamento da síntese ao adotar essa opção é aumentado significativamente, o que pode inviabilizar seu uso em tempo de execução.

É importante ressaltar que nesse experimento buscamos expandir a infraestrutura de execução da síntese para ambientes de menor capacidade computacional. Por esse motivo, o processamento foi realizado em um *desktop* convencional, conforme descrito na Seção 8.1.3. Com isso, buscamos comprovar que o tempo de processamento e a sobrecarga são aceitáveis, mesmo considerando ambientes restritos como este. Contudo, cenários de maior escala podem utilizar ambientes de nuvem para executar a síntese de recursos, de forma a se beneficiar de maior capacidade computacional e, assim, reduzir ainda mais o tempo de processamento. Como trabalho futuro, esperamos aperfeiçoar a implementação realizada para ser possível também explorar o paralelismo através de componentes como unidades de processamento gráfico (*Graphics Processing Unit* – GPU), dado o crescente avanço em sistemas de *middleware* para virtualização destes componentes em nuvem [81, 195, 242] e o surgimento de novos modelos de programação que permitem explorar sua capacidade [193, 272].

## 8.3 Estabilidade da síntese de recursos

A avaliação da estabilidade da síntese de recursos é baseada em executar repetidamente a abordagem proposta para uma mesma entrada, ou considerando mudanças pontuais na entrada original. Com base nessas execuções suscetivas, é verificado se o resultado se mantém o mesmo em relação àquele inicialmente obtido, ou são identificadas quais divergências ocorreram ao se comparar novos resultados com o inicial. Esse critério refere-se exclusivamente a aspectos dinâmicos do gerenciamento de recursos. O objetivo em avaliá-lo é analisar a viabilidade de reexecutar a síntese como parte do mecanismo para adaptação dinâmica e também porque é desejável haver al-

gum grau de determinismo entre implantações sucessivas de um mesmo conjunto de coreografias sob condições iguais ou semelhantes.

A estabilidade na seleção de recursos é importante para garantir que novas execuções da síntese não alterem significativamente mapeamentos de serviços para recursos previamente definidos. Caso contrário, a reimplantação de uma quantidade significativa de serviços poderá impactar negativamente a encenação das coreografias já em execução.

Embora não tenha sido projetada para a adaptação dinâmica, a análise proporcionada com este experimento é importante para avaliar a abordagem de síntese quanto ao seu reuso em tempo de execução para absorver mudanças na carga das coreografias.

### 8.3.1 Definição do experimento

O objeto de estudo do experimento é o mapeamento estabelecido entre serviços e recursos pela abordagem de síntese proposta.

O objetivo de sua realização é verificar quão extenso é o número de mudanças que precisam ser realizadas na alocação de recursos, em virtude da seleção de outros tipos de recurso ao executar novamente a síntese. Em complemento a isso, busca-se verificar se as adaptações indicadas pela reexecução da síntese são concentradas no conjunto de serviços que de fato sofreram algum tipo de mudança.

### 8.3.2 Planejamento do experimento

Conforme descrito no Capítulo 6, na abordagem de síntese proposta, quando há múltiplas instâncias de um mesmo serviço, elas são tratadas de forma conjunta, sendo todas mapeadas para o mesmo tipo de recurso que foi selecionado pela síntese para o serviço em questão. Dessa forma, neste experimento consideramos “serviço” e “instância de serviço” como sinônimos. Com base nisso, a variável analisada nesse experimento consiste no **número de mudanças de pares serviço/recurso**, que mapeia cada serviço das coreografias de entrada para o tipo concreto de recurso selecionado para sua implantação.

Neste experimento, consideramos um conjunto de coreografias para os quais os recursos já foram selecionados e analisamos o resultado da síntese frente a alterações na carga para apenas uma dessas coreografias, permanecendo os serviços das demais coreografias com a carga original. Dessa forma, avaliamos a quantidade de mudanças nos pares serviço/recurso e se estas mudanças ocorrem somente em serviços que compõem a coreografia afetada pela alteração na carga, ou se elas são propagadas para serviços que não compõem essa coreografia. Essa análise é necessária porque a

estimativa de recursos é realizada na síntese visando ao balanceamento global da QoS entre os serviços, não considerando coreografias isoladamente. Com isso, mesmo as adaptações sendo pontuais em uma única coreografia, elas conceitualmente podem influenciar os demais serviços.

As mudanças no mapeamento de recursos podem ocorrer em duas situações: em virtude do mapeamento para outro tipo canônico na estimativa de recursos, o que consequentemente leva à seleção de outro tipo concreto; ou elas podem ser resultado da consolidação do recurso para o qual o serviço havia sido anteriormente mapeado. Nesse último caso, as mudanças ocorrem porque pode haver o mapeamento de serviços para recursos com maior capacidade excedente, favorecendo a consolidação de recursos. Se mudanças devido à consolidação ocorrem, consideramos que a sobrecarga causada pela adaptação é aceitável quando busca-se redução do custo de utilização dos recursos selecionados ou no custo de comunicação entre os serviços. Considerando esta diferenciação, classificamos as mudanças que podem ocorrer no mapeamento de recursos em quatro categorias:

- *Interna (R)* : Mudanças que ocorrem para serviços que compõem a coreografia afetada pela alteração na carga, e que correspondem exclusivamente à alteração do tipo de recurso para o qual serviços são mapeados, sem ocorrência de consolidação.
- *Interna (C)* : Mudanças que ocorrem para serviços que compõem a coreografia afetada pela alteração na carga, e que são causadas pela consolidação do recurso anteriormente selecionado no mapeamento.
- *Externa (R)* : Mudanças que ocorrem para serviços que não compõem a coreografia originalmente afetada, e que correspondem exclusivamente à alteração do tipo de recurso para o qual o serviço é mapeado, sem ocorrência de consolidação.
- *Externa (C)* : Mudanças que ocorrem para serviços que não compõem a coreografia originalmente afetada, e que são causadas pela consolidação do recurso anteriormente selecionado no mapeamento.

A propagação das mudanças para serviços que não foram afetados com alteração na carga é o principal aspecto que deve ser evitado na síntese de recursos, uma vez que os recursos previamente selecionados para estes serviços continuam sendo suficientes para satisfazer as restrições impostas sobre eles. Além disso, é desejável que a reexecução da síntese não gere a necessidade de reimplantar uma quantidade significativa de serviços dentro de uma mesma coreografia. Devido a alterações na carga, mesmo sendo inevitável haver a necessidade de reimplantação de ao menos parte dos serviços que compõem a coreografia que foi afetada, essas mudanças devem ser reduzidas para que a encenação desta coreografia não seja afetada de modo considerável.

Assim como no experimento anterior, a entrada foi gerada sinteticamente. Consideramos um conjunto formado por duas coreografias, variando o nível de compartilhamento de serviços entre elas. Não avaliamos cenários com mais coreografias porque, dado que a carga varia em apenas uma coreografia, a inclusão de mais coreografias na entrada poderia ocasionar a dispersão das mudanças externas entre as coreografias que não tiveram alteração na carga. Dessa forma, ao considerar apenas uma coreografia adicional, as mudanças externas ficam concentradas nela, permitindo avaliar com maior exatidão o objeto do experimento.

A Figura 8.9 mostra o padrão do grafo de dependências gerado a partir da entrada considerada no experimento. Cada grafo de dependência corresponde a um cenário no experimento. Os serviços destacados fazem parte da coreografia que tem alteração na carga, ao passo que os serviços que não estão destacados fazem parte somente da coreografia que não sofre mudanças. Os serviços destacados e com borda tracejada são compartilhados pelas duas coreografias.

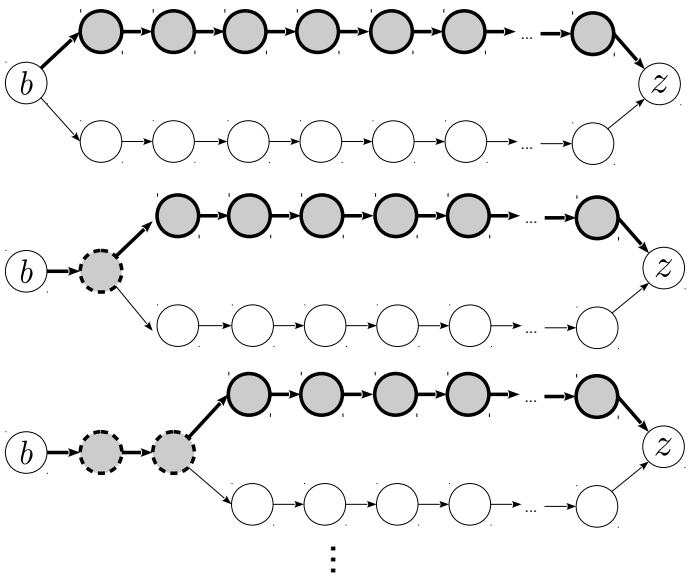



**Figura 8.9:** *Topologia das composições consideradas no experimento sobre estabilidade da síntese.*

Consideramos que as coreografias possuem a mesma quantidade de serviços (10 no experimento) e variamos a quantidade de serviços compartilhados em 20%, 40%, 60% e 80% da quantidade total de serviços em uma coreografia. Para permitir controle sobre a carga, consideramos que todos os serviços implementam uma mesma operação. Dessa forma, é possível estabelecer a carga suportada, dada uma determinada seleção de recursos.

Para a segunda coreografia, estabelecemos uma carga fixa de 10 req/s e variamos a carga da primeira coreografia de 10 a 1.400 req/s, em intervalos de 10 requisições. A carga mínima nesse caso foi estabelecida para garantir que o uso do tipo canô-

nico menos robusto, por todos os serviços, seria suficiente para satisfazer as restrições, dada a demanda de recursos calculada de acordo com as operações envolvidas nas coreografias. De maneira semelhante, a carga máxima foi estabelecida por ser suficiente para requerer o mapeamento de ao menos um dos serviços da coreografia com carga afetada para o tipo canônico de recurso mais robusto. Dessa forma, a variação na carga considerando este intervalo permite forçar mudanças gradativas para tipos canônicos com maior capacidade, até que todos os tipos canônicos sejam utilizados como parte do resultado.

Em cada coreografia, estabelecemos uma única restrição de QoS fim-a-fim usando a latência como métrica. O valor alvo para essa restrição é mantido o mesmo em todos os cenários, como sendo $1s$. Não estabelecemos nenhuma restrição em relação a atributos do recurso, pois elas não afetam a estabilidade da síntese, uma vez que são avaliadas de maneira determinística, obtendo sempre resultados específicos para uma mesma entrada.

As mudanças em pares serviço/recurso são medidas de acordo com um mapeamento inicial gerado como resultado da primeira iteração em cada cenário do experimento. De maneira mais precisa, inicialmente executamos a síntese de recursos com a carga estabelecida como 10 requisições em ambas as coreografias. Dado o número de instruções configurado para a operação usada, é possível garantir que o mapeamento de recursos seria gerado usando o tipo canônico com menor capacidade para todos os serviços. Contudo, para comprovar essa hipótese, a síntese foi executada 30 vezes com essa mesma entrada e confirmamos que em todas as execuções o resultado foi exatamente o mesmo. Portanto, esse resultado foi adotado como o mapeamento inicial na verificação das mudanças nos demais casos do experimento.

Devido ao aumento gradativo na carga, a partir de um certo ponto todos os pares referentes à primeira coreografia deverão sofrer mudança para que seja possível acomodar novas requisições. Por consequência, a partir deste ponto, o número de mudanças nos pares serviço/recurso será sempre igual ao número de serviços, se um único mapeamento inicial for utilizado na comparação. Isso torna difícil obter conclusões sobre a estabilidade para os serviços da coreografia que teve alteração na carga. Em virtude disso, realizamos novamente o experimento definindo múltiplos mapeamentos para serem usados na comparação para detecção de mudanças. Cada um destes mapeamentos é gerado de acordo com a carga suportada pelos tipos canônicos. Dessa forma, o comportamento esperado é que entre o ponto em que foi gerado um mapeamento inicial e o ponto em que foi gerado um mapeamento adicional, a quantidade inicial de mudanças seja zero (ou um valor reduzido) e que esta quantidade aumente gradativamente. Isso permite diferenciar os casos em que as mudanças são consequência do aumento na carga ou da instabilidade da síntese.

### 8.3.3 Operação do experimento

Para permitir a execução do experimento, foi implementado um programa auxiliar que gera a estrutura do grafo de dependências na topologia estabelecida, dado o nível de compartilhamento de serviços.

Assim como nos experimentos anteriores, visando maior confiabilidade, cada iteração do experimento foi repetida 30 vezes e os resultados foram gerados com intervalo de confiança de 95%. Como os erros foram desprezíveis, eles não foram considerados na análise e interpretação dos resultados.

### 8.3.4 Análise e interpretação dos resultados

A Figura 8.10 apresenta a quantidade de mudanças (por categoria) ao considerar um único mapeamento inicial.

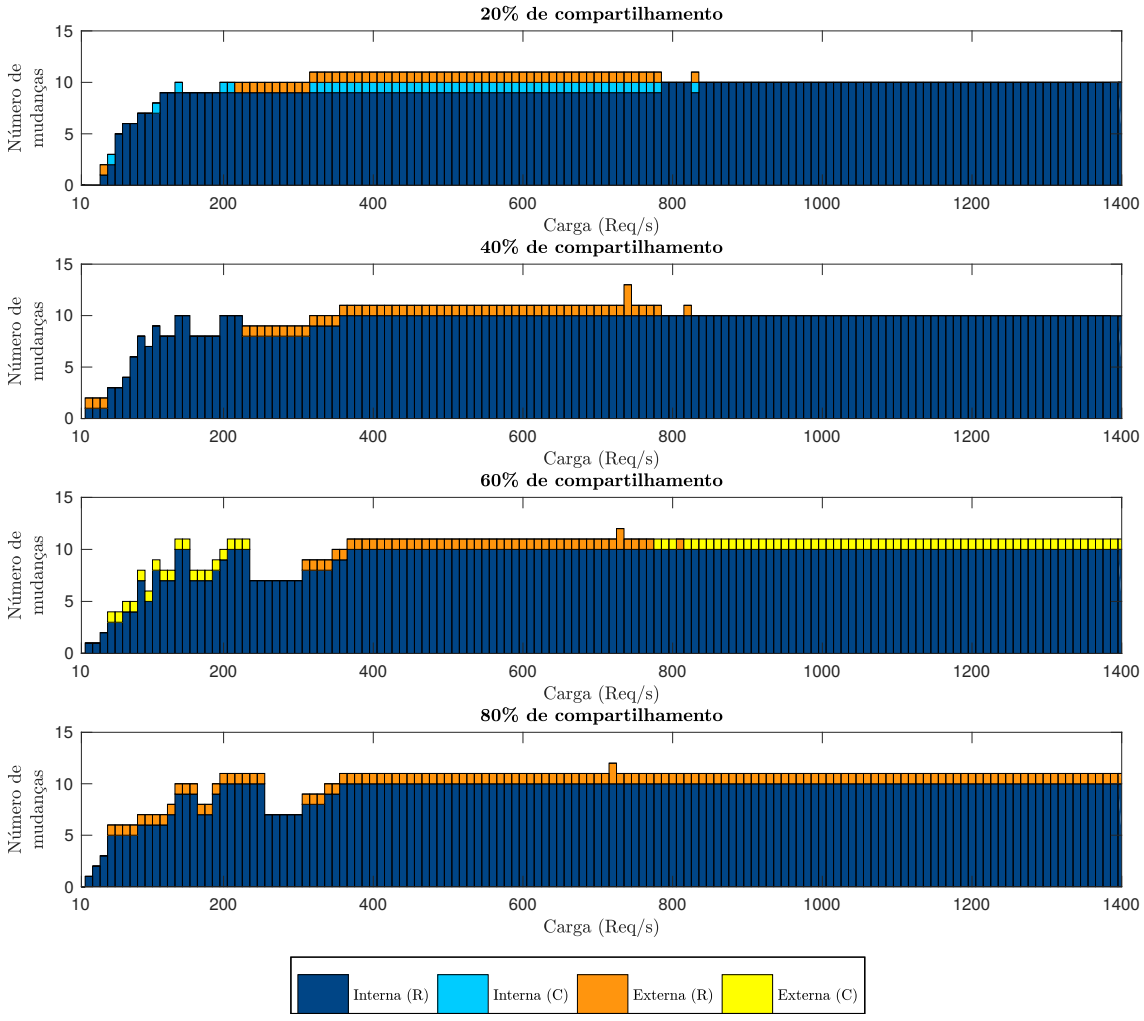



**Figura 8.10:** *Quantidade de mudanças de acordo com a carga considerando um único mapeamento inicial.*

Com o aumento da carga, é esperado que mudanças ocorram, uma vez que é necessário selecionar recursos com maior capacidade para ser possível satisfazer as restrições.

Como mostrado nos gráficos, essas mudanças aumentam até que todos os serviços que tiveram alteração na carga (no total 10 no experimento) tenham seus tipos de recurso redefinidos. A partir desse ponto, a quantidade de mudanças é mantida como o total de serviços, uma vez que o mapeamento considerado como inicial deixa de ser válido para todos os serviços.

Ao considerar 20% de compartilhamento de serviços no experimento, além de mudanças de tipos de recurso para ambas as coreografias, ocorrem também algumas mudanças associadas à consolidação de recursos para a coreografia que teve aumento na carga.

Analisando o mapeamento de recursos resultante pudemos notar que esse comportamento acontece porque os tipos selecionados para os serviços compartilhados possuem mais capacidade excedente que nos outros níveis de compartilhamento.

Comportamento semelhante ocorre quando há 60% de compartilhamento, mas considerando os serviços que não compõem a coreografia alterada. Nesse caso, a consolidação ocorre porque a demanda de recursos para esses serviços é menor, podendo ser atendida pela capacidade excedente nos recursos selecionados para a coreografia que teve alteração na carga.

Apesar do aumento significativo na carga, as adaptações ficaram concentradas nos serviços que tiveram alteração. Em quase todos os casos em que foram verificadas mudanças externas devido ao balanceamento realizado pela estimativa de recursos, elas ocorrem para apenas um serviço, não sendo superior a 3 serviços no cenário considerado no experimento.

Este comportamento indica que o mapeamento da síntese pode ser considerado estável por não interferir significativamente em serviços que não sofreram alteração.

Esses resultados não permitem obter outras conclusões sobre os serviços da coreografia que sofreu alteração na carga porque, conforme discutido anteriormente, o número de mudanças a partir de um certo ponto será o mesmo, uma vez que o mapeamento inicial não será mais válido para nenhum dos serviços desta coreografia.

Dessa forma, para permitir uma análise mais ampla sobre a estabilidade com relação à mudanças internas, consideramos também diferentes mapeamentos iniciais. Os resultados são apresentados na Figura 8.11. As linhas vermelhas indicam os pontos em que novos mapeamentos iniciais foram gerados. Esses pontos coincidem com o limite de carga suportado por recursos com capacidade estabelecida pelos tipos canônicos.

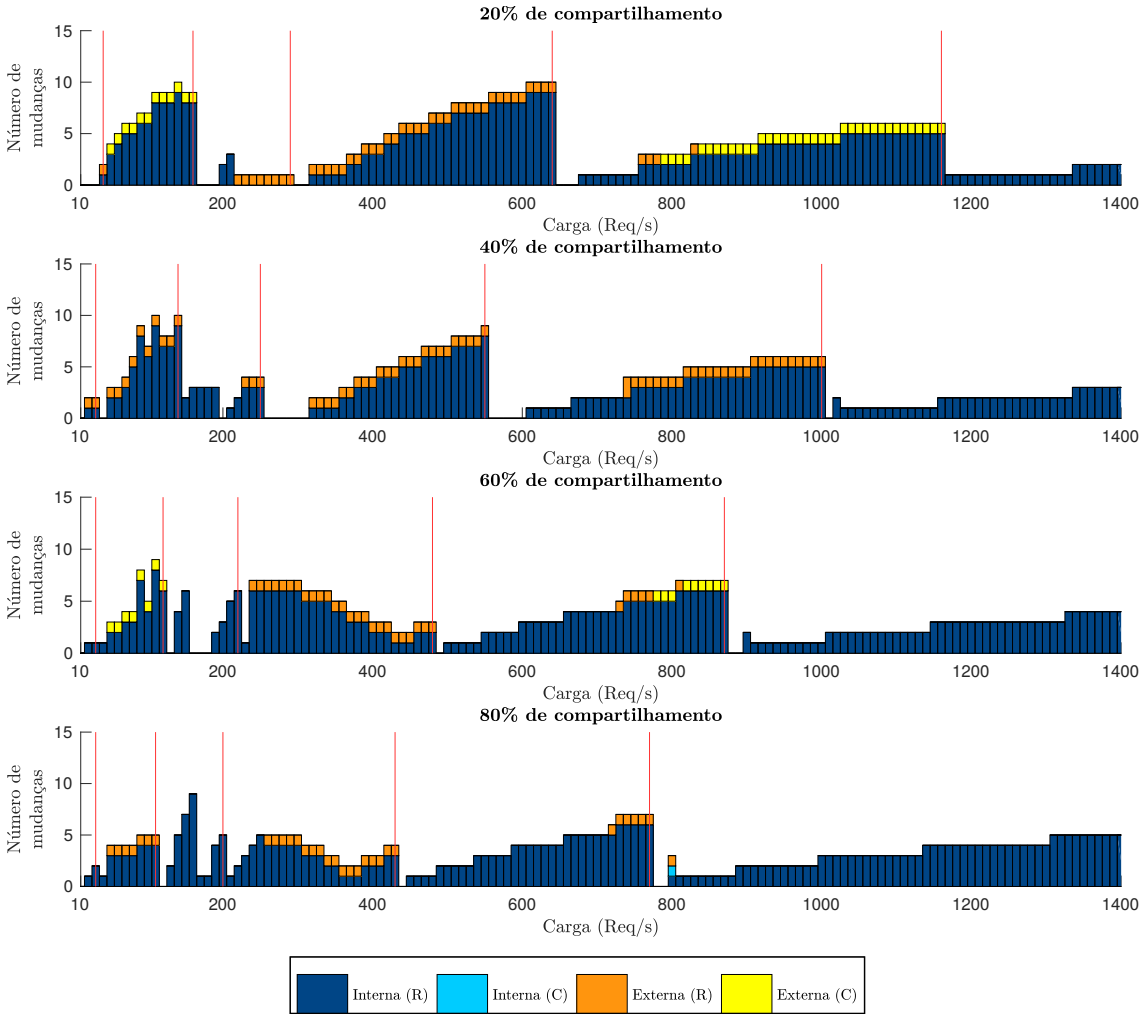



**Figura 8.11:** *Quantidade de mudanças de acordo com a carga considerando múltiplos mapeamentos iniciais.*

Esse método de comparação permite verificar com mais exatidão as mudanças necessárias em virtude da carga. Conforme esperado, elas ocorrem de maneira gradativa, a partir do ponto em que um novo mapeamento inicial é considerado. Ao realizar esse procedimento, não foram verificadas mudanças associadas à consolidação de recursos para a coreografia que teve aumento na carga. Com relação à outra coreografia, mudanças nessa categoria ocorrem, sobretudo para cargas menores.

Novamente é possível notar que as mudanças são concentradas na coreografia que teve alteração na carga, sendo que as mudanças externas foram limitadas a apenas 1 serviço em todos os casos. Contudo, é possível verificar que há instabilidade nos resultados da síntese. O comportamento esperado devido ao aumento na carga é que as mudanças ocorram de forma gradativa. Contudo, como pode ser visto neste experimento, por exemplo, para 60% de compartilhamento de serviços, o número de mudanças internas devido à alteração do tipo – Interna (R) –, para 160 req/s é 0, ao passo que para 150 req/s é 6. Em outras palavras, as mudanças a partir da geração de

um mapeamento inicial são decrescentes, não apresentando o comportamento esperado, o que constitui um inconveniente porque pode requerer que adaptações desnecessárias sejam acionadas. Isso acontece em virtude de dois fatores:

1. o aumento na carga faz com que seja mais vantajoso selecionar uma quantidade menor de tipos com maior capacidade do que um número intermediário de tipos com menor capacidade. Por exemplo, a heurística seleciona o tipo `LARGE` para 2 serviços que anteriormente eram mapeados para o tipo `SMALL`, em vez de selecionar o tipo `MEDIUM` para 3 serviços que anteriormente eram mapeados para o tipo `SMALL`; ou
2. a instabilidade inerente da heurística WS-HEU, utilizada na estimativa de recursos. Essa heurística obtém o resultado fazendo alterações aleatórias em uma solução originalmente encontrada por um método guloso. Dessa forma, apesar do uso da heurística assegurar que o resultado será 96% do valor ótimo, não é possível garantir que este resultado será sempre o mesmo.

## 8.4 Análise geral dos resultados

Neste capítulo foram apresentados alguns experimentos para avaliar a abordagem proposta para a síntese de recursos. A avaliação foi realizada com base no protótipo descrito no capítulo anterior e nas coreografias e recursos usados nos exemplos apresentados no decorrer da tese, além de entradas sintéticas. A realização dos experimentos teve como objetivo analisar a efetividade da abordagem proposta e verificar a viabilidade de seu uso também em tempo de execução. Para isso, avaliamos seu tempo de processamento e a estabilidade nos resultados.

Dentre as principais conclusões obtidas com os experimentos, podem ser citadas:

- A consolidação de recursos acontece com mais frequência quando a carga sobre os serviços não é muito elevada. O motivo desse comportamento é que cargas menores levam ao mapeamento dos serviços para tipos canônicos que representam recursos com maior capacidade excedente. Particularmente, a ocorrência de cargas muito elevadas leva ao mapeamento para o tipo canônico mais robusto, que não possui nenhuma capacidade excedente. Outros fatores que favorecem a consolidação de recursos são: o uso da técnica de troca de tipos usando como critério a localidade, a disponibilidade de um conjunto maior de tipos concretos na seleção de recursos e o emprego de uma quantidade menor de tipos canônicos na estimativa de recursos.

- A consolidação de recursos reduz de forma significativa o custo dos recursos selecionados, nos casos em que ela é possível. Isso justifica o fato da abordagem proposta assumir o uso desta técnica, em conjunção com a técnica de troca de tipo usando como critério a localidade.
- O uso de consolidação de recursos reduz mais significativamente o atraso de comunicação do que a troca de tipos usando como critério a localidade. Para ambas as técnicas, apesar do valor médio de atraso de comunicação entre pares de serviços ser aparentemente elevado, a maioria dos valores para essa variável (considerando o 75º percentil no cenário do experimento) é baixo. Dessa forma, o uso dessas técnicas reduz o atraso de comunicação, embora imponha sobrecarga adicional no processamento da síntese de recursos.
- Considerar uma quantidade maior de tipos canônicos na estimativa de recursos faz com que essa atividade seja realizada de maneira mais precisa, fazendo com que o custo de utilização dos recursos selecionados seja menor. Contudo, a imprecisão na estimativa ao considerar menos tipos é compensada pela possibilidade de realizar a consolidação de recursos para cargas menores, fazendo com que o custo seja até menor em alguns casos que consideram um conjunto mais limitado de tipos canônicos. De maneira particular, a quantidade de tipos canônicos a ser considerada na geração deste modelo deve levar em consideração a carga esperada para as coreografias. Isso é necessário porque o uso de uma quantidade menor de tipos favorece a ocorrência de consolidação, desde que a demanda de recursos não seja muito elevada, o que ocorre em casos onde a carga é moderada. Com isso, em trabalhos futuros a quantidade de tipos canônicos deve ser balanceada de acordo com a entrada.
- Mesmo considerando casos extremos na entrada, o tempo de processamento da síntese de recursos pode ser considerado satisfatório ao se comparar com o tempo gasto na alocação de recursos. No cenário considerado no experimento, o acréscimo de restrições de QoS causou sobrecarga em média $\approx 5$ vezes maior que o acréscimo de restrições em relação a atributos do recurso e $\approx 30$ vezes maior que o acréscimo de serviços. Este resultado não constitui um problema pois acreditamos que em cenários reais, como o ilustrado pelo exemplo de cenário apresentado no Capítulo 2, a ocorrência de restrições de QoS será inferior à ocorrência de restrições em relação a atributos dos recursos.
- O aumento na quantidade de serviços na entrada da síntese causa pouca sobrecarga no tempo de processamento necessário para obter o resultado. No experimento realizado, a inclusão de 5 serviços na entrada (fixando os demais parâmetros) causou uma sobrecarga média adicional de apenas $0,24s$. Com isso, argumentamos que há indícios de que o uso da abordagem também é factível

para cenários de grande escala. Contudo, para consolidar esta afirmação é preciso uma análise mais abrangente, por exemplo, considerando variações maiores no tamanho da entrada. Esta análise é tida como trabalho futuro.

- Ao executar novamente a abordagem de síntese proposta para coreografias previamente analisadas, as mudanças na seleção de recursos ficam concentradas em serviços que tiveram aumento na carga. Grande parte das mudanças sobre os serviços que tiveram alteração na carga se deve à consolidação de recursos, o que representa economia no custo dos recursos selecionados. Em virtude dos resultados obtidos, é possível afirmar que grande parte das mudanças verificadas são esperadas em virtude do aumento na carga. Além disso, a síntese possui comportamento estável, de forma que mudanças não desejáveis ocorrem em uma frequência reduzida. Isso permite concluir que é possível utilizar em trabalhos futuros a abordagem de síntese como parte do mecanismo para adaptação dinâmica.

Apesar de ter sido possível obter conclusões satisfatórias, como trabalho futuro, cenários adicionais devem ser considerados nos experimentos apresentados e novos experimentos devem ser realizados para avaliar outros aspectos relacionados à abordagem. Dentre estes aspectos, é preciso avaliar a efetividade da síntese, de forma a identificar o quão próximo os resultados obtidos estão próximos de soluções ótimas para o problema. Além disso, é preciso incluir nesta avaliação outros critérios de efetividade, além do custo financeiro de utilização dos recursos e do atraso de comunicação. Mais precisamente, é preciso avaliar a efetividade da síntese de acordo com a satisfação das restrições em tempo de execução dos serviços. De maneira particular, deve-se avaliar, em ambientes reais de nuvem, se os recursos selecionados pela síntese conseguem de fato oferecer a QoS requisitada nas restrições. Não incluímos esta análise em nossa avaliação pois, conforme será discutido no próximo capítulo, algumas suposições assumidas em nossa abordagem (como admitir que a capacidade descrita para um recurso descreve de maneira precisa o desempenho que este irá apresentar) precisam ser revistas e melhor exploradas para garantir a satisfação das restrições em todos os casos.

Além de avaliar a efetividade da abordagem, é necessário compará-la com outras abordagens. Este critério não foi considerado em nossa avaliação pois não encontramos nenhuma abordagem que considerasse todos os aspectos que propomos em nossa solução, de forma que a comparação ficaria limitada a um escopo reduzido do problema tratado. Apesar dessa limitação, reconhecemos a necessidade de realizar esta comparação adotando esta limitação ou meramente considerando abordagens mais simplificadas como, por exemplo, alocar recursos de maneira *ad hoc* para cada serviço e fazer adequações nesta alocação com base em monitoramento, de forma

a contornar violações nas restrições. Contudo, a adoção de estratégias minimalistas como esta, ou a simples criação de novas instâncias dos serviços para suprimir o efeito do compartilhamento de serviços entre as coreografias, constituem alternativas que, muito provavelmente, ocasionarão desperdício de recursos ou violação excessiva das restrições. Contudo, é preciso uma avaliação mais detalhada para comprovar esta afirmação e medir o ganho de fato obtido com o uso da abordagem apresentada.

Por fim, como outro aspecto que precisa ser avaliado, está o fato de ponderamos no decorrer da tese que nossa abordagem é de fácil utilização. Entre os fatores que favorecem este argumento está o fato de que a especificação de restrições torna-se simples ao eliminarmos aspectos como a modelagem de pesos e critério de decisão, e ao propormos uma linguagem elementar para a especificação de restrições em relação a atributos do recurso. Além disso, devido à adoção de adaptadores para a notação do grafo de processo, é possível se beneficiar da abordagem mesmo fazendo uso de linguagens já consolidadas como BPMN. Contudo, outros experimentos devem ser realizados para comprovar cientificamente estes critérios de usabilidade.

CAPÍTULO 9

# Trabalhos futuros

O principal objetivo do trabalho – propor uma abordagem para a implantação automatizada e eficiente de múltiplas coreografias de serviço, sujeitas a restrições não-funcionais associadas aos serviços e à sua encenação – foi atingido. Contudo, identificamos diversas possibilidades de extensão deste trabalho que, por restrição de tempo, não puderam ser desenvolvidas. Além disso, detectamos alguns pontos que devem ser melhor investigados como desdobramento dos objetivos iniciais da pesquisa desenvolvida. Neste capítulo discutimos os principais.

## 9.1 Maior nível de abstração na avaliação de restrições não-funcionais

Propomos um arcabouço que facilita a especificação de restrições não-funcionais. Contudo, seu uso pressupõe que esta tarefa é realizada quantificando o valor alvo para essas restrições. Isso é uma desvantagem, pois o usuário nem sempre consegue discriminar com exatidão os valores suportados para que a implantação e encenação das coreografias sejam realizadas de maneira satisfatória. Em virtude dessa possível imprecisão, a experiência tida pelos clientes na utilização das coreografias pode ser insatisfatória ou a implantação das coreografias pode ser realizada de maneira excessivamente restrita, causando gastos desnecessários. Mesmo considerando a especificação dos valores alvos usando faixas de tolerância o problema persiste pois ainda assim nem sempre é possível definir os limites com exatidão. Dado esta limitação, uma possível extensão em nossa abordagem seria oferecer um maior nível de abstração sobre a especificação e consequente avaliação de restrições não-funcionais.

Uma alternativa possível seria descrever o valor alvo para as restrições usando, por exemplo, linguagem natural como em *"O cumprimento de pedidos deve ser realizado de maneira* ***rápida****"*. Na fase inicial de nossa pesquisa sugerimos a adoção desta estratégia, como pode ser visto em nossa publicação [108]. De acordo com a definição proposta por Chung *et al* [62], o objetivo inicial era tratar restrições não-funcionais

como metas flexíveis (*soft-goals*), isto é, metas que precisam ser satisfeitas não absolutamente mas de maneira suficientemente adequada, e para as quais o critério de satisfação não é definido de forma clara (quantificada). Esse tipo de notação é desejável por ser facilmente compreendido pelo usuário, independente de seu conhecimento técnico. Essa exigência de alto nível do usuário seria então interpretada de formas diferentes, dependendo dos serviços que impactam sobre ela e do contexto de seu uso, conforme realizado em alguns trabalhos [170, 254]. Com isso, a quantificação das restrições é realizada em tempo de execução, de acordo com a experiência dos clientes e execuções prévias dos serviços.

Uma vez que a quantificação de restrições de maneira automática deve lidar com diversos desafios associados ao tratamento de incertezas e de influências entre as restrições, optamos por simplificar a especificação das restrições adotando a quantificação do valor alvo pelo usuário, para que fosse possível focarmos no desenvolvimento da síntese de recursos. Contudo, esta estratégia pode ser incorporada em trabalhos futuros na arquitetura proposta como parte da camada de síntese de recursos.

## 9.2 Priorização de restrições não-funcionais

Na abordagem proposta não consideramos a inclusão de prioridades sobre as restrições estabelecidas. Esse comportamento foi adotado por assumirmos que são estabelecidas apenas restrições essenciais, que deviam necessariamente ser satisfeitas com mesmo grau de importância. Contudo, em alguns cenários pode haver restrições opcionais que se não forem satisfeitas não impactarão de forma significativa a encenação da coreografia. Como exemplo, pode-se definir que a latência de encenação de uma dada coreografia deve ser inferior a um certo valor e que é desejável que seus serviços sejam implantados em uma certa localidade, embora essa localidade possa ser alterada caso isso seja necessário para que a latência requisitada seja satisfeita. Neste exemplo, a localidade requisitada deixa de ser uma restrição que necessariamente deve ser satisfeita pela síntese.

A priorização de restrições também pode ser estabelecida através da definição de pesos, que podem ser usados para influenciar a seleção de recursos favorecendo a satisfação de uma determinada restrição. Como exemplo, para uma dada coreografia pode haver restrições associadas à vazão e à segurança, cada qual com seu valor alvo, de forma que a segurança tem maior prioridade que a vazão. Com isso, a seleção de recursos realizada pela síntese terá como resultado um conjunto de recursos que satisfaçam ambas as restrições mas que privilegiem a escolha de recursos que ofereçam um maior nível de segurança.

Apesar das vantagens descritas, a possibilidade de definir essa priorização tem o inconveniente de requerer que o usuário assuma mais responsabilidades, dificultando a implantação de coreografias. Essa dificuldade constitui outra razão pela qual preferimos simplificar a especificação de restrições. Além disso, a incorporação de prioridades nas restrições em nossa abordagem requer a adequação do arcabouço de especificação de restrições, além da adaptação do mecanismo de síntese propriamente dito e da redefinição do conceito de função de utilidade adotado em nosso trabalho, de forma a considerar a influência dos pesos sobre as restrições. Diante disso, o oferecimento de priorização de restrições como trabalho futuro requer também uma análise sobre o balanceamento entre os benefícios obtidos e o impacto na usabilidade e nos critérios avaliados para a solução.

## 9.3 Inclusão do usuário no *loop*

A estimativa e a seleção de recursos em nossa abordagem adotam o custo de utilização dos recursos como critério de decisão nos casos em que há diferentes opções que satisfaçam todas as restrições. De maneira semelhante, na consolidação de recursos a demanda de comunicação entre os serviços é usada como critério de decisão. O uso desses atributos foi fixado na abordagem por considerarmos que eles representam os critérios mais críticos nas atividades em questão. Contudo, reconhecemos que pode haver cenários em que o uso de outros atributos como critério de decisão pode ser algo necessário, como, por exemplo, quando deseja-se implantar o maior número possível de serviços em um mesmo provedor[1]. Diante disso, uma extensão possível do trabalho seria permitir a configuração dos critérios usados nas decisões realizadas.

Além de realizar essa configuração, propomos que o usuário seja incluído no *loop* em outras etapas da abordagem. Mais precisamente, sugerimos também que poderiam ser oferecidas diferentes soluções como resultado da síntese de recursos, em que cada solução privilegiaria algum critério como, por exemplo, menor custo para utilização dos recursos, implantação em um único provedor, dentre outros. Com base nos critérios estabelecidos para gerar os resultados, o usuário então selecionaria a melhor opção, que seria então usada como resultado final. Além disso, um aprimoramento ainda mais elaborado seria considerar *feedbacks* fornecidos pelo usuário sobre os resultados da síntese para aprimorar decisões futuras ou estabelecer novos critérios na geração de resultados.

[1] Em nossa abordagem o comportamento de privilegiar a seleção de recursos de um mesmo provedor pode ser obtido através da definição de restrições classificatórias usando, por exemplo, a reputação do provedor como critério. Contudo, a possibilidade de defini-lo como critério de decisão promoverá ainda mais sua ocorrência.

Atualmente não consideramos essas estratégias porque sua adoção tem a desvantagem destacada na subseção anterior, que é conferir ao usuário maiores responsabilidades, dificultando o uso da abordagem.

## 9.4 Aperfeiçoamento da avaliação de restrições de QoS

Com relação à restrições de QoS há alguns aspectos que precisam ser melhor consolidados para que a abordagem possua maior robustez e seja mais alinhada aos cenários reais de sua utilização. São eles: precisão na estimativa de QoS, tratamento efetivo de restrições de QoS sensíveis a localização e estabelecimento de SLAs.

### 9.4.1 Estimativa de QoS

O isolamento da estratégia utilizada para estimar a QoS para cada métrica foi uma simplificação adotada em nosso trabalho para que fosse possível focarmos no desenvolvimento do mecanismo de síntese de recursos. Contudo, para que a abordagem proposta seja amplamente empregada em um cenário com múltiplos provedores de nuvem é preciso aperfeiçoar como esse aspecto é tratado. Mais precisamente, é necessário propor abordagens concretas para a estimativa de QoS, além de tratamentos mais precisos para lidar com imprecisões nessas estimativas.

Há diversos resultados já consolidados que visam estimar a QoS com base em dados reais relacionados aos provedores de nuvem. Nesse sentido, as estimativas são obtidas usando estratégias baseadas em *benchmarking* ou se baseando no uso prévio de recursos nestes provedores [3, 133, 166, 238]. Outra alternativa possível é considerar soluções que fornecem dados de monitoramento de atributos de QoS. Po exemplo, CloudHarmony [65], além de fornecer dados sobre disponibilidade dos provedores (que usamos em nossa implementação), também dispõe informações sobre o desempenho de recursos nesses ambientes.

O grande desafio em incluir em nossa abordagem a estimativa de QoS de maneira precisa, mesmo considerando soluções como essas, é lidar com a variabilidade do desempenho oferecido por recursos instanciados em nuvens públicas. As informações oferecidas pelos provedores públicos de IaaS sobre a capacidade de *hardware* e o desempenho dos tipos de recurso oferecidos não são suficientes para prever como as aplicações neles implantadas irão se comportar [213]. Isso acontece porque são utilizadas diferentes gerações de máquinas físicas na infraestrutura destes provedores, conforme declarado por eles mesmos (por exemplo, AWS [16] e Microsoft Azure [183]). Além disso, há flutuação na utilização dessa infraestrutura a longo prazo (diferentes períodos do ano) e a curto prazo (diferentes horas do dia). Como consequência, o de-

sempenho realmente oferecido pode variar de acordo com o período de tempo em que a utilização é realizada e de acordo com a região utilizada, ou mesmo considerando recursos de uma mesma região, como indicado em [162, 196, 202, 213].

Consideramos que este desafio não invalida a abordagem desenvolvida, uma vez que tratamos a estimativa de QoS como algo isolado. Dessa forma, o funcionamento da síntese de recursos em nossa solução não sofre modificações mesmo considerando mudanças na estratégia para estimativa de QoS.

### 9.4.2 Restrições de QoS sensíveis a localização dos recursos

Em nosso trabalho assumimos que as restrições de QoS são especificadas com base em métricas de QoS diretamente relacionadas à capacidade dos recursos utilizados, como por exemplo latência de processamento. Contudo, ao se considerar restrições fim-a-fim, existem algumas métricas de QoS cuja avaliação depende também da sobrecarga imposta pela comunicação entre os serviços. Um exemplo imediato seria tempo de resposta fim-a-fim, cujos valores devem considerar não somente a latência de processamento em cada serviço, mas também a latência de comunicação entre eles. Em nossa abordagem consideramos a satisfação de restrições que se encaixam nessa categoria e propomos um tratamento inicial.

No tratamento proposto, as restrições de QoS sensíveis a localização dos recursos são avaliadas em duas etapas. A primeira etapa é realizada durante a estimativa de recursos, juntamente com a avaliação integral das demais restrições de QoS. Nesta primeira etapa apenas a parcela da restrição que é dependente da capacidade de recurso (como latência de processamento no tempo de resposta) é avaliada. Após isso, durante a seleção de recursos, para cada serviço que está relacionado a uma restrição de QoS ainda não totalmente avaliada, ou seja, aquelas que dependem da localização dos recursos, é selecionado um conjunto de tipos concretos de recurso admissíveis, em vez de um único tipo concreto. Essa seleção é realizada durante a consolidação de recursos e também considera um custo adicional admissível para os recursos candidatos. Dessa forma, o resultado da consolidação de recursos seria o mapeamento de conjuntos de serviços para uma tupla que representa o tipo mais favorável (selecionado de acordo com a estratégia de troca usando como critério a localidade), juntamente com outros tipos que podem ser considerados como substitutos a esse (embora possam representar uma redução global menor no atraso de comunicação).

Após a seleção de recursos, a parcela das restrições que é dependente da localização dos serviços (como latência de comunicação no tempo de resposta) seria verificada usando o tipo de recurso mais favorável especificado em cada tupla. Caso alguma restrição fosse violada ao considerar a parcela remanescente nesta restrição, propusemos criar uma nova instância do problema MMKP para tentar mapear os

serviços a tipos que permitam a satisfação das restrições ainda violadas. Para isso, os tipos admissíveis incluídos na consolidação de recursos seriam usados como recursos candidatos.

O tratamento proposto na avaliação dessa categoria de restrições foi motivado pelo fato dos tipos concretos de recurso geralmente possuírem tipos com capacidade equivalente em regiões distintas, tornando viável a substituição de modo a satisfazer as restrições sensíveis a localização. Contudo, o tratamento proposto precisa ser melhor fundamentado e ainda é preciso desenvolver uma análise mais aprofundada da sua viabilidade, assim como propor alternativas quando soluções factíveis não são encontradas. Por essa razão, consideramos como trabalho futuro o tratamento deste aspecto.

### 9.4.3 Estabelecimento de SLAs

Além de melhorias na estimativa, é preciso tornar a abordagem proposta mais alinhada ao modelo de negócio atualmente adotado em soluções de nuvem. Uma extensão nesse sentido seria considerar a criação de acordos de nível de serviço (*Service Level Agreements* – SLAs), juntamente com políticas de compensação e penalidade quando eles não são cumpridos. Esses acordos especificam as expectativas e obrigações mínimas das partes envolvidas no processo de negócio [44] e, dessa forma, seu estabelecimento pode oferecer maior confiança na utilização da abordagem, assim como permitir maior maleabilidade no tratamento de restrições uma vez que possíveis violações podem ser neles discriminadas.

Um desafio em propor essa adequação é gerenciar os múltiplos níveis de SLA, dado que qualquer acordo estabelecido com o usuário deve ter como base acordos firmados com provedores de nuvem. Contudo, os SLAs atualmente oferecidos por provedores de nuvem pública são limitados aos atributos de custo e disponibilidade, havendo pouca ou nenhuma garantia com relação ao desempenho oferecido e demais atributos de qualidade. Diante disso, além de propor mecanismos para o estabelecimento e gerenciamento de SLAs, é preciso também oferecer estratégias para obter garantias sobre os atributos não assegurados pelos provedores públicos, de forma a ser possível considerá-los.

## 9.5 Aperfeiçoamento do modelo de recursos

Por simplicidade, o modelo de recursos utilizado na síntese é gerado considerando apenas tipos de VM cuja instanciação é realizada sob demanda. Contudo, o modelo de negócio adotado por provedores de nuvem inclui um conjunto com diferentes categorias de recurso. Como discutido em [250], além da cobrança por uso sob

demanda, que constitui a base do modelo de nuvem, os recursos podem ser oferecidos também na forma de assinaturas por um determinado período de tempo, considerando um modelo de cobrança pré-pago em que o pagamento é realizado antes da utilização dos recursos. Outra alternativa é a cobrança através da combinação dessas categorias, com instâncias dedicadas definidas pela assinatura e adição de novas instâncias sob demanda, quando há a necessidade de uma quantidade maior de recurso. Opções mais sofisticadas também estão disponíveis. Como exemplo, a Amazon permite aos usuários licitarem a capacidade não utilizada de sua infraestrutura por meio de *Spot Instances* [15], através do qual na solicitação do recurso é fornecido um preço que o usuário está disposto a pagar, sendo o recurso instanciado somente se este preço for igual ou superior ao valor de mercado (que é atualizado a cada hora). Cada uma dessas categorias tem vantagens e desvantagens que podem ser exploradas na síntese de recursos, de forma a obter soluções ainda mais satisfatórias.

Diante disso, uma possível extensão na abordagem seria definir as categorias de recurso que fossem mais adequadas para cada cenário. Contudo, para que isso seja possível é necessário conhecimento sobre o perfil de uso dos recursos pela aplicação sendo implantada, além de estratégias para que a flutuação de custo dos recursos seja refletida no modelo gerado e para que incertezas associadas a pagamentos prévios na alocação de recursos não causem prejuízos desnecessários.

Além da inclusão de novos tipos no modelo de recursos, a modelagem atualmente proposta deve ser aperfeiçoada de forma a refletir todos os aspectos existentes na utilização de recursos em nuvem. Mais precisamente, fizemos algumas simplificações como ignorar a cobrança realizada por provedores de nuvem pública pelos dados trafegados em rede e pelo armazenamento de dados (imagens das VMs, discos adicionais, *etc.*). Incorporamos em nosso modelo apenas o custo associado à execução da VM, sendo os demais atributos ignorados. Adotamos essa simplificação porque estas cobranças somente são realizadas quando um certo limite é ultrapassado. Como exemplo, na Amazon a cobrança por transferência de dados só é realizada quando o volume de dados trafegados ultrapassa 1 GB por mês. Contudo, reconhecemos que esse aspecto deve ser incluído em nossa abordagem para que os resultados obtidos representem de fato o custo esperado. Essa adequação também deve lidar com desafios associados ao conhecimento do perfil das aplicações e tratamento de incertezas.

Outra característica que precisa ser melhor fundamentada em nossa abordagem é o fato de que os provedores definem os parâmetros de suas ofertas de diferentes maneiras. Como exemplo, Microsoft Azure realiza a cobrança da transferência de dados de acordo com a largura de banda contratada, ao passo que a Amazon cobra pelo volume de dados transmitidos. Além dos desafios já discutidos, essa característica incorpora a necessidade de considerar este nível de heterogeneidade na geração

do modelo de recursos.

## 9.6 Análise dos efeitos do compartilhamento de recursos

O compartilhamento de recursos, tanto físicos quanto virtuais, tem a vantagem de reduzir os custos associados não somente à implantação de serviços mas também ao próprio gerenciamento dos recursos. Por exemplo, como forma de explorar a prevalência de sistemas com arquitetura *multicore* e *manycore*, esta estratégia é cada vez mais adotada no escalonamento de recursos físicos, juntamente com outras práticas conhecidas como computação verde (*Green Computing*) [126], visando a redução do consumo de energia em *datacenters*.

A alocação de múltiplas VMs em um mesmo recurso físico é possível devido à capacidade de isolamento de desempenho oferecida pelos *hypervisors*, em componentes como núcleos da CPU, memória e disco. Contudo, componentes como *cache* da CPU e largura de banda para entrada e saída (em memória, em rede e em disco) são muito difíceis de serem isolados nos *hypervisors* atualmente existentes [192]. Isso pode comprometer o desempenho com o aumento do número de VMs co-localizadas na mesma máquina física. Adicionalmente, a co-localização de VMs aumenta o risco de exploração de vulnerabilidades de uma VM por outra, diminuindo a segurança [145].

Estes mesmos problemas são visualizados no nível das aplicações ao implantar dois ou mais serviços compartilhando a mesma VM, como realizado na técnica de consolidação de recursos proposta em nossa abordagem. Neste caso, além dos problemas listados, o isolamento entre serviços, que constitui uma das principais vantagens ao se utilizar virtualização, desaparece quando eles são implantados na mesma VM.

Dentre as estratégias que podem ser utilizadas para lidar com estes problemas está a modelagem de desempenho dos serviços, visando a identificação de padrões de uso dos recursos. Com isso, a alocação de recursos pode ser potencialmente melhorada para permitir a co-localização de serviços de forma a não introduzir interferência entre eles. Contudo, conforme evidenciado nas seções anteriores deste capítulo, a adoção desta estratégia requer conhecimento sobre a aplicação (ou sobre cada serviço isoladamente) para que seja possível definir seu perfil. Além disso, há diversos desafios que precisam ser analisados. De maneira mais precisa, o problema de degradação de desempenho é agravado ao se considerar o gerenciamento de recursos em múltiplos *datacenters* geograficamente distribuídos, uma vez que este problema aumenta devido à alta variação e frequente escassez de recursos de rede na Internet [271]. Ademais, a ausência de controle sobre os recursos físicos em nuvens públicas dificulta a resolução deste problema de maneira efetiva.

Como trabalho futuro, estratégias como definir o perfil de uso de recursos pelas aplicações podem ser incluídas como parte do mecanismo de consolidação de recursos proposto. Com isso, as decisões tomadas ao realizar esta tarefa poderão levar em consideração um melhor balanceamento entre a interferência no desempenho causada pelo compartilhamento de recursos e as reduções no custo e no atraso de comunicação proporcionadas ao implantar múltiplos serviços em um mesmo recurso. Porém, o desenvolvimento dessa melhoria requer também a análise de mecanismos para tratamento de problemas decorrentes da co-localização de VMs nos cenários em que é possível gerenciar os recursos físicos (por exemplo, quando se usa uma nuvem privada).

## 9.7 Adaptação dinâmica

O principal foco da abordagem apresentada nesta tese é a estimativa e a seleção inicial dos recursos necessários para satisfazer um conjunto de restrições não-funcionais impostas sobre um conjunto de coreografias de serviços. Contudo, um importante aspecto que deve ser considerado é garantir que essas restrições se mantenham satisfeitas durante a encenação das coreografias.

As restrições não-funcionais podem ser violadas em tempo de execução devido a diferentes motivos. De imediato, existem aqueles que são inerentes do cenário considerado, como flutuação na carga das coreografias implantadas ou submissão de novas coreografias que compartilham serviços com aquelas previamente implantadas (o que também causa flutuação na carga dos serviços implantados). Além disso, a elasticidade de uso da infraestrutura, aliada à pressão para acomodar mudanças nas regras de negócio, requer que as coreografias de serviços também sejam adaptáveis [160]. Desta forma, é natural supor que poderão ocorrer alterações nas restrições não-funcionais estabelecidas, além de mudanças diretas nas coreografias, como adaptações nos padrões de conexão dos serviços para refletir alterações na lógica da composição ou no modelo de negócio por ela implementado.

Em complemento às mudanças diretamente associadas às coreografias, a dinamicidade presente em ambientes de nuvem faz com que aplicações implantadas nesse tipo de ambiente tenham que lidar com mudanças que podem ocorrer nos recursos disponibilizados. Como exemplos de mudanças nessa categoria, pode haver alteração nos tipos concretos oferecidos ou variação nos tipos previamente existentes, como aumento no custo financeiro ou encerramento da disponibilização de um determinado tipo em algumas regiões. Na contramão disso, podem surgir novas oportunidades, como o oferecimento de novos tipos de recurso ou a ampliação de tipos disponibilizados em certas regiões.

Além de mudanças requisitadas e aquelas relacionadas ao ambiente, há também adaptações relacionadas à solução proposta, uma vez que suposições erradas ou estimativas mal elaboradas ou desatualizadas podem invalidar a seleção de recursos obtida. Como exemplo, a QoS de fato oferecida ao utilizar um recurso pode ser diferente da QoS estimada, fazendo com que a estimativa de recursos (e consequentemente a seleção) precise ser adaptada.

Em nossa proposta, consideramos todas essas mudanças e propomos uma arquitetura que permite a inclusão de mecanismos de adaptação para lidar com todas elas. De maneira preliminar, nesta seção propomos uma solução apenas para parte daquelas mudanças listadas como inerentes no início dessa discussão. Uma vez que optamos por aperfeiçoar o tratamento da abordagem com relação às atividades realizadas durante a implantação inicial das coreografias e por limitação de tempo não foi possível aprofundar esta parte da pesquisa. Dessa forma, o tratamento das adaptações possíveis de maneira completa, assim como a consolidação da solução descrita a seguir, são considerados como trabalho futuro. Para tal, diversos desafios associados à predição de mudanças, assim como ao desenvolvimento de soluções preventivas e/ou corretivas, precisam ser investigados para que a abordagem seja plenamente satisfatória em tempo de execução.

Conforme anunciado, definimos um tratamento para quando há mudanças na carga sobre serviços previamente implantados. Essas mudanças podem ser ocasionadas em dois cenários: nova demanda sobre coreografias previamente implantadas ou inclusão de novas coreografias que compartilham serviços com aquelas já em encenação. Em ambos os casos, propomos como estratégia para absorver as mudanças na carga a reexecução da síntese de recursos. O argumento para propor tal alternativa é que a abordagem de síntese proposta possui desempenho satisfatório o suficiente para ser utilizada em tempo de execução. Outro argumento é que a solução obtida com a abordagem de síntese é estável, ou seja, as mudanças a serem realizadas na alocação de recursos serão pontuais, concentradas naqueles serviços que tiveram mudança na carga, sem afetar consideravelmente os demais serviços. Esses argumentos são reforçados pelos resultados dos experimentos discutidos no Capítulo 8. Contudo, para que seja possível consolidar o uso desse tratamento como estratégia adaptativa é preciso explorar outros aspectos como, por exemplo, decidir os gatilhos que indicam a necessidade de reexecução da síntese.

## 9.8 Expansão da abordagem para outros componentes da pilha de *software*

A abordagem proposta tem como foco o gerenciamento de recursos apenas levando em consideração os serviços. Esta perspectiva é aceitável ao assumir que a sobrecarga causada por componentes que viabilizam a comunicação e a coordenação dos serviços é negligenciável. Contudo, em alguns cenários, sobretudo de maior escala, isso não é realista. Diante disso, uma das extensões propostas para o trabalho é considerar componentes do *middleware* de comunicação e aqueles responsáveis pela coordenação entre os serviços como entidades de primeira classe no problema. Com isso, a estimativa e seleção de recursos seriam realizadas levando em consideração também as demanda de recursos por esses componentes, fazendo com que estas atividades fossem realizadas de maneira mais precisa. Contudo, para que isso seja possível é necessário, dentre outras coisas, investigar formas de estimar o impacto causado por estes componentes na QoS da coreografia, como por exemplo definindo modelos analíticos sobre desempenho no nível de *middleware* [43].

Outra consideração relevante é que em nossa abordagem adotamos uma visão monolítica dos serviços, na qual os componentes que implementam cada serviço são tratados como um único componente. Apesar de não inviabilizar a abordagem desenvolvida, o desacoplamento dos componentes (ou camadas) que formam um serviço pode permitir que a especificação de restrições não-funcionais e o gerenciamento de recursos sejam realizados de maneira mais precisa. Como exemplo, se houvesse tal desacoplamento apenas a camada de dados do serviço de pagamento discutido no exemplo do cenário deveria ser implantada em um ambiente privado, podendo as demais camadas desse serviço ser implantadas de maneira distribuída em recursos alocados em nuvens públicas. Isso garante maior liberdade na implantação dos serviços, facilitando ainda mais a satisfação das restrições. A principal dificuldade em propor tal tratamento é o aumento na complexidade, uma vez que esse desacoplamento aumenta a granularidade e adiciona um nível extra de dependências. Com isso, além da influência que um serviço pode ter sobre outro, passa a ser necessário considerar a influência que componentes de um mesmo serviço têm entre si.

## 9.9 Descentralização da arquitetura

Da maneira como foi proposta, a execução dos componentes da arquitetura ocorre de maneira centralizada. Do ponto de vista da implantação das coreografias isso não constitui necessariamente um problema, dado que as atividades são realizadas de maneira desacoplada de sua encenação. Contudo, se considerarmos a realização des-

sas atividades em tempo de execução a centralização da arquitetura passa a ser um obstáculo, sobretudo se considerarmos o uso da mesma estratégia de síntese proposta em tempo de implantação. Isso acontece porque qualquer violação significativa nas restrições gerará a necessidade de processamento, tornando assim a síntese de recursos um gargalo.

A descentralização da arquitetura tem como motivação os mesmos benefícios alcançados com a coordenação dos serviços por meio de coreografias. Em outras palavras, a seleção de recursos por meio da cooperação entre componentes distribuídos da arquitetura permite que decisões sejam tomadas localmente, sem a necessidade de que haja sempre coordenação global. Contudo, é preciso revisar não somente a arquitetura, mas principalmente a implementação desenvolvida, além de propor soluções quando a descentralização não é algo imediato. Como exemplo, para garantir que soluções próximas do valor ótimo sejam obtidas, a avaliação de restrições de QoS na estimativa de recursos de forma distribuída pode requerer uma demanda de comunicação muito elevada entre os componentes que implementam essa funcionalidade.

## 9.10 Adequação da abordagem para o estilo arquitetural de microsserviços

A grande tendência no desenvolvimento baseado em serviços é propor o uso de serviços com escopo cada vez mais bem definido, assumindo a responsabilidade por tarefas reduzidas. Dessa forma, as aplicações passam a ser desenvolvidas como um conjunto de serviços pequenos, cada um funcionando em seu próprio processo e se comunicando por meio de mecanismos de baixa sobrecarga [203]. Além de benefícios como separação de interesses e modularização, esta estratégia de desenvolvimento, conhecida como estilo arquitetural de microsserviços [189], permite que serviços escalem de maneira independente, permitindo melhor uso dos recursos.

Apesar de nossa abordagem não oferecer uma solução que considera todos os aspectos desse cenário, estabelecemos isso como algo a ser realizado. De fato, algumas partes da abordagem proposta já foram projetadas tendo esse estilo arquitetural em mente. Um exemplo é o mecanismo de consolidação de recursos, que visa permitir o compartilhamento de recursos entre um conjunto de serviços, o que é uma premissa quando se tem serviços com escopo bem limitado e, consequentemente, baixa demanda de recurso. Desvantagens como a perda de isolamento entre os serviços, resultante do compartilhamento de VMs, podem ser facilmente amenizadas ao considerar adequações na abordagem proposta como, por exemplo, usando contêineres como unidade de alocação em vez de VMs.

CAPÍTULO 10

# Considerações finais

Nesta tese, propomos uma abordagem que se beneficia da multiplicidade de tipos de recurso disponibilizados em ambientes de nuvem para implantar um conjunto de coreografias de serviços sujeitas a restrições não-funcionais. Esta abordagem visa ao uso eficiente dos recursos de forma a reduzir os custos associados à utilização dos recursos e o atraso de comunicação entre os serviços. A adoção cada vez mais frequente de soluções baseadas em serviços vem mudando a forma como sistemas são concebidos, projetados e implementados. As aplicações passam a ser desenvolvidas como composições de serviços que são executados de maneira distribuída. Neste novo cenário, coreografias de serviços constituem um modelo de composição promissor, por permitir que a coordenação desses serviços seja realizada sem a necessidade de controladores centrais. Todavia, a encenação de coreografias deve ser realizada visando não apenas as funcionalidades associadas à aplicação, mas também propriedades não-funcionais oriundas do modelo de negócio que essa aplicação implementa ou resultantes da busca por soluções cada vez mais competitivas e com maior qualidade. Para garantir que essas propriedades sejam oferecidas, é preciso atuar nas atividades realizadas, desde a implantação dos serviços, dado que o ambiente de execução tem forte impacto sobre sua satisfação. Isso gera diversos desafios, sobretudo ao considerarmos que um mesmo serviço pode assumir diferentes responsabilidades resultantes de seu compartilhamento em múltiplas composições.

A abordagem proposta tem como objetivo permitir a implantação de um conjunto de coreografias visando a satisfação de restrições não-funcionais requeridas sobre os serviços. Os fundamentos considerados nessa abordagem englobam atividades associadas a todo o ciclo de gerenciamento de recursos, além do desenvolvimento de modelos empregados na automação dessas atividades. Mais precisamente, apresentamos mecanismos para a modelagem de coreografias de serviços com restrições não-funcionais associadas. Tendo como base os modelos criados usando estes mecanismos, propomos a geração de uma estrutura que representa uma visão conjunta das coreografias e das restrições. O objetivo em gerar esta estrutura é unificar a representação de cada serviço e, assim, tornar explícitas as dependências entre os ser-

viços, facilitando a identificação dos efeitos de seu provável compartilhamento. Além disso, propomos uma forma de modelagem de recursos que considera um conjunto de atributos comumente utilizados na descrição destas entidades, e que visa abstrair sua representação ao se considerar um ambiente de nuvem híbrida com infraestrutura privada e múltiplos provedores de nuvem pública. Devido à vasta variedade de tipos de recurso, apresentamos uma estratégia que, a partir da modelagem inicial, gera uma representação adicional dos recursos considerando a intersecção de tipos e, assim, reduz a sobrecarga da estimativa de recursos.

Alicerçados nos modelos de coreografias e recursos, desenvolvemos uma estratégia de síntese que obtém a estimativa de recursos através da satisfação de restrições específicas para cada serviço e do balanceamento da contribuição dos serviços em restrições fim-a-fim, globalmente para todas as coreografias. Após a estimativa, a estratégia de síntese desenvolvida seleciona os tipos de recurso mais apropriados para os serviços enquanto promove o compartilhamento de recursos e reduz não somente o custo associado ao uso dos recursos alocados mas também o atraso de comunicação entre os serviços. Com base na abordagem desenvolvida, propomos uma arquitetura que coordena todas as atividades relacionadas ao gerenciamento de recursos.

Em nossa abordagem, a satisfação de restrições não-funcionais estabelecidas sobre um conjunto de coreografias representa o principal fator considerado para decidir a configuração de recursos adequada para implantar essas coreografias. Diante disso, um dos aspectos que fazem com que a abordagem desenvolvida seja eficiente é o fato da análise realizada na avaliação destas restrições considerar os diferentes papéis assumidos pelos serviços devido ao seu provável compartilhamento. Essa eficiência é reforçada pela redução do atraso de comunicação entre os serviços ao se adotar a localidade dos recursos como outro critério na seleção, e ao se explorar a capacidade excedente nos tipos de recurso disponíveis, através da consolidação de recursos.

Além da utilização de uma estrutura conjunta para a representação de todas as coreografias, foram propostos outros métodos que favorecem a eficiência e o desempenho da abordagem de síntese desenvolvida. Dentre eles está a manipulação dos modelos de recurso como forma de viabilizar a utilização de um vasto conjunto de tipos; a categorização das restrições não-funcionais associada ao tratamento da atividade de estimativa de recursos isoladamente da atividade de seleção de recursos; e a abstração tanto da estimativa de QoS quanto da interação com provedores de nuvem.

Apesar de ser necessária uma análise mais abrangente para obter conclusões definitivas, apresentamos alguns experimentos realizados com um protótipo de parte da arquitetura proposta. Mais especificamente, foram implementados os componentes associados às atividades realizadas em tempo de implantação das coreografias, necessárias para operacionalizar sua encenação. Os resultados obtidos com esses experi-

mentos indicam a efetividade, o desempenho e a estabilidade da abordagem de síntese proposta. Foi possível concluir, dentre outras considerações, que *i)* o uso das técnicas propostas reduz os custos de utilização dos recursos, ao mesmo tempo em que reduz o atraso de comunicação entre os serviços; *ii)* o tempo necessário para obter a solução da síntese de recursos, mesmo com casos extremos na entrada, é aceitável, sobretudo ao se considerar a duração de todo o processo de implantação das coreografias; e *iii)* as soluções da síntese possuem comportamento estável, com possíveis adaptações na alocação de recursos concentradas nos serviços que sofreram aumento na carga em tempo de execução.

Com base nos resultados obtidos, argumentamos que o uso da abordagem proposta pode beneficiar usuários interessados em implantar uma ou mais coreografias de serviços, através da automação das atividades relacionadas ao gerenciamento de recursos. Ao adotar essa abordagem, é possível implantar um conjunto de coreografias de maneira eficiente, pois se leva em consideração a interferência causada pelo compartilhamento de serviços entre elas, possibilitando satisfazer todas as restrições não-funcionais enquanto é obtida a redução dos custos associados.

Por fim, argumentamos que a contribuição desenvolvida nesta tese é ampla o suficiente para considerar outros cenários além da implantação de coreografias de serviços. A solução elaborada pode ser generalizada para categorias de problemas que apresentam as mesmas características do cenário considerado. Mais precisamente, a abordagem de síntese pode ser empregada (com as devidas modificações) em outros cenários em que seja necessário mapear um conjunto de recursos (em geral) para entidades que utilizam esses recursos na realização de alguma tarefa ou atividade de forma cooperada, sujeita a um conjunto de restrições. Como exemplo, algumas das contribuições desenvolvidas podem ser aplicadas no gerenciamento de recursos para redes elétricas inteligentes (*Smart Grids*) [88, 177].

A seguir discutimos com maior detalhamento as contribuições da tese.

## 10.1 Contribuições do trabalho

Uma das principais contribuições do trabalho é o fato da abordagem proposta permitir a implantação conjunta de múltiplas coreografias de serviços, oferecendo uma solução que contempla todas as atividades relacionadas ao gerenciamento de recursos. Além disso, a abordagem considera diferentes categorias de restrições não-funcionais e diferentes padrões de interação entre os serviços. Grande parte desses aspectos são tratados de maneira muito superficial nos demais trabalhos encontrados, o que motivou o desenvolvimento da abordagem apresentada. Esta abordagem permite preencher essa lacuna, ao mesmo tempo em que reduz o custo de utilização dos recur-

sos e também o atraso de comunicação entre os serviços. As demais contribuições do trabalho são discutidas a seguir.

- *Definição de uma nova notação para representação de coreografias de serviços:* A representação de coreografias usando a notação do grafo de processo, e sua consequente transformação na notação do grafo de dependências, permite explicitar as dependências entre os serviços e, principalmente, explorar os efeitos do compartilhamento de um mesmo serviço entre múltiplas composições. Com isso, a implantação das coreografias modeladas pode ser facilmente realizada explorando esses aspectos.

- *Proposta de um arcabouço para a especificação de restrições não-funcionais:* O arcabouço proposto abstrai a representação de restrições de QoS, propiciando que a especificação de restrições desta categoria não fique subordinada a um conjunto preestabelecido de métricas. Além disso, a linguagem proposta para representação de restrições em relação a atributos do recurso facilita a incorporação de restrições desta categoria por gerar automaticamente o código que é usado no processo de avaliação implementado pelo arcabouço.

- *Formalização do problema de síntese de recursos:* Propomos a formalização do problema da síntese de recursos como um problema de otimização e discutimos sua complexidade. A formalização proposta teve como objetivo tornar os conceitos relacionados não-ambíguos, estabelecendo uma nomenclatura padrão adotada no decorrer do trabalho, além de evidenciar a complexidade do problema.

- *Indicação de uma estratégia para estimativa de recursos:* Definimos uma estratégia de estimativa de recursos baseada em um procedimento de emparelhamento entre os serviços que compõem as coreografias e tipos canônicos que representam os recursos disponíveis. Essa estratégia considera a adição de instâncias de serviços como forma de viabilizar a satisfação de restrições em cenários de grande demanda.

- *Proposta de uma estratégia para consolidação de recursos:* Propomos uma estratégia para consolidação de recursos como forma de aproveitar possíveis capacidades excedentes nos recursos selecionados. Esta estratégia obtém como resultado adicional a redução no atraso de comunicação entre os serviços. A obtenção dessa redução é facilitada pela adoção de uma estratégia auxiliar que diminui a distância física entre os serviços implantados, considerando a troca

de tipos de recurso por aqueles que possuem melhor localidade.

- *Definição de uma arquitetura que considera todas as atividades relacionadas à implantação de coreografias de serviços:* Como forma de integrar as estratégias desenvolvidas, propomos uma arquitetura que inclui também aspectos das demais atividades relacionadas ao gerenciamento de recursos. Essa arquitetura descreve uma solução para a implantação de coreografias de serviços e para a adaptação de recursos em tempo de execução. Implementamos parte dessa arquitetura em um protótipo que considera as atividades relacionadas à implantação dos serviços.

### 10.1.1 Publicações decorrentes da tese

A pesquisa desenvolvida durante o doutorado foi iniciada tendo como foco principal o gerenciamento de recursos em nuvem para a satisfação de restrições de QoS em tempo de execução para aplicações formadas por serviços isolados. Para tal, pretendíamos desenvolver uma abordagem dinâmica baseada no uso de modelos em tempo de execução. A extensão do escopo da pesquisa para coreografias de serviços foi visando propor uma abordagem que fosse mais alinhada ao cenário atual de desenvolvimento de *software*. Com isso, identificamos também um conjunto de necessidades que deviam ser satisfeitas em tempo de implantação antes de propor uma abordagem dinâmica, o que culminou nos resultados finais da tese. Dessa forma, as primeiras publicações foram direcionadas ao aspecto dinâmico da proposta.

- GOMES, R. A.; COSTA, F. M.; NISHI, L.. **Escalabilidade Dinâmica em Nuvens Construídas a partir de Recursos Computacionais Compartilhados**. In: *XI Workshop de Computação em Clouds e Aplicações*, 2013, Brasília - DF. Anais do XI Workshop de Computação em Clouds e Aplicações, 2013.
- GOMES, R. A.; COSTA, F. M.. **Satisfying Requirements by Cloud Resources Using Models at Runtime**. In: *First Latin-American School on Software Engineering*, 2013, Rio de Janeiro. Proceedings of the First Latin-American School on Software Engineering: Basics and State-of-the-Art, 2013. v. II.

Durante todo o doutorado mantivemos colaboração com pesquisadores do *Institut National de Recherche en Informatique et en Automatique* (INRIA) – Paris, França. Como resultado dessa colaboração desenvolvemos o primeiro artigo completo. Esse trabalho descreveu a proposta inicial de maneira mais ampla:

- GOMES, R.; COSTA, F.; BENCOMO, N.. **On Modeling and Satisfaction of Non-Functional Requirements using Cloud Computing**. In: *2nd IEEE Latin Ameri-*

*can Conference on Cloud Computing and Communications*, 2013, Maceió - AL. Proceedings of the 2nd IEEE LatinCloud, 2013.

Uma vez que o gerenciamento dinâmico de recursos para a satisfação de restrições não-funcionais se manteve como um dos principais temas desenvolvidos na pesquisa, realizamos uma análise detalhada sobre o uso de estratégias de escalabilidade de recursos em nuvem visando este objetivo:

- GOMES, R.; COSTA, F.; ROCHA, R.. **Analysing Scalability Strategies for Service Choreographies on Cloud Environments**. In: Pop, Florin;Potop-Butucaru, Maria. (Org.). *Adaptive Resource Management and Scheduling for Cloud Computing*. 1ed.: Springer, 2014, v. 8907, p. 128-143.

Após avaliar os resultados obtidos com a análise das estratégias de escalabilidade, concretizamos a arquitetura proposta (apresentada no Capítulo 7) e iniciamos o desenvolvimento da abordagem de síntese de recursos. Os primeiros resultados obtidos foram com relação à formalização do modelo para representação de coreografias de serviços, que culminou na representação do grafo de dependências. O objetivo desse artigo foi enfatizar o problema de compartilhamento de serviços entre coreografias e apresentar a proposta inicial para sua solução:

- GOMES, R.; LIMA, J.; COSTA, F.; ROCHA, R.; GEORGANTAS, N.. **A Model-Based Approach for the Pragmatic Deployment of Service Choreographies**. In: Antonio Celesti; Philipp Leitner. (Org.). *Advances in Service-Oriented and Cloud Computing: Workshops of ESOCC 2015*, Taormina, Italy, September 15-17, 2015, Revised Selected Papers. 1ed.: Springer International Publishing, 2016, v. 567, p. 153-165.

Os resultados parciais relacionados à tese também foram publicados em dois *doctoral symposiums*:

- GOMES, R.; COSTA, F.. **Non-Funcional Requirements Modeling in Cloud Computing**. In: *Doctoral Symposium of the 21st IEEE International Requirements Engineering Conference*, Rio de Janeiro - RJ, 2013
- GOMES, R.; COSTA, F.; ROCHA, R.; GEORGANTAS, N.. **A Middleware-Based Approach for QoS-aware Deployment of Service Choreography in the Cloud**. In: *11th Middleware Doctoral Symposium*, 2014, Bordeaux. MDS '14 Proceedings of the 11th Middleware Doctoral Symposium, 2014.

# Referências Bibliográficas


[1] Aarts, E.; Korst, J. **Simulated annealing and Boltzmann machines**. New York, NY; John Wiley and Sons Inc., 1988.

[2] ACM. **ACM Digital Library**, 2016. http://dl.acm.org, [Online; acesso em 11-Dez-2016].

[3] Acs, S.; Zsolt, N.; Gergely, M. **A Novel Approach for Performance Characterization of IaaS Clouds**. In: *European Conference on Parallel Processing*, p. 109–120. Springer, 2014.

[4] Aiello, M.; Giorgini, P. **Applying the Tropos methodology for analysing web services requirements and reasoning about Qualities of Services**. Relatório técnico, University of Trento, 2004.

[5] Alam, S.; Chowdhury, M. M.; Noll, J. **Senaas: An event-driven sensor virtualization approach for internet of things cloud**. In: *2010 IEEE International Conference on Networked Embedded Systems for Enterprise Applications (NESEA)*, p. 1–6. IEEE, 2010.

[6] Alhamazani, K.; Ranjan, R.; Mitra, K.; Rabhi, F.; Jayaraman, P. P.; Khan, S. U.; Guabtni, A.; Bhatnagar, V. **An overview of the commercial cloud monitoring tools: research dimensions, design issues, and state-of-the-art**. *Computing*, 97(4):357–377, 2015.

[7] Alonso, G.; Casati, F.; Kuno, H.; Machiraju, V. **Web services: Concepts, Architectures and Applications**. Springer-Verlag, 2004.

[8] Alrifai, M.; Risse, T.; Nejdl, W. **A hybrid approach for efficient Web service composition with end-to-end QoS constraints**. *ACM Transactions on the Web (TWEB)*, 6(2):7, 2012.

[9] Amato, F.; Moscato, F. **Exploiting cloud and workflow patterns for the analysis of composite cloud services**. *Future Generation Computer Systems*, 67:255–265, 2017.

[10] Amazon. **Amazon EC2 Service Level Agreement**, 2016. https://aws.amazon.com/ec2/sla, [Online; acesso em 08-Dez-2016].

[11] Amazon. **Amazon Web Services - Broad & Deep Core Cloud Infrastructure Services**, 2016. http://aws.amazon.com, [Online; acesso em 07-Set-2016].

[12] Amazon. **AWS CloudFormation**, 2016. https://aws.amazon.com/cloudformation, [Online; acesso em 03-Dez-2016].

[13] Amazon. **AWS CloudWatch - Cloud & Network Monitoring Services**, 2016. http://aws.amazon.com/cloudwatch, [Online; acesso em 08-Dez-2016].

[14] Amazon. **AWS Total Cost of Ownership (TCO) Calculator**, 2016. http://awstcocalculator.com, [Online; acesso em 04-Dez-2016].

[15] Amazon. **How AWS Pricing Works**, 2016. http://aws.amazon.com/whitepapers/, [Online; acesso em 04-Dez-2016].

[16] Amazon. **Amazon EC2 FAQs**, 2017. https://aws.amazon.com/ec2/faqs/#What_is_an_EC2_Compute_Unit_and%20_why_did_you_introduce_it, [Online; acesso em 20-Jan-2017].

[17] Aotearoa. **The Multi-Goal Cloud Decision-Making Tool**, 2016. http://aotearoadecisions.appspot.com, [Online; acesso em 04-Dez-2016].

[18] Apache. **Tomcat**, 2017. http://tomcat.apache.org, [Online; acesso em 23-Jan-2017].

[19] Ardagna, D.; Di Nitto, E.; Mohagheghi, P.; Mosser, S.; Ballagny, C.; D'Andria, F.; Casale, G.; Matthews, P.; Nechifor, C.-S.; Petcu, D. **MODAClouds: A model-driven approach for the design and execution of applications on multiple clouds**. In: *Modeling in Software Engineering (MISE), 2012 ICSE Workshop on*, p. 50–56. IEEE, 2012.

[20] Armstrong, S. **Cloud Performance – Why Long Distance Relationships Don't Work**, 2014. https://www.appneta.com/blog/saas-performance-location/, [Online; acesso em 23-Dez-2016].

[21] Atzori, L.; Iera, A.; Morabito, G. **The internet of things: A survey**. *Comput Networks*, 54(15):2787–2805, 2010.

[22] Austin, D.; Barbir, A.; Ferris, C.; Garg, S. **Web services architecture requirements**. Relatório técnico, W3C Working Draft, 2002.

[23] Autili, A. D. S. M.; Di Ruscio, D.; Inverardi, P. **Synthesizing an automata-based representation of BPMN2 choreography diagrams**. *ModComp at MoDELS*, 14, 2014.

[24] Autili, M.; Inverardi, P.; Tivoli, M. **CHOReOS: large scale choreographies for the future internet**. In: *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week-IEEE Conference on*, p. 391–394. IEEE, 2014.

[25] Azure, M. **Pricing calculator**, 2016. https://azure.microsoft.com/en-us/pricing/calculator, [Online; acesso em 04-Dez-2016].

[26] Barbeau, M.; Kranakis, E. **Principles of ad-hoc networking**. John Wiley & Sons, 2007.

[27] Barker, A.; Walton, C. D.; Robertson, D. **Choreographing web services**. *IEEE Transactions on Services Computing*, 2(2):152–166, 2009.

[28] Barros, A.; Dumas, M.; Oaks, P. **A critical overview of the web services choreography description language (WS-CDL)**. *BPTrends Newsletter*, 3:1–24, 2005.

[29] Barros, A.; Dumas, M.; Ter Hofstede, A. H. **Service Interaction Patterns**. In: *International Conference on Business Process Management*, p. 302–318. Springer, 2005.

[30] Bartolini, C.; Bertolino, A.; Ciancone, A.; De Angelis, G.; Mirandola, R. **Quality Requirements for Service Choreographies**. In: *Proceedings of the 8th International Conference on Web Information Systems and Technologies (WEBIST)*, p. 143–148, Porto, Portugal, 2012. SciTePress.

[31] Bartolini, C.; Bertolino, A.; De Angelis, G.; Ciancone, A.; Mirandola, R. **Non-functional analysis of service choreographies**. In: *ICSE Workshop on Principles of Engineering Service Oriented Systems (PESOS)*, p. 8–14. IEEE, 2012.

[32] Baset, S. A. **Cloud SLAs: present and future**. *ACM SIGOPS Operating Systems Review*, 46(2):57–66, 2012.

[33] Bencomo, N.; Blair, G.; Götz, S.; Morin, B.; Rumpe, B. **Report on the 7th International Workshop on Models@run.time**. *ACM SIGSOFT Software Engineering Notes*, 38(1):27–30, 2013.

[34] Berkhin, P. **A survey of clustering data mining techniques**. In: *Grouping multidimensional data*, p. 25–71. Springer, 2006.

[35] Bernardi, S.; Merseguer, J.; Petriu, D.; Vinyals, V. **An UML profile for dependability analysis and modeling of software systems**. Relatório técnico, University of Zaragoza, 2008.

[36] Bernardi, S.; Merseguer, J.; Petriu, D. C. **Adding dependability analysis capabilities to the MARTE profile**. In: *International Conference on Model Driven Engineering Languages and Systems*, p. 736–750. Springer, 2008.

[37] Bertolino, A.; De Angelis, G.; Polini, A.; Sabetta, A. **Trends and research issues in SOA validation**. *Performance and Dependability in Service Computing: Concepts, Techniques and Research Directions*, p. 98, 2011.

[38] Bi, J.; Zhu, Z.; Tian, R.; Wang, Q. **Dynamic provisioning modeling for virtualized multi-tier applications in cloud data center**. In: *2010 IEEE 3rd International Conference on Cloud Computing*, p. 370–377. IEEE, 2010.

[39] Blair, G.; Bencomo, N.; France, R. B. **Models@run.time**. *Computer*, 42(10):22–27, 2009.

[40] Blair, G.; Kon, F.; Cirne, W.; Milojicic, D.; Ramakrishnan, R.; Reed, D.; Silva, D. **Perspectives on cloud computing: interviews with five leading scientists from the cloud community**. *Journal of Internet Services and Applications*, 2(1):3–9, 2011.

[41] Bocciarelli, P.; D'Ambrogio, A. **A BPMN extension for modeling non functional properties of business processes**. In: *Proceedings of the 2011 Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, p. 160–168. Society for Computer Simulation International, 2011.

[42] Boehm, B. W.; Brown, J. R.; Kaspar, H.; Lipow, M.; MacLeod, G. J.; Merrit, M. J. **Characteristics of software quality**, volume 1. North-Holland Publishing Company, 1978.

[43] Bouloukakis, G.; Georgantas, N.; Kattepur, A.; Issarny, V. **Timeliness Evaluation of Intermittent Mobile Connectivity over Pub/Sub Systems**. In: *8th ACM/SPEC International Conference on Performance Engineering (ICPE)*, 2017.

[44] Buco, M. J.; Chang, R. N.; Luan, L. Z.; Ward, C.; Wolf, J. L.; Yu, P. S. **Utility computing SLA management based upon business objectives**. *IBM Systems Journal*, 43(1):159–178, 2004.

[45] Burke, P. J. **The output of a queuing system**. *Operations research*, 4(6):699–704, 1956.

[46] Buschmann, F.; Henney, K.; Schimdt, D. **Pattern-oriented Software Architecture: On Patterns and Pattern Language**, volume 5. John Wiley & Sons, 2007.

[47] Buyya, R.; Ranjan, R.; Calheiros, R. **Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities**. In: *International Conference on High Performance Computing Simulation*, p. 1–11, June 2009.

[48] Buyya, R.; Beloglazov, A.; Abawajy, J. H. **Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges**. *CoRR*, abs/1006.0308, 2010.

[49] Buyya, R.; Ranjan, R.; Calheiros, R. N. **Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services**. In: *Algorithms and architectures for parallel processing*, p. 13–31. Springer, 2010.

[50] Buyya, R.; Yeo, C. S.; Venugopal, S.; Broberg, J.; Brandic, I. **Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility**. *Future Generation Computer Systems*, 25(6):599–616, 2009.

[51] Calheiros, R. N.; Ranjan, R.; Beloglazov, A.; De Rose, C. A.; Buyya, R. **CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms**. *Software: Practice and Experience*, 41(1):23–50, 2011.

[52] Canonical. **Ubuntu: The leading operating system for PCs, tablets, phones, IoT devices, servers and the cloud**, 2017. https://www.ubuntu.com, [Online; acesso em 23-Jan-2017].

[53] Casa Civil. **Lei do Marco Civil da Internet no Brasil**, 2014. http://www.cgi.br/lei-do-marco-civil-da-internet-no-brasil/, [Online; acesso em 02-Dez-2016].

[54] Cavalcante, E.; Batista, T.; Lopes, F.; Delicato, F. C.; Pires, P. F.; Rodriguez, N.; de Moura, A. L.; Mendes, R. **Optimizing services selection in a cloud multiplatform scenario**. In: *2012 IEEE Latin*

*America Conference on Cloud Computing and Communications (LATINCLOUD)*, p. 31–36. IEEE, 2012.

[55] Charrada, F. B.; Tebourski, N.; Tata, S.; Moalla, S. **Approximate placement of service-based applications in hybrid clouds**. In: *2012 IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, p. 161–166, June 2012.

[56] Chauvel, F.; Ferry, N.; Morin, B.; Rossini, A.; Solberg, A. **Models@Runtime to Support the Iterative and Continuous Design of Autonomic Reasoners**. In: *Proceedings of the 8th International Workshop on Models at Run.time*. ACM, 2013.

[57] CHEF. **Achieve speed, scale, and consistency by automating your infrastructure with Chef**, 2016. https://www.chef.io/chef, [Online; acesso em 07-Set-2016].

[58] Chiba, S.; Nishizawa, M. **An easy-to-use toolkit for efficient Java bytecode translators**. In: *International Conference on Generative Programming and Component Engineering*, p. 364–376. Springer, 2003.

[59] CHOReOS. **Enactment Engine Documentation**, 2014. http://www.choreos.eu/bin/view/Documentation/enactment_engine_doc, [Online; acesso em 07-Set-2016].

[60] CHOReOS. **Large Scale Choreographies for the Future Internet**, 2016. http://www.choreos.eu, [Online; acesso em 08-Dez-2016].

[61] Chung, L.; do Prado Leite, J. C. S. **On non-functional requirements in software engineering**. In: *Conceptual modeling: Foundations and applications*, p. 363–379. Springer, 2009.

[62] Chung, L.; Nixon, B. A.; Yu, E.; Mylopoulos, J. **Non-functional requirements in software engineering**, volume 5. Springer Science & Business Media, 2012.

[63] Ciancone, A.; Filieri, A.; Drago, M. L.; Mirandola, R.; Grassi, V. **KlaperSuite: an integrated model-driven environment for reliability and performance analysis of component-based systems**. In: *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, p. 99–114. Springer, 2011.

[64] Clark, J.; Casanave, C.; Kanaskie, K.; Harvey, B.; Smith, N.; Yunker, J.; Riemer, K. **ebXML Business Process Specification Schema Version 1.01**. *UN/CEFACT and OASIS*, p. 9, 2001.

[65] CloudHarmony. **Transparence for the cloud**, 2016. https://cloudharmony.com, [Online; acesso em 06-Dez-2016].

[66] Collberg, C.; Myles, G.; Stepp, M. **An empirical study of Java bytecode programs**. *Software-Practice and Experience*, 37(6):581–642, 2007.

[67] Computing, A.; others. **An architectural blueprint for autonomic computing**. *IBM White Paper*, 2006.

[68] Correia, I.; Gouveia, L.; da Gama, F. S. **Solving the variable size bin packing problem with discretized formulations**. *Computers & Operations Research*, 35(6):2103 – 2113, 2008. Part Special Issue: {OR} Applications in the Military and in Counter-Terrorism.

[69] CSMIC. **Service Measurement Index Framework Version 2.1**, 2014. https://spark.adobe.com/page/PN39b, [Online; acesso em 04-Dez-2016].

[70] Daras, P.; Williams, D.; Guerrero, C.; Kegel, I.; Laso, I.; Bouwen, J.; Meunier, J.; Niebert, N.; Zahariadis, T. **Why do we need a content-centric future Internet? Proposals towards content-centric Internet architectures**. *Information Society and Media Journal*, 2009.

[71] de Camargo, R.; Goldchleger, A.; Carneiro, M.; Kon, F. **Grid: An Architectural Pattern**. In: *Proceedings of the 11th Conference on Pattern Languages of Programs*, 2004.

[72] de Mello, R. F.; Yang, L. T. **Prediction of dynamical, nonlinear, and unstable process behavior**. *The Journal of Supercomputing*, 49(1):22–41, 2009.

[73] Decker, G.; Kopp, O.; Barros, A. **An Introduction to Service Choreographies**. *Information Technology*, 50(2/2008):122–127, 2008.

[74] Decker, G.; Kopp, O.; Leymann, F.; Pfitzner, K.; Weske, M. **Modeling service choreographies using BPMN and BPEL4Chor**. In: *International conference on Advanced Information Systems Engineering*, p. 79–93. Springer, 2008.

[75] Decker, G.; Kopp, O.; Leymann, F.; Weske, M. **BPEL4Chor: Extending BPEL for modeling choreographies**. In: *IEEE International Conference on Web Services*, p. 296–303. IEEE, 2007.

[76] Deng, Y.; Masoud Sadjadi, S.; Clarke, P. J.; Hristidis, V.; Rangaswami, R.; Wang, Y. **CVM – A communication virtual machine**. *Journal of Systems and Software*, 81(10):1640–1662, 2008.

[77] Di Costanzo, A.; De Assuncao, M. D.; Buyya, R. **Harnessing cloud technologies for a virtualized distributed computing infrastructure**. *internet Computing, IEEE*, 13(5):24–33, 2009.

[78] Di Marco, A.; Pompilio, C.; Bertolino, A.; Calabrò, A.; Lonetti, F.; Sabetta, A. **Yet another meta-model to specify non-functional properties**. In: *Proceedings of the International Workshop on Quality Assurance for Service-Based Applications*, p. 9–16. ACM, 2011.

[79] Dillon, T.; Wu, C.; Chang, E. **Cloud computing: issues and challenges**. In: *24th IEEE International Conference on Advanced Information Networking and Applications*, p. 27–33. IEEE, 2010.

[80] Dolstra, E.; Bravenboer, M.; Visser, E. **Service configuration management**. In: *Proceedings of the 12th international workshop on Software configuration management*, p. 83–98. ACM, 2005.

[81] Duato, J.; Pena, A. J.; Silla, F.; Fernandez, J. C.; Mayo, R.; Quintana-Orti, E. S. **Enabling CUDA acceleration within virtual machines using rCUDA**. In: *18th International Conference on High Performance Computing (HiPC)*, p. 1–10. IEEE, 2011.

[82] Eisa, M.; Younas, M.; Basu, K.; Zhu, H. **Trends and Directions in Cloud Service Selection**. In: *2016 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, p. 423–432, March 2016.

[83] ej-techonologies. **JProfiler – The Award-Winning All-in-One Java Profiler**, 2016. http://www.ej-technologies.com/products/jprofiler/overview.html, [Online; acesso em 01-Nov-2016].

[84] Elsevier. **Elsevier's Scopus**, 2016. https://www.scopus.com, [Online; acesso em 11-Dez-2016].

[85] Emerson, E. A. **Temporal and modal logic.** *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, 995(1072):5, 1990.

[86] Engler, L. **BPEL gold: Choreography on the Service Bus**. Tese de doutorado, University of Stuttgart, 2009.

[87] European Commission. **Open source API and plataform for multiple clouds**, 2016. http://www.mosaic-cloud.eu, [Online; acesso em 08-Dez-2016].

[88] Farhangi, H. **The Path of the Smart Grid**. *IEEE Power and Energy Magazine*, 8(1):18–28, January 2010.

[89] Ferry, N.; Chauvel, F.; Rossini, A.; Morin, B.; Solberg, A. **Managing multi-cloud systems with CloudMF**. In: *Proceedings of the Second Nordic Symposium on Cloud Computing & Internet Technologies*, p. 38–45. ACM, 2013.

[90] Ferry, N.; Rossini, A.; Chauvel, F.; Morin, B.; Solberg, A. **Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems**. In: *CLOUD 2013: IEEE 6th International Conference on Cloud Computing*, p. 887–894, 2013.

[91] Fielding, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. Tese de doutorado, University of California, 2000.

[92] Flügge, M.; Tourtchaninova, D. **Ontology-derived Activity Components for Composing Travel Web Services**. In: *Berliner XML Tage*, p. 133–150, 2004.

[93] Foster, H.; Uchitel, S.; Kramer, J. M. J. **Model-Based Analysis of Obligations in Web Service Choreography**. In: *Advanced International Conference on Telecommunications / International Conference on Internet and Web Applications and Services*, volume 00, p. 149, 2006.

[94] Foster, I.; Kesselman, C.; Nick, J.; Tuecke, S. **The Physiology of the Grid**. *An Open Grid Services Architecture for Distributed Systems Integration*, 2002.

[95] Foster, I.; Kesselman, C.; Tuecke, S. **The Anatomy of the Grid**. *International Journal of Supercomputer Applications*, 15(3):200–222, 2001.

[96] Foster, I.; Zhao, Y.; Raicu, I.; Lu, S. **Cloud computing and grid computing 360-degree compared**. In: *Grid Computing Environments Workshop*, p. 1–10. IEEE, 2008.

[97] Fowler, M. **Patterns of Enterprise Application Architecture**. Addison-Wesley Professional, 2002.

[98] Friesen, D. K.; Langston, M. A. **Variable sized bin packing**. *Society for Industrial and Applied Mathematis Journal on Computing*, 15(1):222–230, 1986.

[99] Garey, M. R.; Johnson, D. S. **Computers and Intractibility: A Guide to the Theory of NP-Completeness**. Freeman, 1979.

[100] Garg, S. K.; Versteeg, S.; Buyya, R. **SMICloud: A framework for comparing and ranking cloud services**. In: *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, p. 210–218. IEEE, 2011.

[101] Garg, S. K.; Versteeg, S.; Buyya, R. **A framework for ranking of cloud computing services**. *Future Generation Computer Systems*, 29(4):1012–1023, 2013.

[102] Garg, S.; Yeo, C.; Anandasivam, A.; Buyya, R. **Environment-conscious scheduling of HPC applications on distributed cloud-oriented data centers**. *Journal of Parallel Distributed Computing*, 71(6):732–749, 2011.

[103] Gargoyle Software Inc.. **HtmlUnit**, 2016. http://htmlunit.sourceforge.net, [Online; acesso em 15-Nov-2016].

[104] Georgantas, N.; Bouloukakis, G.; Beauche, S.; Issarny, V. **Service-oriented Distributed Applications in the Future Internet: The Case for Interaction Paradigm Interoperability**. In: *European Conference on Service-Oriented and Cloud Computing (ESOCC)*, 2013.

[105] Giese, H.; Wagner, R. **Incremental model synchronization with triple graph grammars**. In: *Model Driven Engineering Languages and Systems*, p. 543–557. Springer, 2006.

[106] Goldchleger, A. **InteGrade: Um Sistema de Middleware para Computação em Grade Oportunista**. Dissertação de mestrado, Department of Computer Science - University of São Paulo, Dec. 2004.

[107] Goldman, A.; Ngoko, Y.; Milojicic, D. **An analytical approach for predicting QoS of web services choreographies**. In: *Proceedings of the 10th International Workshop on Middleware for Grids, Clouds and e-Science*, p. 4. ACM, 2012.

[108] Gomes, R.; Costa, F.; Bencomo, N. **On modeling and satisfaction of non-functional requirements using cloud computing**. In: *2nd IEEE Latin American Conference on Cloud Computing and Communications*, p. 1–6, Dec 2013.

[109] Gomes, R.; Lima, J.; Costa, F.; da Rocha, R.; Georgantas, N. **A Model-Based Approach for the Pragmatic Deployment of Service Choreographies**, p. 153–165. Springer International Publishing, Cham, 2016.

[110] Google. **Google App Engine: Platform as a Service**, 2016. https://developers.google.com/appengine, [Online; acesso em 08-Dez-2016].

[111] Google. **Google Cloud Platform Compute Engine**, 2016. https://cloud.google.com/compute, [Online; acesso em 07-Set-2016].

[112] Google. **Google Drive**, 2016. https://drive.google.com, [Online; acesso em 08-Dez-2016].

[113] Google. **Google Scholar**, 2016. https://scholar.google.com.br, [Online; acesso em 11-Dez-2016].

[114] Google. **Creating an Instance with a Custom Machine Type**, 2017. https://cloud.google.com/compute/docs/instances/creating-instance-with-custom-machine-type, [Online; acesso em 02-Jan-2017].

[115] Google. **Google Maps Geocoding API**, 2017. https://developers.google.com/maps/documentation/geocoding/intro, [Online; acesso em 13-Jan-2017].

[116] Goudarzi, H.; Ghasemazar, M.; Pedram, M. **SLA-based optimization of power and migration cost in cloud computing**. In: *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, p. 172–179. IEEE, 2012.

[117] GRAPHIQ. **Compare Cloud Computing Providers**, 2016. http://cloud-computing.softwareinsider.com, [Online; acesso em 13-Dez-2016].

[118] Grozev, N.; Buyya, R. **Performance modelling and simulation of three-tier applications in cloud and multi-cloud environments**. *The Computer Journal*, 58(1):1–22, 2015.

[119] Gudgin, M.; Hadley, M.; Mendelsohn, N.; Moreau, J.-J.; Nielsen, H. F.; Karmarkar, A.; Lafon, Y. **SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)**. Relatório técnico, W3C Recommendation, 2007.

[120] Guttmann-Beck, N.; Hassin, R. **Approximation algorithms for minimum k-cut**. *Algorithmica*, 27(2):198–207, 2000.

[121] Haeser, G.; Ruggiero, M. G. **Aspectos teóricos de simulated annealing e um algoritmo duas fases em otimização global**. *Trends in Applied and Computational Mathematics*, 9(3):395–404, 2008.

[122] Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; Witten, I. H. **The WEKA data mining software: an update**. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[123] Hartigan, J. A.; Wong, M. A. **Algorithm AS 136: A k-means clustering algorithm**. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[124] Hill, Z.; Li, J.; Mao, M.; Ruiz-Alvarez, A.; Humphrey, M. **Early observations on the performance of Windows Azure**. In: *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, p. 367–376. ACM, 2010.

[125] Hofreiter, B.; Huemer, C. **A model-driven top-down approach to inter-organizational systems: From global choreography models to executable BPEL**. In: *10th IEEE Conference on E-Commerce Technology and the 5th IEEE Conference on Enterprise Computing, E-Commerce and E-Services*, p. 136–145. IEEE, 2008.

[126] Hooper, A. **Green computing**. *Communication of the ACM*, 51(10):11–13, 2008.

[127] Huang, K.-C.; Shen, B.-J. **Service deployment strategies for efficient execution of composite SaaS applications on cloud platform**. *Journal of Systems and Software*, 107:127–141, 2015.

[128] Humble, J.; Farley, D. **Continuous Delivery**. Addison-Wesley, 2011.

[129] Hummaida, A. R.; Paton, N. W.; Sakellariou, R. **Adaptation in Cloud Resource Configuration: A Survey**. *Journal of Cloud Computing*, 5(1):1–16, 2016.

[130] Idera. **Uptime Cloud Monitor**, 2016. https://www.idera.com/infrastructure-monitoring-as-a-service, [Online; acesso em 09-Dez-2016].

[131] IEEE. **IEEE Xplore Digital Library**, 2016. http://ieeexplore.ieee.org/Xplore, [Online; acesso em 11-Dez-2016].

[132] Iosup, A.; Ostermann, S.; Yigitbasi, M. N.; Prodan, R.; Fahringer, T.; Epema, D. **Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing**. *IEEE Transactions on Parallel and Distributed Systems*, 22(6):931–945, June 2011.

[133] Iosup, A.; Yigitbasi, N.; Epema, D. **On the performance variability of production cloud services**. In: *Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, p. 104–113. IEEE, 2011.

[134] ISO. **Country Codes - ISO 3166**, 2017. http://www.iso.org/iso/home/standards/country_codes.htm, [Online; acesso em 13-Jan-2017].

[135] Issarny, V.; Georgantas, N.; Hachem, S.; Zarras, A.; Vassiliadist, P.; Autili, M.; Gerosa, M. A.; Hamida, A. B. **Service-oriented middleware for the Future Internet: state of the art and research directions**. *Journal of Internet Services and Applications*, 2(1):23–45, 2011.

[136] ITU-T. **Message Sequence Chart. Recommendation Z.120**. Relatório técnico, ITU-T, 2000.

[137] Javadi, B.; Abawajy, J.; Buyya, R. **Failure-aware resource provisioning for hybrid Cloud infrastructure**. *J Parallel Distrib Comput*, 72(10):1318–1331, 2012.

[138] Jeffers, J.; Reinders, J. **Intel Xeon Phi coprocessor high-performance programming**. Newnes, 2013.

[139] Jensen, F. V. **An introduction to Bayesian networks**, volume 210. UCL press London, 1996.

[140] Jordan, D.; Evdemon, J.; Alves, A.; Arkin, A.; Askary, S.; Barreto, C.; Bloch, B.; Curbera, F.; Ford, M.; Goland, Y. **Web services business process execution language version 2.0**. *OASIS standard*, 11:11, 2007.

[141] Juju. **Model-driven operations for hybrid cloud services**, 2016. https://jujucharms.com, [Online; acesso em 04-Dez-2016].

[142] Junior, R. M. **Diretrizes relacionadas à segurança da informação e comunicações para o uso de computação em nuvem nos órgãos e entidades da administração pública federal**. *Norma Complementar 14/IN01/DSIC/GSIPR, Departamento de Segurança da Informação e Comunicações*, 2012.

[143] Kang, J.; Park, S. **Algorithms for the variable sized bin packing problem**. *European Journal of Operational Research*, 147(2):365 – 372, 2003. Fuzzy Sets in Scheduling and Planning.

[144] Kattepur, A.; Georgantas, N.; Issarny, V. **QoS composition and analysis in reconfigurable web services choreographies**. In: *20th International Conference on Web Services (ICWS)*, p. 235–242. IEEE, 2013.

[145] Kaufman, L. M. **Can Public-Cloud Security Meet Its Unique Challenges?** *IEEE Security Privacy*, 8(4):55–57, July 2010.

[146] Kavantzas, N.; Burdett, D.; Ritzinger, G.; Fletcher, T.; Lafon, Y.; Barreto, C. **Web services choreography description language version 1.0**. *W3C candidate recommendation*, 9, 2005.

[147] Keahey, K. **Nimbus: Open Source Infrastructure-as-a-Service Cloud Computing Software**. *Workshop on adapting applications and computing services to multi-core and virtualization*, 2009.

[148] Keller, S. E.; Kahn, L. G.; Panara, R. B. **Specifying software quality requirements with metrics**. *System and Software Requirements Engineering*, p. 145–163, 1990.

[149] Khaluf, L.; Gerth, C.; Engels, G. **Pattern-based modeling and formalizing of business process quality constraints**. In: *International Conference on Advanced Information Systems Engineering*, p. 521–535. Springer, 2011.

[150] Khan, S. **Quality adaptation in a multi-session adaptive multimedia system: model and architecture**. Tese de doutorado, Department of Electronical and Computer Engineering, University of Victoria, 1998.

[151] Kim, H.; El-Khamra, Y.; Rodero, I.; Jha, S.; Parashar, M. **Autonomic management of application workflows on hybrid computing infrastructure**. *Scientific Programming*, 19(2):75–89, 2011.

[152] Knuth, D. E. **The Art of Computer Programming**, volume 3, chapter Generating All Tuples and Permutations, p. 426–458. Upper Saddle River, 1999.

[153] Kopp, O.; Binz, T.; Breitenbücher, U.; Leymann, F. **BPMN4TOSCA: A domain-specific language to model management plans for composite applications**. In: *International Workshop on Business Process Modeling Notation*, p. 38–52. Springer, 2012.

[154] Krutz, R. L.; Vines, R. D. **Cloud security: A comprehensive guide to secure cloud computing**. Wiley Publishing, 2010.

[155] Laboratory, C. **InterGrid: Internetworking Islands Of Grids**, 2016. http://www.cloudbus.org/intergrid, [Online; acesso em 08-Dez-2016].

[156] Lambert, D. M.; Cooper, M. C. **Issues in supply chain management**. *Industrial marketing management*, 29(1):65–83, 2000.

[157] Lawton, G. **Developing Software Online With Platform-as-a-Service Technology**. *Computer*, 41(6):13–15, June 2008.

[158] Lazowska, E. D.; Zahorjan, J.; Graham, G. S.; Sevcik, K. C. **Quantitative system performance: computer system analysis using queueing network models**. Prentice-Hall, Inc., 1984.

[159] Leite, L.; Lago, N.; Gerosa, M. A.; Kon, F. **Um Middleware para Encenação Automatizada de Coreografias de Serviços Web em Ambientes de Computação em Nuvem**. In: *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2013)*, p. 1–14, 2013.

[160] Leite, L. A.; Oliva, G. A.; Nogueira, G. M.; Gerosa, M. A.; Kon, F.; Milojicic, D. S. **A systematic literature review of service choreography adaptation**. *Service Oriented Computing and Applications*, p. 1–18, 2012.

[161] Leite, L. A. F. **Implantação automatizada de composições de serviços web de grande escala**. Dissertação de mestrado, Universidade de São Paulo, 2014.

[162] Lenk, A.; Menzel, M.; Lipsky, J.; Tai, S.; Offermann, P. **What Are You Paying For? Performance Benchmarking for Infrastructure-as-a-Service Offerings**. In: *2011 IEEE 4th International Conference on Cloud Computing*, p. 484–491, July 2011.

[163] Leong, L.; Petri, G.; Gill, B.; Dorosh, M. **Gartner: Magic Quadrant for Cloud Infrastructure as a Service, Worldwide**, 2016. https://www.gartner.com/doc/3400818/magic-quadrant-cloud-infrastructure-service, [Online; acesso em 20-Ago-2016].

[164] Lewis, D. **What is Web 2.0?** *Crossroads*, 13(1):3–3, Sept. 2006.

[165] Leymann, F. **Web Services Flow Language (WSFL 1.0)**. *IBM Corporation*, 2001. Disponível em: http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf.

[166] Li, A.; Yang, X.; Kandula, S.; Zhang, M. **CloudCmp: comparing public cloud providers**. In: *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, p. 1–14. ACM, 2010.

[167] Liu, C.; Mao, Y.; Van der Merwe, J.; Fernandez, M. **Cloud resource orchestration: A data-centric approach**. In: *Proceedings of the biennial Conference on Innovative Data Systems Research (CIDR)*, p. 1–8. Citeseer, 2011.

[168] Liu, Y.; Ngu, A. H.; Zeng, L. Z. **QoS computation and policing in dynamic web service selection**. In: *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, p. 66–73. ACM, 2004.

[169] Luckham, D. C.; Frasca, B. **Complex event processing in distributed systems**. *Computer Systems Laboratory Technical Report CSL-TR-98-754. Stanford University, Stanford*, 28, 1998.

[170] Maiden, N.; Lockerbie, J.; Zachos, K.; Bertolino, A.; De Angelis, G.; Lonetti, F. **A requirements-led approach for specifying QoS-aware service choreographies: An experience report**. In: *International Working Conference on Requirements Engineering: Foundation for Software Quality*, p. 239–253. Springer, 2014.

[171] Malizia, N. **How To Build A Boolean Expression Evaluator**, 2017. https://unnikked.ga/how-to-build-a-boolean-expression-evaluator-518e9e068a65#.1rx0ze323, [Online; acesso em 06-Fev-2017].

[172] Mani, A.; Nagarajan, A. **Understanding quality of service for Web services**. Relatório técnico, IBM, 2005.

[173] Manvi, S. S.; Shyam, G. K. **Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey**. *Journal of Network and Computer Applications*, 41:424–440, 2014.

[174] Mao, M.; Humphrey, M. **A Performance Study on the VM Startup Time in the Cloud**. In: *2012 IEEE Fifth International Conference on Cloud Computing*, p. 423–430, June 2012.

[175] Mao, Z.; Yang, J.; Shang, Y.; Liu, C.; Chen, J. **A game theory of cloud service deployment**. In: *2013 IEEE Ninth World Congress on Services*, p. 436–443, June 2013.

[176] Martin, P.; Soltani, S.; Powley, W.; Hassannezhad, M. **QoS-Aware Cloud Application Management**. *Cloud Computing and Big Data*, 23:20, 2013.

[177] Masters, G. M. **Renewable and Efficient Electric Power Systems**. John Wiley & Sons, 2013.

[178] Mell, P.; Grance, T. **Draft NIST working definition of cloud computing**. Relatório técnico, National Institute of Standard and Technology (NIST), 2009. Disponível em: https://www.nist.gov/sites/default/files/documents/itl/cloud/cloud-def-v15.pdf.

[179] Mendling, J.; Lassen, K. B.; Zdun, U.; others. **Transformation strategies between block-oriented and graph-oriented process modelling languages**. In: *Multikonferenz Wirtschaftsinformatik*, volume 2, p. 297–312. unknown, 2006.

[180] Menzel, M.; Ranjan, R. **CloudGenius: decision support for web server cloud migration**. In: *Proceedings of the 21st international conference on World Wide Web*, p. 979–988. ACM, 2012.

[181] Menzel, M.; Schönherr, M.; Tai, S. **(MC$^2$)$^2$: criteria, requirements and a software prototype for Cloud infrastructure decisions**. *Software: Practice and experience*, 43(11):1283–1297, 2013.

[182] Microsoft. **Microsoft Azure**, 2016. https://azure.microsoft.com, [Online; accessed 07-Set-2016].

[183] Microsoft. **Sizes for Windows virtual machines in Azure**, 2017. https://docs.microsoft.com/en-us/azure/virtual-machines/virtual-machines-windows-sizes, [Online; acesso em 20-Jan-2017].

[184] Milchtaich, I. **Congestion games with player-specific payoff functions**. *Games and economic behavior*, 13(1):111–124, 1996.

[185] MODAClouds. **MOdel-Driven Approach for design and execution of applications on multiple Clouds**, 2016. http://www.modaclouds.eu, [Online; acesso em 08-Dez-2016].

[186] Moscato, F.; Aversa, R.; Di Martino, B.; Fortiş, T.-F.; Munteanu, V. **An analysis of mOSAIC ontology for cloud resources annotation**. In: *Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS)*, p. 973–980. IEEE, 2011.

[187] Moser, M.; Jokanovic, D. P.; Shiratori, N. **An algorithm for the multidimensional multiple-choice knapsack problem**. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 80(3):582–589, 1997.

[188] Mostow, J. **Toward better models of the design process**. *AI magazine*, 6(1):44, 1985.

[189] Newman, S. **Building Microservices**. O'Reilly Media, Inc., 2015.

[190] Ngan, L. D.; Kanagasabai, R. **OWL-S based semantic cloud service broker**. In: *2012 IEEE 19th International Conference on Web Services (ICWS)*, p. 560–567. IEEE, 2012.

[191] Nielsen, J. **Powers of 10: Time scales in user experience**. *Alertbox*, 5, 2009. Disponível em: http://www.useit.com/alertbox/timeframes.html.

[192] Novakovic, D. M.; Vasic, N.; Novakovic, S.; Kostic, D.; Bianchini, R. **Deepdive: Transparently identifying and managing performance interference in virtualized environments.** In: *USENIX Annual Technical Conference*, p. 219–230, 2013.

[193] NVIDIA Corporation. **NVIDIA's next generation CUDA compute architecture: Kepler GK110**. *Whitepaper*, 2012.

[194] OASIS. **Topology and Orchestration Specification for Cloud Applications Version 1.0 Working Draft 07**, 2012. https://www.oasis-open.org/committees/download.php/46274/TOSCA-v1.0-wd07.zip, [Online; acesso em 03-Dez-2016].

[195] Oikawa, M.; Kawai, A.; Nomura, K.; Yasuoka, K.; Yoshikawa, K.; Narumi, T. **DS-CUDA: a middleware to use many GPUs in the cloud environment**. In: *High Performance Computing, Networking, Storage and Analysis (SCC)*, p. 1207–1214. IEEE, 2012.

[196] OLoughlin, J.; Gillam, L. **Towards Performance Prediction for Public Infrastructure Clouds: An EC2 Case Study**. In: *2013 IEEE 5th International*

*Conference on Cloud Computing Technology and Science*, volume 1, p. 475–480, Dec 2013.

[197] OMG. **A UML profile for modeling and analysis of real time embedded systems**. *v. 1.0*, 2009.

[198] OMG. **Business Process Model and Notation (BPMN) Version 2.0**. *Object Management Group*, 2011.

[199] OpenStack. **Open source software for creating private and public clouds**, 2016. https://www.openstack.org, [Online; acesso em 03-Dez-2016].

[200] Oracle. **The Java Tutorials**, 2017. http://docs.oracle.com/javase/tutorial/essential/concurrency, [Online; acesso em 26-Jan-2017].

[201] Ostermann, S.; Iosup, A.; Yigitbasi, N.; Prodan, R.; Fahringer, T.; Epema, D. **A performance analysis of EC2 cloud computing services for scientific computing**. In: *Cloud computing*, p. 115–131. Springer, 2009.

[202] Ou, Z.; Zhuang, H.; Nurminem, J. K.; Yla-Jaaski, A.; Hui, P. **Exploiting Hardware Heterogeneity within the same instance type of Amazon EC2**. In: *Proceedings of the 4th USENIX Workshop on Hot Topics in Cloud Computing*, Boston, MA, June 2012.

[203] Pahl, C. **Containerization and the PaaS Cloud**. *IEEE Cloud Computing*, 2(3):24–31, 2015.

[204] Papadimitriou, D.; others. **Future Internet – The Cross-ETP Vision Document**. *European Technology Platform, Alcatel Lucent*, 8, 2009.

[205] Papazoglou, M. P.; Traverso, P.; Dustdar, S.; Leymann, F. **Service-oriented computing: State of the art and research challenges**. *Computer*, 40(11):38–45, 2007.

[206] Parliament, E. **Directive 95/46/EC of the European Parliament and of the Council**, 2016. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:en:HTML, [Online; acesso em 02-Dez-2016].

[207] Parwekar, P. **From Internet of Things Towards Cloud of Things**. In: *Proceedings of the 2nd International Conference on Computer and Communication Technology (ICCCT)*, p. 329–333. IEEE, 2011.

[208] Pavlovski, C. J.; Zou, J. **Non-functional requirements in business process modeling**. In: *Proceedings of the 5th Asia-Pacific conference on*

*Conceptual Modelling*, volume 79, p. 103–112. Australian Computer Society, Inc., 2008.

[209] Pearson, S. **Taking account of privacy when designing cloud computing services**. In: *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, p. 44–52. IEEE Computer Society, 2009.

[210] Pelleg, D.; Moore, A. **X-means: Extending K-means with Efficient Estimation of the Number of Clusters**. In: *ICML*, volume 1, 2000.

[211] Peltz, C. **Web services orchestration and choreography**. *Computer*, 36(10):46–52, 2003.

[212] Petcu, D.; Di Martino, B.; Venticinque, S.; Rak, M.; Máhr, T.; Lopez, G. E.; Brito, F.; Cossu, R.; Stopar, M.; Stankovski, V.; others. **Experiences in building a mosaic of clouds**. *Journal of Cloud Computing: Advances, Systems and Applications*, 2(1):12, 2013.

[213] Phillips, S. C.; Engen, V.; Papay, J. **Snow White Clouds and the Seven Dwarfs**. In: *2011 IEEE Third International Conference on Cloud Computing Technology and Science*, p. 738–745, Nov 2011.

[214] Ponnekanti, S. R.; Fox, A. **Sword: A developer toolkit for web service composition**. In: *Proceedings of the 11th International World Wide Web Conference*, Honolulu, HI, 2002.

[215] Poulin, M. **Collaboration patterns in the SOA ecosystem**. In: *Proceedings of the Third Workshop on Behavioural Modelling*, p. 12–16. ACM, 2011.

[216] Puppet. **Marionette Collective**, 2016. https://docs.puppet.com/mcollective, [Online; acesso em 03-Dez-2016].

[217] Puppet. **The Puppet approach**, 2016. https://puppet.com/product/how-puppet-works, [Online; acesso em 03-Dez-2016].

[218] Qiu, Z.; Zhao, X.; Cai, C.; Yang, H. **Towards the theoretical foundation of choreography**. In: *Proceedings of the 16th international conference on World Wide Web*, p. 973–982. ACM, 2007.

[219] Quinton, C.; Romero, D.; Duchien, L. **Automated selection and configuration of cloud environments using software product lines principles**. In: *2014 IEEE 7th International Conference on Cloud Computing*, p. 144–151. IEEE, 2014.

[220] Rackspace. **Rackspace Public Cloud**, 2016. https://www.rackspace.com/cloud, [Online; acesso em 07-Set-2016].

[221] Rahman, M.; Ranjan, R.; Buyya, R.; Benatallah, B. **A taxonomy and survey on autonomic management of applications in grid computing environments**. *Concurrency and Computation: Practice and Experience*, 2011.

[222] Ray, M. **Spiceweasel description**, 2016. https://github.com/mattray/spiceweasel, [Online; acesso em 03-Dez-2016].

[223] Reig, G.; Alonso, J.; Guitart, J. **Prediction of job resource requirements for deadline schedulers to manage high-level SLAs on the cloud**. In: *Proceedings of the 9th IEEE International Symposium on Network Computing and Applications (NCA)*, p. 162–167. IEEE, 2010.

[224] Reuters, T. **Web of Science**, 2016. https://webofknowledge.com, [Online; acesso em 11-Dez-2016].

[225] Rhee, W. T.; Talagrand, M. **Martingale inequalities and NP-complete problems**. *Mathematics of Operations Research*, 12(1):177–181, 1987.

[226] RightScale. **PlanForCloud – Cloud Cost Calculator**, 2016. http://www.planforcloud.com, [Online; acesso em 04-Dez-2016].

[227] RightScale. **Universal Cloud Management Platform**, 2016. http://www.rightscale.com, [Online; acesso em 03-Dez-2016].

[228] Roman, G.-C. **A taxonomy of current issues in requirements engineering**. *Computer*, 18(4):14–23, 1985.

[229] Rosario, S.; Benveniste, A.; Jard, C. **Flexible probabilistic QoS management of transaction based web services orchestrations**. In: *IEEE International Conference on Web Services*, p. 107–114. IEEE, 2009.

[230] Rosenberg, F.; Enzi, C.; Michlmayr, A.; Platzer, C.; Dustdar, S. **Integrating Quality of Service Aspects in Top-Down Business Process Development using WS-CDL and WS-BPEL**. In: *Enterprise Distributed Object Computing Conference, 2007. EDOC 2007. 11th IEEE International*, p. 15–15. IEEE, 2007.

[231] Rouse, M. **Cloud bursting**, 2017. http://searchcloudcomputing.techtarget.com/definition/cloud-bursting, [Online; acesso em 06-Jan-2017].

[232] Saaty, T. L. **How to make a decision: the analytic hierarchy process**. *European Journal of Operational Research*, 48(1):9–26, sep 1990.

[233] Saaty, T. L. **Relative Measurement and its Generalization in Decision Making: Why Pairwise Comparisons are Central in Mathematics for the Measurement of Intangible Factors – The Analytic Hierarchy/Network Process**. *Review of the Royal Academy of Exact, Physical and Natural Sciences, Series A: Mathematics (RACSAM)*, 102(2):251–318, June 2008.

[234] Sahni, S. **Approximate algorithms for the 0/1 knapsack problem**. *Journal of the ACM (JACM)*, 22(1):115–124, 1975.

[235] Sandru, C.; Petcu, D.; Munteanu, V. I. **Building an open-source platform-as-a-service with intelligent management of multiple cloud resources**. In: *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing*, p. 333–338. IEEE Computer Society, 2012.

[236] Sandru, C.; Venticinque, S. **Agents layer to support cloud applications**. In: *Intelligent Distributed Computing VI*, p. 281–286. Springer, 2013.

[237] SAP. **Businnes by Design Software Solutions**, 2016. http://www.sap.com/pc/tech/cloud/software/business-management-bydesign, [Online; acesso em 08-Dez-2016].

[238] Schad, J.; Dittrich, J.; Quiané-Ruiz, J.-A. **Runtime measurements in the cloud: observing, analyzing, and reducing variance**. *Proceedings of the VLDB Endowment*, 3(1-2):460–471, 2010.

[239] Schmidt, D. C. **Model-driven engineering**. *IEEE Computer Society*, 39(2):25, 2006.

[240] Sedlacek, J.; Hurka, T. **VisualVM – All-in-One Java Troubleshooting Tool**, 2016. https://visualvm.github.io, [Online; acesso em 01-Nov-2016].

[241] Shachter, R. D. **Probabilistic inference and influence diagrams**. *Operations Research*, 36(4):589–604, 1988.

[242] Shi, L.; Chen, H.; Sun, J.; Li, K. **vCUDA: GPU-accelerated high-performance computing in virtual machines**. *IEEE Transactions on Computers*, 61(6):804–816, 2012.

[243] Singh, S.; Chana, I. **A survey on resource scheduling in cloud computing: Issues and challenges**. *Journal of Grid Computing*, 14(2):217–264, 2016.

[244] Siqueira, F.; Cahill, V. **Quartz: A QoS architecture for open systems**. In: *Proceedings of the 20th International Conference on Distributed Computing Systems*, p. 197–204. IEEE, 2000.

[245] Sirin, E.; Hendler, J.; Parsia, B. **Semi-automatic composition of web services using semantic descriptions**. *Proceedings of Web Services: Modeling, Architecture and Infrastructure workshop*, 3:17–24, 2003.

[246] Soltani, S.; Martin, P.; Elgazzar, K. **QuARAM Recommender: Case-Based Reasoning for IaaS Service Selection**. In: *Cloud and Autonomic Computing (ICCAC), 2014 International Conference on*, p. 220–226. IEEE, 2014.

[247] Springer. **Springer Link**, 2016. http://link.springer.com, [Online; acesso em 11-Dez-2016].

[248] Strauch, S.; Andrikopoulos, V.; Breitenbuecher, U.; Kopp, O.; Leymann, F. **Non-Functional Data Layer Patterns for Cloud Applications**. In: *Proceedings of the 4th IEEE International Conference on Cloud Computing Technology and Science*, p. 601–605. IEEE Computer Society, 2012.

[249] Su, J.; Bultan, T.; Fu, X.; Zhao, X. **Towards a theory of web service choreographies**. In: *Web Services and Formal Methods*, p. 1–16. Springer, 2008.

[250] Suleiman, B.; Sakr, S.; Jeffery, R.; Liu, A. **On understanding the economics and elasticity challenges of deploying business applications on public cloud infrastructure**. *Journal of Internet Services and Applications*, 3(2):173–193, 2012.

[251] Sun, Y. L.; Harmer, T.; Stewart, A.; Wright, P. **Mapping application requirements to cloud resources**. In: *Euro-Par 2011: Parallel Processing Workshops*, p. 104–112. Springer, 2012.

[252] Sundareswaran, S.; Squicciarini, A.; Lin, D. **A brokerage-based approach for cloud service selection**. In: *Proceedings of the IEEE 5th International Conference on Cloud Computing (CLOUD)*, p. 558–565. IEEE, 2012.

[253] Takacs, L. **On Erlang's formula**. *The annals of mathematical statistics*, 40(1):71–78, 1969.

[254] Torres, R.; Bencomo, N.; Astudillo, H. **Mitigating the obsolescence of quality specifications models in service-based systems**. In: *Model-Driven Requirements Engineering Workshop (MoDRE), 2012 IEEE*, p. 68–76. IEEE, 2012.

[255] University, T. P. S. **CiteSeerX**, 2016. http://citeseerx.ist.psu.edu, [Online; acesso em 11-Dez-2016].

[256] Vaquero, L. M.; Rodero-Merino, L.; Caceres, J.; Lindner, M. **A break in the clouds: towards a cloud definition**. *ACM SIGCOMM Computer Communication Review*, 39(1):50–55, 2008.

[257] Vilaplana, J.; Solsona, F.; Teixidó, I.; Mateo, J.; Abella, F.; Rius, J. **A queuing theory model for cloud computing**. *The Journal of Supercomputing*, 69(1):492–507, 2014.

[258] Vincent, H.; Issarny, V.; Georgantas, N.; Francesquini, E.; Goldman, A.; Kon, F. **CHOReOS: scaling choreographies for the internet of the future**. In: *Middleware'10 Posters and Demos Track*, p. 8. ACM, 2010.

[259] Vincenty, T. **Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations**. *Survey review*, 23(176):88–93, 1975.

[260] Wang, H.; Ma, P.; Yu, Q.; Yang, D.; Li, J.; Fei, H. **Combining Quantitative Constraints with Qualitative Preferences for Effective Non-Functional Properties-Aware Service Composition**. *Journal of Parallel and Distributed Computing*, 100:71 – 84, 2017.

[261] Wang, M.; Liu, J.; Wang, H.; Cheung, W. K.; Xie, X. **On-demand e-supply chain integration: A multi-agent constraint-based approach**. *Expert Syst Appl*, 34(4):2683–2692, 2008.

[262] Wieder, A.; Bhatotia, P.; Post, A.; Rodrigues, R. **Conductor: orchestrating the clouds**. In: *Proceedings of the 4th International Workshop on Large Scale Distributed Systems and Middleware*, p. 44–48. ACM, 2010.

[263] Wikipedia. **Mashup (música)**, 2016. https://pt.wikipedia.org/wiki/Mashup_(musica), [Online; acesso em 21-Dez-2016].

[264] Wikipedia. **Breadth-first search**, 2017. https://en.wikipedia.org/wiki/Breadth-first_search, [Online; acesso em 04-Jan-2017].

[265] Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M. C.; Regnell, B.; Wesslén, A. **Experimentation in Software Engineering: An Introduction**. Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[266] Wolf, K. **Does my service have partners?** In: *Transactions on Petri Nets and Other Models of Concurrency II*, p. 152–171. Springer, 2009.

[267] Wright, P.; Sun, Y.; Harmer, T.; Keenan, A.; Stewart, A.; Perrott, R. **A constraints-based resource discovery model for multi-provider cloud environments**. *Journal of Cloud Computing: Advances, Systems and Applications*, 1(1):6, 2012.

[268] Wu, L.; Garg, S. K.; Buyya, R. **SLA-based resource allocation for software as a service provider (SaaS) in cloud computing environments**. In: *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, p. 195–204. IEEE, 2011.

[269] Wu, Q.; Zhu, Q.; Jian, X.; Ishikawa, F. **Broker-based SLA-aware composite service provisioning**. *Journal of Systems and Software*, 96:194–201, 2014.

[270] Xiong, K.; Perros, H. **Service Performance and Analysis in Cloud Computing**. In: *Proceedings of the World Conference on Services*, p. 693–700. IEEE, 2009.

[271] Xu, F.; Liu, F.; Jin, H.; Vasilakos, A. V. **Managing Performance Overhead of Virtual Machines in Cloud Computing: A Survey, State of the Art, and Future Directions**. *Proceedings of the IEEE*, 102(1):11–31, Jan 2014.

[272] Yang, C.-T.; Huang, C.-L.; Lin, C.-F. **Hybrid CUDA, OpenMP, and MPI parallel programming on multicore GPU clusters**. *Computer Physics Communications*, 182(1):266–269, 2011.

[273] Yang, L.; Foster, I.; Schopf, J. M. **Homeostatic and tendency-based CPU load predictions**. In: *Proceedings of the Parallel and Distributed Processing Symposium*, p. 9–9. IEEE, 2003.

[274] Ye, Z.; Bouguettaya, A.; Zhou, X. **QoS-aware cloud service composition based on economic models**. In: *International Conference on Service-Oriented Computing*, p. 111–126. Springer, 2012.

[275] Ye, Z.; Zhou, X.; Bouguettaya, A. **Genetic algorithm based QoS-aware service compositions in cloud computing**. In: *International Conference on Database Systems for Advanced Applications*, p. 321–334. Springer, 2011.

[276] Yu, J.; Buyya, R.; Tham, C. K. **Cost-based scheduling of scientific workflow applications on utility grids**. In: *First International Conference on e-Science and Grid Computing (e-Science'05)*, p. 8–pp. IEEE, 2005.

[277] Yu, J.; Buyya, R.; Tham, C. K.; others. **QoS-based scheduling of workflow applications on service grids**. In: *Proceedings of the 1st IEEE International Conference on e-Science and Grid Computing*, 2005.

[278] Yu, T.; Zhang, Y.; Lin, K.-J. **Efficient algorithms for Web services selection with end-to-end QoS constraints**. *ACM Transactions on the Web (TWEB)*, 1(1):6, 2007.

[279] Zaha, J. M.; Barros, A.; Dumas, M.; ter Hofstede, A. **Let's dance: A language for service behavior modeling**. In: *OTM Confederated International Conferences On the Move to Meaningful Internet Systems*, p. 145–162. Springer, 2006.

[280] Zaha, J. M.; Dumas, M.; Ter Hofstede, A.; Barros, A.; Decker, G. **Service interaction modeling: Bridging global and local views**. In: *2006 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC'06)*, p. 45–55. IEEE, 2006.

[281] Zeleny, M. **Multiple criteria decision making**. McGraw-Hill, 1982.

[282] Zhang, Q.; Cheng, L.; Boutaba, R. **Cloud computing: state-of-the-art and research challenges**. *Journal of Internet Services and Applications*, 1(1):7–18, 2010.

[283] Zou, J.; Pavlovski, C. J. **Modeling architectural non functional requirements: from use case to control case**. In: *2006 IEEE International Conference on e-Business Engineering (ICEBE'06)*, p. 315–322. IEEE, 2006.

APÊNDICE A

# Variáveis usadas na representação formal

| Variável | Descrição |
|---|---|
| $\mathcal{S}$ | Conjunto de $n$ serviços $\{s_1, s_2, ..., s_n\}$ |
| $n$ | Número de serviços |
| $\mathcal{O}_{s_i}$ | Grupo de operações providas por um serviço $s_i$ |
| $\mathfrak{o}$ | Operação em um determinado serviço ($\mathfrak{o} \in \mathcal{O}$) |
| $d[]_{\mathfrak{o}}$ | Vetor de perfil de uma operação $\mathfrak{o}$ |
| $\mathcal{V}$ | Um conjunto de $t$ tipos de VM $\{v_1, v_2, ..., v_t\}$ disponíveis para implantação |
| $\mathscr{V}$ | Um vetor que representa os tipos concretos referentes aos tipos canônicos, em que $\mathscr{V}[j], 1 \leq j \leq \Psi$ especifica os tipos concretos representados pelo tipo canônico $j$ |
| $t$ | Número de tipos de VM |
| $\Psi$ | Número de nós canônicos |
| $v$ | Um tipo de VM (tipo concreto de recurso) |
| $v'$ | Um tipo canônico de recurso |
| $\zeta[]_v$ | Vetor com capacidade do tipo de VM $v$ |
| $\iota[]_v$ | Vetor com atributos que descrevem o tipo de VM $v$. $\iota[]_v \supset \zeta[]_v$ |
| $c_v$ | Custo financeiro de utilização de um VM do tipo $v$ |
| $\ell_v$ | Localização do tipo de VM $v$ |
| $p_v$ | Provedor de nuvem do tipo de VM $v$ |
| $m$ | Uma métrica de QoS |
| $\mathbb{D}_m$ | Domínio de uma métrica de QoS $m$ |
| $\oplus_m$ | Operador que define como o valor da métrica de QoS $m$ é incrementado por cada novo evento/operação |
| $\bigwedge_m$ | Limite inferior em uma métrica de QoS $m$ |
| $\bigvee_m$ | Limite superior em uma métrica de QoS $m$ |

*Continua na próxima página*

Tabela A.1 – *Continuação da página anterior*

| Variável | Descrição |
|---|---|
| $\mathcal{U}$ | Uma função de utilidade $(\mathcal{S},\mathcal{V}) \rightarrow \mathbb{D}$, que determina o valor de QoS estimada de acordo com uma determinada métrica quando um serviço $s \in \mathcal{S}$ é implantado em um recurso VM $v \in \mathcal{V}$ |
| $\mathcal{U}_{ijk}$ | O valor de QoS estimado pela função de utilidade $\mathcal{U}$ para a restrição de QoS $k$, quando o serviço $s_i, (1 \leq i \leq n)$ é implantado no recurso $v_j, (1 \leq j \leq t)$ |
| $\Omega$ | Conjunto de serviços ($\Omega \in \mathcal{S}$) sobre os quais uma restrição deve ser aplicada |
| $\square$ | Um operador relacional |
| $\tau_k$ | Valor-alvo na restrição $k$. Se $k$ for uma restrição de QoS, $\tau \in \mathbb{D}_m$, onde $m$ é a métrica de QoS |
| $x_k$ | Uma variável inteira que indica se a restrição $k$ foi satisfeita |
| $\phi$ | Atributo do recurso sobre o qual uma restrição deve ser aplicada |
| $\boxminus$ | Um operador de classificação |
| $\kappa$ | Conjunto de $\rho$ restrições não-funcionais |
| $\rho$ | Número de restrições |
| $\rho^q$ | Número de restrições de QoS |
| $\rho^e$ | Número de restrições eliminatórias |
| $\rho^r$ | Número de restrições classificatórias |
| $\mathcal{F}$ | Conjunto de pares $(s,v'), s \in \mathcal{S}, v' \equiv v \in \mathcal{V}$, que representam o resultado do mapeamento de serviços em tipos de VM |
| $f_i$ | Mapeamento do serviço $s_i$ para o tipo de VM $v_j$. Tem-se que $f_i(s_i)$ e $f_i(v'_i)$ representam o serviço e o tipo de VM no mapeamento, respectivamente |
| $f_i(s_i)$ | Serviço no mapeamento $f_i$ |
| $f_i(v'_i)$ | Tipo de VM no mapeamento $f_i$ |
| $\varphi_{ij}$ | Uma função objetivo que representa o ganho ao alocar o tipo de VM $j$ para implantar o serviço $i$ |
| $\varphi(\mathcal{F})$ | O perfil conjunto de todos os tipos de VM em $\mathcal{F}$ |
| $b$ | Nó inicial em uma estrutura de grafo |
| $Z$ | Conjunto de nós finais em um grafo de processo |
| $z$ | Nó final em uma estrutura de grafo |
| $L$ | Conjunto de conectores em um grafo de processo |
| $E$ | Conjunto de arestas que definem o fluxo em um grafo de processo |
| $\underrightarrow{\mathfrak{e}}$ | Nó origem de uma arestas $\mathfrak{e} \in E$ |

*Continua na próxima página*

Tabela A.1 – *Continuação da página anterior*

| **Variável** | **Descrição** |
|---|---|
| $\overrightarrow{\mathfrak{e}}$ | Nó destino de uma aresta $\mathfrak{e} \in E$ |
| $k$ | Uma restrição |
| $\mathcal{K}$ | Um grupo em restrição |
| $\mathbb{K}$ | Conjunto de grupos em restrição |
| $\lambda$ | Carga de requisições em uma certa coreografia |
| $\mathcal{L}$ | Conjunto de descritores de carga em uma árvore de paralelismo |
| $\Lambda$ | Conjunto de árvores de paralelismo relacionadas a uma restrição |

APÊNDICE B

# Esquema XML da representação do grafo de processo e das restrições não-funcionais

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema attributeFormDefault="unqualified"
  elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="vertexType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="START" />
      <xs:enumeration value="END" />
      <xs:enumeration value="LOAD" />
      <xs:enumeration value="SERVICE" />
      <xs:enumeration value="ANDS" />
      <xs:enumeration value="ANDJ" />
      <xs:enumeration value="XORS" />
      <xs:enumeration value="XORJ" />
      <xs:enumeration value="ORS" />
      <xs:enumeration value="ORJ" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="positiveDouble">
    <xs:restriction base="xs:double">
      <xs:minInclusive value="0.0" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="positiveInteger">
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0" />
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="processGraph">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="vertices">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="vertex" maxOccurs="unbounded"
                minOccurs="3">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="serviceDescriptor" minOccurs="0">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element type="xs:string" name="name" />
                          <xs:element name="dependencies" minOccurs="0">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="dependency" maxOccurs="unbounded"
                                    minOccurs="0">
                                  <xs:complexType>
                                    <xs:simpleContent>
                                      <xs:extension base="xs:string">
                                        <xs:attribute type="xs:string" name="
                                            serviceSpecName" use="optional" />
```

```
50                                      <xs:attribute type="xs:string" name="
                                            serviceSpecRole" use="optional" />
51                                    </xs:extension>
52                                  </xs:simpleContent>
53                                </xs:complexType>
54                              </xs:element>
55                            </xs:sequence>
56                          </xs:complexType>
57                        </xs:element>
58                        <xs:element name="logicalOperations" minOccurs="1">
59                          <xs:complexType>
60                            <xs:sequence>
61                              <xs:element name="logicalOperation" minOccurs="1"
                                    maxOccurs="unbounded">
62                                <xs:complexType>
63                                  <xs:sequence>
64                                    <xs:element type="xs:string" name="name" minOccurs="
                                          1" maxOccurs="1" />
65                                    <xs:element type="xs:anyURI" name="uri" minOccurs="0
                                          " maxOccurs="1" />
66                                    <xs:element type="xs:string" name="roles" minOccurs=
                                          "0" maxOccurs="1" />
67                                    <xs:element type="xs:string" name="messages"
                                          minOccurs="0" maxOccurs="1" />
68                                    <xs:element type="xs:string" name="usagePolices"
                                          minOccurs="0" maxOccurs="1" />
69                                    <xs:element type="positiveDouble" name="
                                          numberOfInstructions" minOccurs="1" maxOccurs="1
                                          " />
70                                  </xs:sequence>
71                                </xs:complexType>
72                              </xs:element>
73                            </xs:sequence>
74                          </xs:complexType>
75                        </xs:element>
76                        <xs:element type="xs:string" name="serviceType" minOccurs="0"
                              maxOccurs="1" />
77                        <xs:element type="xs:string" name="roles" minOccurs="0"
                              maxOccurs="1" />
78                        <xs:element type="xs:string" name="serverMethod" minOccurs="0"
                              maxOccurs="1" />
79                        <xs:element type="xs:string" name="category" minOccurs="0"
                              maxOccurs="1" />
80                        <xs:element type="xs:anyURI" name="packageURI" minOccurs="0"
                              maxOccurs="1" />
81                        <xs:element type="xs:string" name="packageType" minOccurs="0"
                              maxOccurs="1" />
82                        <xs:element type="xs:string" name="endpointName" minOccurs="0"
                              maxOccurs="1" />
83                        <xs:element type="xs:string" name="port" minOccurs="0" maxOccurs
                              ="1" />
84                        <xs:element type="xs:string" name="numberOfInstances" minOccurs=
                              "0" maxOccurs="1" />
85                        <xs:element type="xs:string" name="version" minOccurs="0"
                              maxOccurs="1" />
86                      </xs:sequence>
87                    </xs:complexType>
88                  </xs:element>
89                </xs:sequence>
90                <xs:attribute type="xs:string" name="id" use="required" />
91                <xs:attribute type="vertexType" name="type" use="required" />
92              </xs:complexType>
93            </xs:element>
94          </xs:sequence>
95        </xs:complexType>
96      </xs:element>
97      <xs:element name="edges">
98        <xs:complexType>
99          <xs:sequence>
100           <xs:element name="edge" maxOccurs="unbounded" minOccurs="2">
101             <xs:complexType mixed="true">
102               <xs:sequence>
103                 <xs:element type="xs:string" name="logicalOperation" minOccurs="0" />
```

```
104                     <xs:element type="positiveDouble" name="load" minOccurs="0" />
105                   </xs:sequence>
106                   <xs:attribute type="xs:string" name="vertexFrom" use="required" />
107                   <xs:attribute type="xs:string" name="vertexTo" use="required" />
108                 </xs:complexType>
109               </xs:element>
110             </xs:sequence>
111           </xs:complexType>
112         </xs:element>
113         <xs:element name="constraints" maxOccurs="unbounded" minOccurs="0">
114           <xs:complexType>
115             <xs:sequence>
116               <xs:element name="qosConstraints">
117                 <xs:complexType>
118                   <xs:sequence>
119                     <xs:element name="qosConstraint" maxOccurs="unbounded" minOccurs="0">
120                       <xs:complexType>
121                         <xs:sequence>
122                           <xs:element type="xs:string" name="id" maxOccurs="1" minOccurs="
                                1" />
123                           <xs:element type="xs:string" name="constraintClass" maxOccurs="1
                                " minOccurs="1" />
124                           <xs:element name="targetOperations" maxOccurs="1" minOccurs="1">
125                             <xs:complexType>
126                               <xs:sequence>
127                                 <xs:element name="targetOperation" maxOccurs="unbounded"
                                      minOccurs="1">
128                                   <xs:complexType>
129                                     <xs:simpleContent>
130                                       <xs:extension base="xs:string">
131                                         <xs:attribute type="xs:string" name="service" use=
                                              "required" />
132                                         <xs:attribute type="xs:string" name="operation"
                                              use="required" />
133                                       </xs:extension>
134                                     </xs:simpleContent>
135                                   </xs:complexType>
136                                 </xs:element>
137                               </xs:sequence>
138                             </xs:complexType>
139                           </xs:element>
140                           <xs:element type="xs:string" name="targetValue" />
141                           <xs:element type="xs:string" name="toleratedValue" minOccurs="0"
                                />
142                           <xs:element type="xs:string" name="valueUnitEnum" />
143                           <xs:element type="xs:string" name="valueUnit" />
144                           <xs:element type="xs:string" name="utilityEstimator" />
145                           <xs:element type="xs:string" name="utilityParametersBuilder" />
146                           <xs:element type="xs:string" name="secondUtilityEstimator"
                                minOccurs="0" />
147                           <xs:element type="xs:string" name="
                                secondUtilityParametersBuilder" minOccurs="0" />
148                           <xs:element name="parallelTrees" maxOccurs="unbounded" minOccurs
                                ="0">
149                             <xs:complexType>
150                               <xs:sequence>
151                                 <xs:element name="parallelTree" maxOccurs="unbounded"
                                      minOccurs="0">
152                                   <xs:complexType>
153                                     <xs:sequence>
154                                       <xs:element type="positiveDouble" name="root"
                                            maxOccurs="1" minOccurs="1" />
155                                       <xs:element name="targetOperations" maxOccurs="1"
                                            minOccurs="1">
156                                         <xs:complexType>
157                                           <xs:sequence>
158                                             <xs:element name="targetOperation" maxOccurs="
                                                  unbounded" minOccurs="2">
159                                               <xs:complexType>
160                                                 <xs:simpleContent>
161                                                   <xs:extension base="xs:string">
162                                                     <xs:attribute type="xs:string" name="
                                                          service" use="required" />
```

```
163                                                    <xs:attribute type="xs:string" name="
                                                           operation" use="required" />
164                                                  </xs:extension>
165                                                </xs:simpleContent>
166                                              </xs:complexType>
167                                            </xs:element>
168                                          </xs:sequence>
169                                        </xs:complexType>
170                                      </xs:element>
171                                    </xs:sequence>
172                                  </xs:complexType>
173                                </xs:element>
174                              </xs:sequence>
175                            </xs:complexType>
176                          </xs:element>
177                        </xs:sequence>
178                      </xs:complexType>
179                    </xs:element>
180                  </xs:sequence>
181                </xs:complexType>
182              </xs:element>
183              <xs:element name="resourceConstraints">
184                <xs:complexType>
185                  <xs:sequence>
186                    <xs:element type="xs:string" name="generalConstraintsDescriptor"
                           maxOccurs="1" minOccurs="0" />
187                    <xs:element name="serviceConstraintDescriptor" maxOccurs="unbounded"
                           minOccurs="0">
188                      <xs:complexType>
189                        <xs:simpleContent>
190                          <xs:extension base="xs:string">
191                            <xs:attribute type="xs:string" name="service" use="required" /
                                 >
192                            <xs:attribute type="xs:string" name="operation" use="optional"
                                 />
193                          </xs:extension>
194                        </xs:simpleContent>
195                      </xs:complexType>
196                    </xs:element>
197                  </xs:sequence>
198                </xs:complexType>
199              </xs:element>
200            </xs:sequence>
201          </xs:complexType>
202        </xs:element>
203      </xs:sequence>
204    </xs:complexType>
205  </xs:element>
206 </xs:schema>
```