

Universidade Federal de Goiás
Instituto de Informática

Jesus José de Oliveira Neto

**Uso de um Modelo de Interceptadores
para Prover Adaptação Dinâmica no
InteGrade**

Goiânia
2008

Jesus José de Oliveira Neto

Uso de um Modelo de Interceptadores para Prover Adaptação Dinâmica no InteGrade

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

Área de concentração: Ciência da Computação.

Orientador: Prof. Dr. Fábio Moreira Costa

Goiânia
2008

Jesus José de Oliveira Neto

Uso de um Modelo de Interceptadores para Prover Adaptação Dinâmica no InteGrade

Dissertação defendida no Programa de Pós-Graduação do Instituto de Informática da Universidade Federal de Goiás como requisito parcial para obtenção do título de Mestre em Ciência da Computação, aprovada em 25 de Abril de 2008, pela Banca Examinadora constituída pelos professores:

Prof. Dr. Fábio Moreira Costa
Instituto de Informática – UFG
Presidente da Banca

Prof. Dr. Renato Fontoura de Gusmão Cerqueira
Departamento de Informática – PUC-Rio

Prof. Dr. André Barros de Sales
Departamento de Computação – UCG

Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, do autor e do orientador(a).

Jesus José de Oliveira Neto

Graduou-se em Ciência da Computação pela Universidade Federal de Goiás (2004). Durante sua graduação, foi monitor da disciplina de Linguagens e Técnicas de Programação em 2003 e no ano seguinte, na disciplina de Projeto e Análise de Algoritmos. Entre 2005 e 2006, foi bolsista DTI do CNPq no Instituto de Informática da UFG. Atualmente está cursando o mestrado em Ciência da Computação pela Universidade Federal de Goiás, concentrando sua pesquisa no desenvolvimento de mecanismos adaptativos para computação em grade.

Obrigado à minha mãe, minha irmã e aos meus avós, por todo o apoio que me deram durante a elaboração desta dissertação.

Agradecimentos

À minha mãe, minha irmã e meus avós pela força que me deram desde o início.

Aos colegas do GEApIS - Grupo de Estudos Aplicados à Internet e Sistemas Distribuídos pelos momentos de descontração e amizade. Agradeço também por toda informação que obtive do laboratório do GEApIS que foi de grande importância para este trabalho.

A todos que, direta ou indiretamente, contribuíram para a realização deste trabalho.

Aquele que conhece bem os outros é um sábio, mas aquele que conhece bem a si mesmo é um iluminado.

Lao-Tsé.

Resumo

José de Oliveira Neto, Jesus. **Uso de um Modelo de Interceptadores para Prover Adaptação Dinâmica no InteGrade**. Goiânia, 2008. 84p. Dissertação de Mestrado. Instituto de Informática, Universidade Federal de Goiás.

Grades computacionais são conjuntos de recursos computacionais que fornecem diversos tipos de serviços, tais como armazenamento e processamento, para aplicações que podem estar espalhadas por diferentes domínios administrativos. Desta forma, várias empresas e instituições acadêmicas têm interesse no seu uso para a execução de aplicações que exijam um alto poder computacional. Entretanto, grades computacionais são ambientes de execução bastante diversificados e complexos, pois possuem alta variação na disponibilidade de recursos, instabilidade de seus nós e variações na distribuição de carga, entre outros problemas. Este trabalho apresenta um modelo de interceptadores dinâmicos e seu uso no InteGrade, um *middleware* de grade oportunista. O uso de interceptadores tem por finalidade prover suporte para adaptação dinâmica no InteGrade através de seu *middleware* de comunicação, assim, contribuindo para que o mesmo tenha condições de lidar com o ambiente de execução altamente variável das grades computacionais sem que sejam necessárias alterações em sua implementação. Desta forma, este trabalho busca fornecer recursos de adaptação dinâmica para o InteGrade e não para aplicações de grade. No entanto, estas aplicações poderão se beneficiar dos recursos de adaptação oferecidos pelo InteGrade.

Palavras-chave

Interceptadores dinâmicos, *middleware* de grade, adaptação dinâmica, reflexão computacional.

Abstract

José de Oliveira Neto, Jesus. **Use of an Interceptor Model to provide Dynamic Adaptation in InteGrade**. Goiânia, 2008. 84p. MSc. Dissertation. Instituto de Informática, Universidade Federal de Goiás.

Computer grids are sets of computational resources that provide diverse types of services, such as storage and processing, on behalf of applications spread across different administrative domains. Many companies and academic institutions have demonstrated interest in their use for the execution of applications that demand huge amounts of computation power and storage. However, computer grids are complex and diversified execution environments, which exhibit high variation of resource availability, node instability and variations on load distribution, among other problems. This work presents a model of dynamic interceptors and its use in InteGrade, an opportunistic grid middleware. The use of interceptors aims to provide dynamic adaptation in InteGrade through its communication middleware, thus contributing to make InteGrade able to deal with the highly variable execution environment of computer grids without requiring changes to its implementation. Therefore, this work aims to offer dynamic adaptation capabilities to InteGrade and not to grid applications. Nevertheless, these applications will be able to benefit from adaptation provided by InteGrade.

Keywords

Dynamic interceptors, grid middleware, dynamic adaptation, computational reflection.

Sumário

Lista de Figuras	xi
Lista de Tabelas	xii
Lista de Códigos de Programas	xiii
Glossário	1
1 Introdução	4
1.1 Motivação	7
1.2 Objetivos do Trabalho	7
2 <i>Middleware</i> de Grade e Adaptação Dinâmica	9
2.1 Plataformas de <i>Middleware</i> Convencionais	9
2.2 Plataformas de <i>Middleware</i> de Grade	11
2.2.1 InteGrade	12
2.2.2 Globus	17
2.2.3 Condor	20
2.3 Adaptação Dinâmica	22
2.3.1 Reflexão Computacional	22
2.3.2 A Linguagem Lua	23
2.3.3 <i>Middleware</i> Reflexivo	28
<i>DynamicTAO</i>	29
<i>Open ORB</i>	31
2.4 <i>Middleware</i> de Grade Reflexivo	32
2.4.1 AutoGrid	33
Arquitetura do AutoGrid	33
A Capacidade de Auto-Gerenciamento do AutoGrid	34
Aspectos Autônômicos do AutoGrid	34
Mecanismo de Domínio de Contexto	34
Mecanismo de Auto-Configuração	35
Mecanismo de Auto-Recuperação	36
Mecanismo de Auto-Otimização	37
2.4.2 AutoMate	37
Arquitetura do AutoMate	39
2.5 Considerações Finais	40

3	O Modelo de Interceptadores Dinâmicos	42
3.1	Conceitos básicos de Interceptadores	42
3.2	Visão Geral do Modelo de Interceptadores Dinâmicos	43
3.3	Arquitetura	45
3.4	Implementação	47
3.4.1	Implementação dos Pontos de Interceptação e do Componente <i>Context</i>	47
3.4.2	Implementação do Componente <i>Monitor</i> A função <i>Activate_Code_Runtime()</i>	48 51
3.4.3	Implementação do Componente <i>Implementor</i>	52
3.5	Comparações do Modelo de Interceptadores com outros Trabalhos	52
3.5.1	Comparação do Modelo de Interceptadores com o AutoGrid	53
3.5.2	Comparações do Modelo de Interceptadores com o AutoMate	53
3.6	O Modelo de Interceptadores no JacORB	54
3.7	O Modelo de Interceptadores no OiL baseado em Componentes	56
3.8	Considerações Finais	58
4	Aplicações do Modelo de Interceptadores Dinâmicos	60
4.1	Travessia de <i>Firewall</i> e NAT	60
4.1.1	Abordagem da OMG para Travessia de <i>Firewall</i> e NAT	62
4.1.2	Implementação da Abordagem da OMG através de Interceptadores Dinâmicos	63
4.1.3	Exemplos de dois Cenários de Uso da Travessia de <i>Firewall</i> e NAT utilizando Interceptadores	65
4.1.4	Avaliação do Uso de Interceptadores para Travessia de <i>Firewall</i> e NAT Descrição do Ambiente Experimental Medição do <i>Overhead</i>	67 67 68
4.1.5	Abordagem <i>Proxy</i> TCP para Travessia de <i>Firewall</i> e NAT	70
4.1.6	O Modelo de Interceptadores para Aplicação da Abordagem <i>Proxy</i> TCP	71
4.2	Frequência de Atualização de Informações da Grade	71
4.2.1	Uso de Interceptadores Dinâmicos para Alterar a Frequência de Atualização	72
4.3	Considerações Finais	73
5	Conclusão	76
5.1	Resultados Obtidos	78
5.2	Trabalhos Futuros	79
	Referências Bibliográficas	80

Lista de Figuras

2.1 Uma plataforma de <i>middleware</i> convencional	10
2.2 Arquitetura do InteGrade [7]	13
2.3 Protocolo de Execução de Aplicações do InteGrade [24]	16
2.4 Componentes da segunda versão do Globus Toolkit [23]	17
2.5 Globus Toolkit na versão 3 [23]	19
2.6 OGSi para WSRF [26]	20
2.7 Arquitetura de um Condor Pool [10]	21
2.8 Estrutura Interna do ORB DynamicTAO [33]	30
2.9 Estrutura do meta-espço em Open ORB [5]	31
2.10 Visão Geral do AutoMate [1]	38
2.11 Arquitetura do AutoMate [1]	39
3.1 Visão geral do projeto	45
3.2 Arquitetura do Modelo de Interceptadores Dinâmicos	46
3.3 Pontos de Interceptação dentro ORB OiL	47
3.4 Versão do OiL baseada em componentes	57
4.1 Abordagem da OMG para Travessia de <i>Firewall</i> e NAT [11]	62
4.2 Os pontos de interceptação utilizados para implementação de travessia de <i>Firewall</i> e NAT	65
4.3 Primeiro Cenário de Uso da Travessia de <i>Firewall</i> e NAT utilizando Interceptadores	65
4.4 Segundo Cenário de Uso da Travessia de <i>Firewall</i> e NAT utilizando Interceptadores	66
4.5 Topologia da rede utilizada nos testes	68
4.6 Abordagem <i>Proxy</i> TCP para Travessia de <i>Firewall</i> e NAT [11]	71
4.7 Ponto de Interceptação para a implementação da abordagem <i>Proxy</i> TCP	72
4.8 Ponto de interceptação utilizado para diminuir a freqüência de atualização quando necessário	73

Lista de Tabelas

4.1 Características de Hardware e Software do Ambiente Experimental	67
4.2 Medição do <i>Overhead</i> Para a Inicialização do LRM	69
4.3 Medição do <i>Overhead</i> Para a Requisição de Execução de uma Aplicação Simples	69

Lista de Códigos de Programas

2.1	Exemplo de utilização da função dofile()	24
2.2	Arquivo carregado pela função dofile()	24
2.3	Arquivo carregado pela função loadstring()	25
2.4	Exemplo de um código equivalente ao retornado pela função loadstring() ou loadfile()	25
2.5	Exemplo de um código que retorna sua função principal	26
2.6	Código definido acima retornado pela função loadstring() ou loadfile()	26
2.7	Carregamento dinâmico de uma função que recebe parâmetros e/ou retorna algum resultado	27
2.8	Utilização da função pcall para chamar as funções carregadas dinamicamente	27
3.1	Componente Monitor	49
3.2	Componente Implementor	52

Glossário

API	Application Programming Interface
AR	Application Repository
ARSC	Application Repository Security Client
ARSM	Application Repository Security Manager
ASCT	Application Submission and Control Tool
BSP	Bulk Synchronous Parallel
BSPLib	BSP Library
CCM	CORBA Component Model
CDRM	Cluster Data Repository Manager
CkpRep	Checkpoint Repository
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
DSRT	Dynamic Soft Real-Time Scheduler
DyReS	Dynamic Reconfiguration System
EM	Execution Manager
EPS	Event Process System
GIIS	Grid Index Information Service
GIOP	General Inter-ORB Protocol
GRAM	Globus Resource Allocation Manager
GRIS	The Grid Resource Information Service

GRM	Global Resource Manager
GSI	Grid Security Infrastructure
HTTP	Hypertext Transfer Protocol
IDL	Interface Definition Language
IOP	Internet Inter-ORB Protocol
IOR	Interoperable Object Reference
LES	Local Event Service
LRM	Local Resource Manager
LuaCCM	Lua CORBA Component Model
LUPA	Local Usage Pattern Analyzer
MCT	Minimum Completion Time
MDS	Monitoring and Discovery Service
MOP	Meta-Object Protocol
MPI	Message Passing Interface
MS	Monitoring Service
MTBF	Mean Time Between Failures
NAT	Network Address Translation
NCC	Node Control Center
OGSA	Open Grid Services Architecture
OGSI	Open Grid Services Infrastructure
OiL	ORB in Lua
OMG	Object Management Group
ORB	Object Request Broker
PDA	Personal Digital Assistant
PKI	Public Key Infrastructure

RFT	Reliable File Transfer
SSL	Security Socket Layer
TLS	Transport Layer Security
WSDL	Web Services Description Language
WSRF	Web Service Resource Framework
XML	Extensible Markup Language