UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

ALEX RABELO FERREIRA

Um Modelo Analítico para Estimar o Consumo de Energia de Sistemas multi-camadas no Nível de Transação

Goiânia
2017

sistema de bibliotecas ufg

UFG

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: **[x] Dissertação** **[ ] Tese**

2. Identificação da Tese ou Dissertação

Nome completo do autor: Alex Rabelo Ferreira

Título do trabalho: Um modelo analítico para estimar o consumo de energia de sistemas multicamadas no nível de transação

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.

Alex Rabelo Ferreira
Assinatura do (a) autor (a)

Data: 19 / 05 / 17

[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.

ALEX RABELO FERREIRA

Um Modelo Analítico para Estimar o Consumo de Energia de Sistemas multi-camadas no Nível de Transação

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 Mestrado Stricto Sensu em Ciência da Computação.

Área de concentração: Ciência da Computação.

Orientadora: Profa. Dra. Sand Luz Corrêa

Co-Orientadora: Profa. Dra. Valéria Quadros dos Reis

Goiânia
2017

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

Rabelo Ferreira, Alex
Um modelo analítico para estimar o consumo de energia de sistemas multi-camadas no nível de transação [manuscrito] / Alex Rabelo Ferreira. - 2017.
LVIII, 58 f.: il.

Orientador: Profa. Dra. Sand Luz Corrêa; co-orientadora Dra. Valéria Quadros dos Reis.
Dissertação (Mestrado) - Universidade Federal de Goiás, Instituto de Informática (INF), Programa de Pós-Graduação em Ciência da Computação, Goiânia, 2017.
Bibliografia.
Inclui gráfico, tabelas, lista de figuras, lista de tabelas.

1. Modelo Analítico. 2. Eficiência Energética. 3. Aplicações Web Multi-camadas. I. Luz Corrêa, Sand, orient. II. Título.

CDU 004

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UFG

ATA Nº 07/2017

ATA DA SESSÃO DE JULGAMENTO DA DISSERTAÇÃO
DE MESTRADO DE ALEX RABELO FERREIRA

Aos vinte e cinco dias do mês de abril de dois mil e dezessete, às catorze horas, na sala 151 do Instituto de Informática da Universidade Federal de Goiás, Campus Samambaia, reuniu-se a banca examinadora designada na forma regimental pela Coordenação do Curso para julgar a dissertação de mestrado intitulada **"Um modelo analítico para estimar o consumo de energia de sistemas multi-camadas no nível de transação"**, apresentada pelo aluno Alex Rabelo Ferreira como parte dos requisitos necessários à obtenção do grau de Mestre em Ciência da Computação, área de concentração Ciência da Computação. A banca examinadora foi presidida pela orientadora do trabalho de dissertação, Professora Doutora Sand Luz Corrêa (INF/UFG), tendo como membros os Professores Doutores Valéria Quadros dos Reis (coorientadora – FACOM/UFMS), Vinicius Tavares Petrucci (DCC/UFBA) e Wellington Santos Martins (INF/UFG). A profa. Valéria Quadros dos Reis participou à distância por webconferência. Aberta a sessão, o candidato expôs seu trabalho. Em seguida, o aluno foi arguido pelos membros da banca e:

(X) tendo demonstrado suficiência de conhecimento e capacidade de sistematização do tema de sua dissertação, a banca concluiu pela **aprovação** do candidato, sem restrições.

( ) tendo demonstrado suficiência de conhecimento e capacidade de sistematização do tema de sua dissertação, a banca concluiu pela **aprovação** do candidato, condicionado a satisfazer as exigências listadas na Folha de Modificação de Dissertação de Mestrado anexa à presente ata, no prazo máximo de 60 dias, a contar da presente data, ficando o professor-orientador responsável por atestar o cumprimento dessas exigências.

( ) não tendo demonstrado suficiência de conhecimento e capacidade de sistematização do tema de sua dissertação, a banca concluiu pela **reprovação** do candidato.

Os trabalhos foram encerrados às 17 horas. Nos termos do Regulamento Geral dos Cursos de Pós-Graduação desta Universidade, lavrou-se a presente ata que, lida e julgada conforme, segue assinada pelos membros da banca examinadora.

Profa. Dra. Sand Luz Corrêa _____
Profa. Dra. Valéria Quadros dos Reis _____
Prof. Dr. Vinicius Tavares Petrucci _____
Prof. Dr. Wellington Santos Martins _____

Alex Rabelo Ferreira

Fez sua graduação em Sistemas de Informação na UFG - Universidade Federal de Goiás. Durante sua graduação, participou do programa de monitoria no Instituto de Informática (INF) da UFG e trabalhou como bolsista em projetos de presquisas na área de sistemas distribuídos. Durante o período acadêmico, participou ativamente como representante dos discentes, participando de reuniões e assembléias.

Eu dedico este trabalho a minha família que sempre me apoiou para eu me qualificar e ser um profissional bem qualificado. Dedico também a minha orientadora que esteve sempre me guiando para conseguir finalizar meu trabalho com qualidade, seguindo o cronograma do mestrado.

Agradecimentos

Gostaria de agradecer primeiramente a Deus, por me proporcionar uma boa saúde durante todo o trabalho, a minha família por apoiar e compreender meus momentos de ausência. Agradecer também as pessoas que contribuíram fortemente com os resultados obtidos por este trabalho, como as pessoas de um grupo de pesquisa formado pela minha orientadora, co-orientadora e o Denner Vidal. Gostaria de agradecer também os professores do INF por contribuir com a resolução de dúvidas que surgiram durante a pesquisa. Gostaria de agradecer ao INF e a UFG como um todo pela estrutura disponibilizada. Por fim, gostaria de agradecer ao CNPQ pela bolsa disponibilizada para eu conseguir me manter financeiramente durante a pesquisa.

Os grande feitos são conseguidos não pela força, mas pela perseverança

Samuel Johnson,
1709-1784.

Resumo

Rabelo Ferreira, Alex. **Um Modelo Analítico para Estimar o Consumo de Energia de Sistemas multi-camadas no Nível de Transação**. Goiânia, 2017. 57p. Dissertação de Mestrado. Instituto de Informática, Universidade Federal de Goiás.

Em grandes centros de dados, técnicas de gerenciamento de energia como *Dynamic Voltage/Frequency Scaling (DVFS)*, consolidação de máquinas virtuais e mecanismos de limitação de energia prometem grande economia de energia quando comparados a métodos tradicionais de gerenciamento de recursos. A maioria dessas técnicas utilizam mecanismos caixas-pretas para monitorar o comportamento da carga de trabalho. Contudo, esse tipo de monitoramento não é satisfatório para prever os fenômenos de rajadas, comumente encontrados em serviços e aplicações Web. Neste trabalho, propomos um modelo analítico para estimar o consumo de energia de Sistemas Web multi-camadas. Diferentemente de outros trabalhos, nosso modelo captura o padrão de consumo desses sistemas na granularidade de transações e para cada camada do sistema. Além disso, nosso modelo se baseia apenas na utilização de CPU e em parâmetros arquiteturais do servidor, os quais podem ser facilmente obtidos nos ambientes de produção atuais. Demonstramos a efetividade do nosso modelo em um ambiente de experimentação real, baseado no *benchmark* TPC-W. Os resultados obtidos mostram que nosso modelo é capaz de estimar o consumo de energia para um sistema Web com um erro relativo médio de 6,5% para o pior cenário, enquanto modelos mais complexos da literatura apresentam erros com a mesma ordem de grandeza.

Palavras–chave

Modelo Analítico, Eficiência Energética, Aplicações Web Multi-camadas

Abstract

Rabelo Ferreira, Alex. **An Analytical Model for Estimating the Energy Consumption of Multi-Tier Systems at the Level of Transactions**. Goiânia, 2017. 57p. MSc. Dissertation. Instituto de Informática, Universidade Federal de Goiás.

In large-scale data centers, power management techniques such as Dynamic Voltage/Frequency Scaling (DVFS), virtual machine consolidation, and power-capping mechanisms promise impressive energy savings compared to traditional resource management strategies. Most of these techniques rely on coarse-grained monitoring of the workload behavior to apply their optimizations. However, coarse-grained monitoring and black box observations are not satisfactory for predicting the behavior of bursty workloads such as those observed in enterprise, Web servers. In this work, we propose an analytical model to estimate the energy consumption of multi-tier Web Systems. Differently from previous works, our model captures the consumption pattern at the level of fine-grained transactions and for each tier of the system. In addition, our model is based only on CPU utilization and server architectural parameters, which can be easily obtained in today's production environments. We demonstrate the effectiveness of our model in a real-world experimentation environment, based on the TPC-W benchmark. Results show that our model is able to estimate the energy consumption for a Web system with an average relative error of 6.5% in the worst-case scenario, whereas more complex models of the literature present errors within the same order of magnitude.

Keywords

Analytic Model, Energy Efficiency, Multi-tier Web Applications

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

Diversos serviços e aplicações Web estão presentes em nossa vida cotidiana e nos negócios. Atualmente, o comércio eletrônico representa uma grande parcela das vendas e compras realizadas no mercado mundial. Redes sociais, como o Twitter e Facebook, são usadas por milhões de usuários para compartilhar conteúdo, e serviços como Netflix e Youtube possibilitam às pessoas acessarem vídeos a qualquer momento através da Internet. Na era da computação em nuvem, a maioria dessas aplicações e serviços são disponibilizados através de grandes *data centers* contendo milhares de servidores, os quais podem consumir até 10 MW por hora em períodos de picos [13]. Adicionalmente, o custo para alimentar um servidor comercial típico, durante seu tempo de vida, já ultrapassa o custo de aquisição [50]. Como consequência, o consumo de energia se tornou uma restrição importante em ambientes de computação de grande escala [4].

Nos últimos anos, muitos esforços foram empreendidos com o objetivo de reduzir o consumo de energia em *data centers* voltados para a computação em nuvem. Técnicas como *Dynamic Voltage/Frequency Scaling* (DVFS) [16, 24, 27, 51, 39, 52, 32], por exemplo, ajustam os estados dos servidores de acordo com a carga percebida. Técnicas de virtualização [23, 8, 58, 14, 34] possibilitam a redução no consumo de energia através da consolidação de máquinas virtuais (VM) em um número reduzido de servidores físicos. Mais recentemente, mecanismos limitadores de energia, como o RAPL (*Running Average Power Limit*) [26, 18], podem ser utilizados para configurar limiares de potência para um servidor.

Todas essas técnicas de gerenciamento de energia levam em consideração algum conhecimento da carga atual para realizarem decisões de alocação de recursos. DVFS, por exemplo, utiliza informações de históricos de consumo de CPU para ajustar a voltagem e a frequência de operação dos processadores. Algoritmos de consolidação de VMs, por sua vez, empregam históricos de utilização de CPU para calcular um valor estático com o qual a alocação de VMs nos servidores pode ser resolvida como um problema de otimização do tipo *bin packing*. Em todas essas técnicas, contudo, a atividade da CPU é monitorada como uma caixa-preta. Enquanto esse tipo de monitoramento não representa um problema para caracterizar o consumo energético de cargas de trabalho com o comportamento mais

regular, como as encontradas em ambientes de processamento de alto desempenho (PAD), ele não apresenta informações suficientes para tratar, de forma acurada, sistemas Web. De fato, um estudo sobre o impacto de DVFS no desempenho de sistemas Web, revelou que essa técnica pode aumentar significativamente o tempo de resposta do sistema e reduzir a vazão em até 20% [59].

Aplicações Web modernas são sistemas formados por componentes diferentes, incluindo interfaces gráficas, componentes lógicos e dados. Em geral, a carga de trabalho desses sistemas consiste em um conjunto de transações que processam pequenos objetos e consultas em diferentes camadas. Outra característica de cargas de trabalho dessa natureza são os fenômenos de rajadas, ou seja, o aumento inesperado de solicitações de serviços, o qual gera picos temporais irregulares na intensidade de chegadas de requisições dos clientes. Devido aos fenômenos de rajadas e atrasos nas diferentes camadas, o padrão de utilização dos recursos nesses sistemas pode alterar de maneira considerável e em um curto espaço de tempo [38].

Nesse contexto, a observação de demandas de serviços de componentes mais finos do sistema, como por exemplo a demanda de serviço gerada pelo processamento de uma página Web, pode prover modelos de comportamento mais acurados. Apesar desse fato, poucos trabalhos em eficiência energética em sistemas ou serviços de Internet têm como foco a caracterização do consumo de energia em um nível mais detalhado. Como consequência, a fim de atingir maior acurácia, grande parte dos modelos existentes na literatura procuram monitorar um grande número de métricas e contadores de desempenho, para então realizar algum mecanismo de poda que retorne as métricas ou contadores mais relacionados com o componente observado. Essa abordagem, no entanto, torna os modelos de energia mais complexos, além de gerar sobrecargas relacionadas à coleta, armazenamento e processamento dos dados coletados.

Neste trabalho, tratamos esse problema propondo um modelo analítico que captura o padrão de consumo de energia de sistemas Web na granularidade de páginas Web e para cada camada do sistema. Além disso, nosso modelo se baseia apenas na utilização de CPU e em parâmetros arquiteturais do servidor, os quais podem ser facilmente obtidos nos ambientes de produção atuais. Nosso modelo se baseia em um arcabouço proposto por Zhang et al. [62], o qual permite estimar demandas de CPU geradas pelo processamento de páginas Web em um hardware particular. Usamos então essas demandas para parametrizar um modelo tradicional de consumo de energia que, juntamente com um método de regressão linear, nos permite estimar o custo energético necessário para processar cada página individualmente.

Avaliamos a efetividade do nosso modelo utilizando um ambiente de experimentação real e o *benchmark* TPC-W [55]. Os resultados obtidos mostram que nosso modelo é capaz de estimar o consumo de energia de um sistema Web com um erro relativo médio

de 6,5% para o pior cenário, enquanto modelos mais complexos da literatura apresentam erros com a mesma ordem de grandeza.

1.1 Objetivo e contribuições

O objetivo geral deste trabalho é propor um modelo analítico para descrever o consumo energético requerido para processar cada tipo de transação que compõe um sistema Web. Para atingir esse objetivo geral, algumas atividades intermediárias foram definidas:

- Implantação de um ambiente de experimentação onde uma aplicação Web multi-camadas baseada no *benchmark* TPC-W foi instalada.
- Coleta e extração de dados de utilização de CPU das diversas camadas que compõem a aplicação.
- Coleta e extração de requisições processadas pelo servidor Web do ambiente implantado.
- Desenvolvimento de um modelo para descrever o consumo de potência e energia em processadores equipados com vários núcleos a partir da generalização de um modelo clássico de consumo de potência e energia em componentes CMOS (*Complementary Metal-Oxide-Semiconductor*).
- Desenvolvimento de um modelo para descrever o consumo de potência e energia requerido por cada tipo de transação que compõe um sistema Web a partir do modelo desenvolvido no item anterior.
- Comparação das estimativas produzidas pelo modelo proposto com estimativas reportadas por um modelo de referência implementado no hardware do processador.

Portanto, este trabalho contribui com os seguintes pontos para a literatura:

- Desenvolvimento de modelo analítico que estimar o consumo de energia para aplicações Web utilizando dados facilmente coletados em ambientes de produção.
- Avaliação do modelo em diferentes cenários, considerando quantidade de requisições, tipos de cargas de trabalho e janelas de monitoramento.
- Artigo aceito para ser publicado no XXXV Simpósio Brasileiro de Redes e Sistemas Distribuídos (SBRC).

1.2 Organização da dissertação

O restante do texto está organizado em mais quatro capítulos, conforme apresentado a seguir. O Capítulo 2 introduz alguns conceitos fundamentais sobre eficiência

energética em sistemas de computação. Esse capítulo também descreve alguns trabalhos relacionados a esta pesquisa. O Capítulo 3 apresenta o modelo de sistema considerado e descreve um modelo da literatura para estimar demandas de CPU em sistemas Web. Nesse capítulo detalhamos também a formulação do modelo de energia proposto neste trabalho. O ambiente de experimentação bem como o *benchmark* usado na avaliação são apresentados no Capítulo 4. Esse capítulo apresenta também a avaliação experimental e os resultados obtidos. Por fim, o Capítulo 5 sintetiza as contribuições do trabalho e discute perspectivas para os trabalhos futuros.

CAPÍTULO 2

Fundamentos e Trabalhos Relacionados

Neste capítulo, discutimos alguns problemas causados pelo consumo elevado de energia em sistemas de computação de grande escala (Seção 2.1). Em seguida, apresentamos uma terminologia para o projeto de sistemas computacionais conscientes do seu consumo energético, a qual é adotada ao longo deste trabalho (Seção 2.2). Discutimos também as principais iniciativas na indústria e na academia para tratar o problema (Seções 2.3 e 2.4). Concluímos o capítulo, descrevendo o escopo deste trabalho dentro desse cenário e discutindo os trabalhos existentes na literatura que compartilham objetivos semelhantes aos aqui apresentados (Seção 2.5).

2.1 O problema energético em sistemas de computação de grande escala

O consumo de energia sempre foi uma questão de projeto importante em sistemas, como redes de sensores e dispositivos portáteis limitados por bateria [21, 41, 61]. Ao longo dos anos, essa questão também se tornou importante em ambientes tipicamente orientados a desempenho, como *data centers* e ambientes de processamento de alto desempenho (PAD). Esse fato se deve em grande parte à elevação do custo da energia elétrica em todo o mundo, tornando o custo para alimentar um servidor empresarial típico, durante seu tempo de vida, maior que seu custo de aquisição [50]. Estima-se que em 2014, os *data centers* nos Estados Unidos consumiram aproximadamente 70 bilhões de kWh, o que representa aproximadamente 1.8% de toda a energia consumida no país [3].

Em ambientes de PAD, o consumo de energia elétrica também é uma das principais questões a serem tratadas, a fim de se construir a próxima geração de sistemas. Observando os supercomputadores que compõem a lista Top500[1] e dimensionando essas máquinas, de forma tradicional, para operarem com sistemas *ExaFlop*, resultaria em

[1] https://www.top500.org/

máquinas que consumiriam Gigawatts de energia. Essas máquinas exigiriam uma planta de energia nuclear de porte médio para alimentá-las [60].

Outro desafio associado ao consumo de energia é o aquecimento, o qual pode afetar severamente a operação normal dos sistemas. Estudos mostram que um aumento de 10^o C na temperatura do componente pode aumentar em até 50% as chances de falhas em servidores e discos [46]. Para prevenir falhas causadas por aquecimento, os sistemas necessitam de soluções de resfriamento e extração de calor que podem elevar ainda mais o custo de operação dos ambientes. No caso de *data centers* e ambientes de PAD, tais custos se traduzem em um consumo adicional de aproximadamente 1 watt para cada watt de potência consumida pelos recursos computacionais [47].

Finalmente, o consumo de energia em sistemas de grande escala também tem implicações no meio ambiente. Segundo o IPCC, o setor de energia é responsável por 25% do CO2 emitido no planeta [43]. Como *data centers* e ambientes de PAD são grandes consumidores de energia elétrica, uma parcela não negligenciável dessa emissão está relacionada à operação desses ambientes. Portanto, o grande adensamento de componentes eletrônicos e a dificuldade em dissipar o calor gerado por eles tornam a questão energética um problema de primeira ordem em ambientes de computação de grande escala.

De fato, a computação verde ganhou grande importância na última década. No entanto, sua origem data de 1992, quando o governo americano estabeleceu especificações para computadores pessoais energeticamente eficientes. Desde de então, o termo tem sido usado para designar o estudo e práticas de projeto, manufatura, utilização e descarte de computadores, servidores e subsistemas associados (como entrada/saída, armazenamento e comunicação) de forma eficiente e efetiva, e com impacto mínimo para o meio-ambiente [40]. Como as fontes de energia primária são limitadas em todo o mundo e dada a demanda atual crescente por tarefas que requerem computação, o comprometimento em reduzir o consumo de energia em sistemas computacionais se tornou crucial, especialmente em ambientes de computação de grande escala. Portanto, a eficiência energética se tornou um ponto central em computação verde. A seguir, apresentamos alguns conceitos importantes referentes a eficiência energética.

2.2 Fundamentos de eficiência energética

2.2.1 Energia e potência

Energia e potência são comumente definidas em termos do trabalho realizado por um sistema. Energia é a quantidade total de trabalho realizado pelo sistema durante um período de tempo, enquanto potência é a taxa com a qual o sistema executa o trabalho.

Em termos formais:

$$E = P \times T, \tag{2-1}$$

onde E denota a energia, P denota a potência e T representa um intervalo de tempo específico. Potência e energia são medidos, respectivamente, em watts (W) e joules (J), sendo $1J = 1W \times 1segundo$.

No contexto de sistemas de computação, o trabalho envolve atividades associadas com a execução de programas, enquanto a potência representa a taxa com a qual o computador consome energia elétrica para executar essas atividades. A energia, por sua vez, é a quantidade total de energia elétrica consumida pelo computador para executar as atividades ao longo do tempo.

Em geral, *data centers* são construídos levando em consideração duas métricas de potência: o consumo médio e o consumo máximo. O preço da energia elétrica paga mensalmente pelo *data center* depende da potência média consumida pelo ambiente naquele mês. Portanto, as técnicas que tentam reduzir o consumo médio de potência têm impacto direto no custo de operação do *data center*. Por outro lado, a potência máxima consumida é importante para estabelecer o limiar máximo suportado pela infraestrutura, como, por exemplo, o sistema de refrigeração, sistema de alimentação, etc. Quando esse limiar é excedido, uma nova infraestura (novo *data center*) deve ser planejada.

2.2.2 Consumo de potência estática e dinâmica

Em circuitos CMOS, o consumo de potência é proveniente de duas fontes: uma estática (P_{static}) e outra dinâmica($P_{dynamic}$) [57]. Logo, podemos reescrever a Equação 2-1 como:

$$E = T \times (P_{static} + P_{dynamic}). \tag{2-2}$$

O consumo estático é causado por "vazamentos"de correntes elétricas presentes em qualquer circuito ativo. Esse tipo de consumo não depende do cenário de utilização e está presente mesmo quando o componente está ocioso. Além disso, sua redução depende de melhorias no processo de fabricação do componente e na tecnologia utilizada. Portanto, não faz parte do escopo deste trabalho.

O consumo dinâmico, por outro lado, é proveniente de atividades nos circuitos elétricos e está relacionado principalmente com o cenário de utilização, a frequência do relógio, e atividades de entrada e saída. Formalmente, a potência dinâmica pode ser definida como:

$$P_{dynamic} = \alpha \times C \times V^2 \times f, \tag{2-3}$$

onde α representa um fator de atividade dentro do circuito, C denota a capacitância física, V é a voltagem (tensão de alimentação) e f é a frequência do relógio [57]. A técnica de

gerenciamento de potência denominada *Dynamic Voltage and Frequency Scaling* (DVFS) tem como base a redução combinada dos parâmetros V e f. A ideia principal consiste em diminuir o desempenho do processador reduzindo a voltagem e a frequência de operação quando o componente não está em sua utilização máxima. No caso ideal, isso poderia causar uma redução cúbica no consumo de potência dinâmica.

2.2.3 Eficiência energética

Eficiência energética (EE) de um sistema computacional pode ser definida como a razão entre desempenho, geralmente medido em termos de tempo de resposta ou vazão, e a potência consumida, ou seja:

$$EE = \frac{Desempenho}{P} \tag{2-4}$$

De acordo com a Equação 2-4, existem duas formas de se alcançar sistemas energeticamente eficientes: (1) melhorando o desempenho, consumindo a mesma potência; ou (2) reduzindo o consumo de potência sem sacrificar demasiadamente o desempenho. No contexto da segunda abordagem, um sistema é considerado energeticamente eficiente se ele é capaz de executar tarefas com o desempenho desejado e assegurar o menor consumo de energia possível [11]. Portanto, eficiência energética e desempenho não são objetivos conflitantes, uma vez que o tempo de resposta ou a vazão desejada, frequentemente podem ser alcançados desativando ou configurando dispositivos para utilizar um baixo consumo de recursos, dos quais não estão sendo utilizados em um determinado instante. De fato, vários autores tratam a eficiência energética em sistemas de computação como um problema de otimização no qual, para minimizar o consumo de energia, os sistemas precisam ajustar dinamicamente os recursos utilizados, de forma que apenas os recursos necessários para executar a tarefa em um dado instante são disponibilizados.

2.2.4 Fontes de consumo

A energia elétrica consumida pelos *data centers* é normalmente proveniente da malha elétrica tradicional. Atualmente, poucos *data centers* utilizam fontes de energia renovável para sua operação [12, 25]. Considerando a totalidade da energia consumida nesses ambientes, aproximadamente 50% é gasta pela infraestrutura de alimentação e pelo sistema de refrigeração, 10% é usado pelos equipamentos de rede e pelo sistema de armazenamento e os demais 40% são consumidos pelos servidores [1]. A energia consumida por um servidor, por sua vez, é a soma da energia consumida pelos seus subsistemas. A Tabela 2.1 ilustra a contribuição desses subsistemas para o consumo total de energia de um servidor típico, bem como o intervalo de variação para o consumo de

potência dinâmica de cada subsistema. Quanto maior o intervalo de variação, maiores as oportunidades de economia. Como podemos observar, o processador é a maior fonte de consumo de energia em um servidor, seguido pela memória e os *slots* PCI. Além disso, o processador possui também o maior intervalo de variação para o consumo de potência dinâmica. Devido a esses fatores, a maioria das técnicas que visam reduzir o consumo de energia nos sistemas de grande escala, têm como foco principal, o processador.

Componente	Contribuição para energia total	Intervalo para potência dinâmica
Processador	37.6%	70% da potência máxima
Memória	16.9%	50% da potência máxima
Disco	5.6%	25% da potência máxima
Slots PCI	23.5%	negligenciável
Placa Mãe	11.7%	negligenciável
Ventilador	4.7%	negligenciável

Tabela 2.1: *Detalhamento do consumo de energia de um servidor típico por componentes [44, 20].*

2.3 Técnicas para redução de consumo de energia

Embora metade da energia consumida pelos *data centers* provenha dos sistemas de alimentação e refrigeração, essa proporção tende a diminuir nos próximos anos devido à construção de novas infraestruturas em lugares frios ou próximos a rios, dos quais as águas podem ser utilizadas para refrigerar os servidores [5]. Como consequência, a parcela de consumo atualmente atribuída aos recursos computacionais deverá crescer, aumentando ainda mais a necessidade de técnicas que reduzam o consumo de energia desses recursos. De maneira geral, existem duas abordagens complementares para reduzir o consumo de energia dos recursos computacionais: I) a incorporação de técnicas de eficiência energética no projeto de hardware dos componentes; e II) o controle adaptativo do consumo de energia em resposta a flutuações no ambiente ou na carga de trabalho.

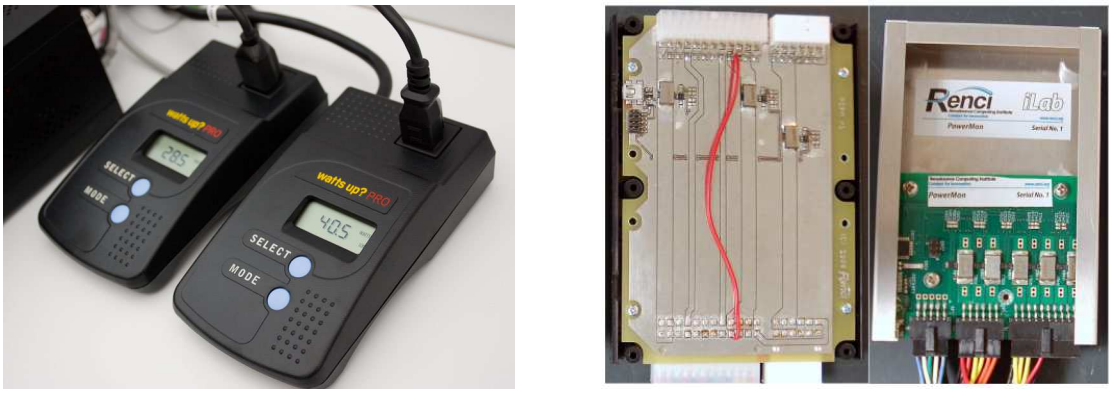
Na primeira abordagem encontram-se técnicas como DVFS e desativação de componentes (*clock gating*). Em DVFS, múltiplos níveis de voltagem e frequência são adicionados aos componentes do sistema, geralmente o processador, de forma que os diferentes níveis possam ser explorados para economizar energia. A desativação dinâmica de componentes, por sua vez, consiste em uma técnica para reduzir o consumo de potência dinâmica pela desativação do sinal do relógio para componentes ou partes de um componente que não estão sendo usados.

A segunda abordagem abrange técnicas como o consumo proporcional de potência, aprovisionamento dinâmico de servidores e consolidação de máquinas virtuais. A primeira técnica consiste em métodos inteligentes que escolhem, dinamicamente, os ní-

veis de voltagem e frequência de operação do processador para otimizar o desempenho e o consumo de potência dos servidores [4, 54]. A segunda técnica consiste em métodos capazes de ligar ou desligar servidores, de acordo com a demanda requerida pela carga de trabalho [15, 22]. Dessa forma, se a demanda aumenta, servidores adicionais são ligados; quando a demanda diminui, servidores subutilizados são desligados ou colocados em estado de baixo consumo. A última técnica é uma evolução natural da segunda em ambientes de computação em nuvem. Ela consiste em alocar máquinas virtuais, necessárias para atender uma certa demanda, no menor número possível de servidores físicos, contudo, sem violar a qualidade de serviço em comum acordo com o cliente [7, 58, 14, 34].

2.4 Técnicas para mensurar o consumo de energia

Na projeção e compreensão dos impactos de técnicas de redução de consumo de energia é necessário mensurar o consumo energético dos recursos computacionais ou dos sistemas. Essas mensurações podem ser realizadas através de medidores reais ou estimadas via modelos. O primeiro caso compreende medidores de corrente alternada ou contínua, como mostrado na Figura 2.1. Medidores de corrente alternada são dispositivos externos que medem a corrente alternada total utilizada para energizar o servidor, sendo geralmente implantados entre a máquina e a fonte de energia. Uma vantagem do uso desses dispositivos é que o processo de medição ocorre de forma não intrusiva. No entanto, medidores de corrente alternada não são capazes de mensurar o consumo de energia de componentes ou sub-sistemas individuais, ou seja, CPU, memória, etc. Dispositivos como WattsUp? e OmegaWatts são exemplos desses tipos de medidores.



(a) *Corrente alternada - Wattsup? PRO* (b) *Corrente contínua - PowerMon*

Figura 2.1: *Medidores de energia*

Medidores de corrente contínua, por outro lado, são dispositivos internos que medem a potência dissipada por componentes internos ao servidor. Portanto, eles fornecem o consumo de energia agregado por toda a máquina ou componente. Adicionalmente, me-

didores para corrente contínua são mais precisos que os de corrente alternada. Contudo, a implantação desses medidores requer modificações físicas em componentes elétricos (cabos e conectores), o que resulta na necessidade da troca dos componentes no decorrer do tempo de uso. PowerMon [6] e PowerInsight [33] são exemplos de medidores de corrente contínua, e são amplamente usados em ambientes de PAD.

Em geral, a implantação de medidores reais é um processo trabalhoso e dispendioso se realizado após a implantação e configuração da infraestrutura de servidores. Uma alternativa mais econômica é utilizar modelos para estimar o consumo energético dos componentes do sistema, ou de toda a infraestrutura. Modelos podem ser construídos para estimar o consumo de energia de *racks*, servidores, componentes (CPU, memória, disco), processos, serviços, aplicações, etc. Atualmente, modelos de energia são implementados tanto em hardware quanto em software, como discutido a seguir.

Running Average Power Limit (RAPL) [49], *Application Power Management* (APM) [2] e *NVIDIA Management Library* (NVML) [42] são exemplos de modelos de energia implementados em hardware, disponíveis em alguns equipamentos da Intel, AMD e NVIDIA, respectivamente. Nesses modelos, o consumo de energia é estimado através de contadores de desempenho localizados no *chip* do processador (RAPL e APM) e da GPU (NVML). Todavia, esses modelos estimam apenas a energia consumida pelo *chip* ao qual estão localizados, não sendo possível estimar o consumo de outros componentes do computador. Além disso, eles são projetados para uma arquitetura específica.

Neste trabalho, O RAPL é usado como um medidor de referência. Ele foi introduzido nos processadores Intel na tecnologia *Sandy Bridge*, podendo ser usado para limitar, medir e dar informações sobre o consumo de energia do processador. As informações são gravadas em registradores de máquina específicos (MSR), os quais podem ser lidos em modo núcleo. Atualmente, o RAPL reporta o consumo de energia dos seguintes componentes do processador: todo o *chip*; agregado de todos os núcleos; e controlador de memória. Uma desvantagem, no entanto, é a impossibilidade de estimar o consumo de energia por núcleo.

Devido às limitações dos medidores reais e dos modelos de energia implementados em hardware, na última década, muitos trabalhos foram propostos tendo como objetivo a construção de modelos de energia implementados em software. O objetivo desta dissertação é a proposta de um novo modelo de energia implementado em software capaz de estimar o consumo de sistemas multi-camadas Web na granularidade de transações. Na seção seguinte, discutimos os principais modelos de energia implementados em software propostos na literatura. Adicionalmente, nossa discussão se limita aos modelos centrados no processador, uma vez que esse é o alvo da nossa proposta, além de ser a fonte responsável pelo maior consumo de energia num servidor.

2.5 Trabalhos relacionados

Em ambientes de PAD, diversos trabalhos utilizam contadores de desempenho (PMC) para estimar o consumo de energia proveniente do processador. A grande maioria destes modelos se baseiam em regressão linear, por exemplo, Joseph e Martonosi [31] apresentam um modelo para estimar o consumo de energia do processador usando heurísticas baseadas em modelos de capacitância. Isci e Martonosi [28] expressam a energia consumida por um Pentium IV como a soma do consumo das subunidades mais importantes dos processadores. Bertran et al. [9], posteriormente, refinam esta ideia e desenvolveram modelos baseados em PMC para processadores com múltiplos núcleos. Entretanto, todos estes trabalhos focaram em caracterizar o consumo energético para cargas com uso intensivo da CPU (PAD), as quais, diferem significantemente de cargas Web. Além disso, a maioria destes trabalhos necessitam de 15 PMCs ou mais, dos quais, muitos não estão disponíveis simultaneamente na maioria dos processadores modernos. Diferentemente, nossa proposta estima o consumo de energia para um processador utilizando somente dois parâmetros facilmente encontrados em ambientes de produção.

O consumo de energia proveniente do processamento de cargas Web e do uso de servidores empresariais já foram o objetivo de trabalhos anteriores [10, 45, 59, 53, 37, 35]. Bohrer et al. [10] apresentaram uma ferramenta de simulação para caracterizar o consumo de servidores Web. Ela estima o tempo de CPU e a energia consumida baseando-se em dados de requisições Web e ciclos da CPU. Contudo, a caracterização está centrada somente no servidor de aplicação, cujas requisições são supostamente servidas integralmente pela memória, ou integralmente pelo disco. Diferentemente, o nosso modelo leva em consideração os objetos que são recuperados ao longo de todas as camadas de uma aplicação Web. Consequentemente, ele é capaz de caracterizar a demanda de CPU e o consumo de energia das requisições em cada camada.

Outro trabalho com caracterização de carga Web é de Liu et al. [35]. Os autores analisaram qual é a melhor configuração para obter eficiência energética com servidores Web implantados em múltiplas camadas. Para isso, os autores fizeram experimentos com dois tipos de *clusters*, um montado com servidores homogêneos e outro com servidores heterogêneos. Além disso, separaram os servidores em grupos de mais e menos potência. Eles concluíram que os servidores mais potentes para a camada de aplicação e os menos potentes para a camada de dados tem o melhor balanço entre eficiência e consumo energético. Contudo, os autores apresentam somente uma análise experimental, sem apresentar nenhuma formulação para caracterizar ou estimar o consumo de energia em sistemas Web.

Ainda no contexto de caracterização de servidores Web, Piga et al. [45] estimaram o consumo energético de sistemas Web através de um conjunto de métodos, incluindo

regressão linear, *CFS* e *K-means*. Esses métodos foram aplicados sobre dados coletados de eventos de hardware e métricas no nível do sistema operacional. O Capítulo 3, irá mostrar que o nosso modelo é mais simples que o proposto por Piga et al., pois utilizamos apenas um método de análise de dados, a regressão linear. Nosso modelo também é mais simples em relação ao conjunto de dados coletados, uma vez que nos baseamos apenas na utilização de CPU dos núcleos do processador. Entretanto, como mostraremos no Capítulo 4, mesmo sendo mais simples, nosso modelo produz erros na mesma ordem de grandeza que o proposto por Piga et al.. Os erros estão na mesma ordem de grandeza pois o nosso modelo se baseia nas demandas de serviços de um componente mais fino do sistema, as quais são geradas pelo processamento de uma transação Web individualmente.

Além dos trabalhos acima apresentados, alguns autores estudaram os efeitos de técnicas de gerenciamento de energia no desempenho de servidores Web. Wang et al. [59] apresentaram um estudo empírico abrangente sobre o impacto de DVFS no desempenho de sistemas Web. Contudo, os autores não propuseram nenhum modelo para estimar o consumo de energia dos servidores. Subramaniam and Feng [53] introduziram um estudo empírico sobre o impacto do uso de mecanismos limitadores de energia no desempenho de servidores Web. Os autores propuseram um modelo para capturar a relação entre desempenho e limiares de energia impostos ao sistema. Contudo, como em Bohrer et al. [10], o trabalho foi centrado apenas na camada de aplicação. Por fim, Metri et al. [37] caracterizaram o consumo de energia de diferentes tipos de aplicações, incluindo o TPC-W, o benchmark utilizado neste trabalho. Contudo, o estudo é centrado em métricas de granularidade grossa, e o objetivo dos autores se limita a dar uma ideia de como diferentes tipos de carga e ambientes, comuns em computação em nuvem, impactam na eficiência energética dos centro de dados.

Em um trabalho recente, Varghese et al. [56] investigaram a viabilidade e os benefícios, em termos de eficiência energética, de se usar um *cluster* de dispositivos embarcados para hospedar serviços Web. Os autores observaram que, nesses ambientes, o consumo de energia pode ser de 17 a 23 vezes menor que o observado em *clusters* de servidores convencionais. Todavia, os melhores resultados foram obtidos para cargas Web estáticas. Os autores observaram que, para cargas dinâmicas, ainda existem desafios a serem explorados. Além disso, os autores não consideraram a camada de dados, a qual normalmente é a mais exigida nesse tipo de sistema.

2.6 Considerações finais

Podemos observar que o consumo energético de servidores é a soma da energia consumida pelos seus componentes, sendo o processador o responsável pela maior parte do consumo. Portanto esse componente é o alvo mais explorado em trabalhos que

buscam reduzir o consumo de energia em ambientes de computação de grande escala. Discutimos também soluções existentes na literatura para reduzir o consumo de energia nesses ambientes, bem como técnicas para mensurar ou estimar a energia consumida por servidores. Ressaltamos que o escopo deste trabalho se refere à técnicas de estimação, mais especificamente, um modelo de consumo de energia implementado em software. Esse modelo, por sua vez, pode ser usado por técnicas de redução de consumo de eneriga para guiar decisões de alocação de recursos. No capítulo seguinte, descreveremos em detalhe o modelo proposto.

CAPÍTULO 3

Proposta de um Modelo para Estimar o Consumo de Energia em Sistemas Web

Neste capítulo, propomos um modelo analítico baseado na técnica de regressão linear para aproximar, de forma acurada, a energia requerida para processar diferentes transações de um sistema Web. Primeiramente, introduzimos alguns fundamentos da técnica de regressão linear (Seção 3.1). Em seguida, apresentamos o sistema alvo considerado neste trabalho (Seção 3.2). Na Seção 3.3, introduzimos um arcabouço teórico proposto na literatura para estimar demandas de CPU por tipo de transações que compõem um sistema Web. Finalmente, as Seções 3.4 e 3.5 mostram como usamos esse arcabouço teórico e a técnica de regressão linear para estimar o consumo de energia de sistemas Web através de componentes mais finos do sistema.

3.1 Regressão linear

O trabalho de análise de dados exige ferramentas como métodos estatísticos ou técnicas de aprendizagem de máquina. A regressão linear é um método estatístico utilizado para estimar valores de uma variável de resposta Y com base nos valores de uma variável única de predição X. Mais especificamente, a regressão linear assume que existe um relacionamento aproximadamente linear entre X e Y, ou seja:

$$Y \approx \beta_0 + \beta_1 X, \tag{3-1}$$

onde β_0 e β_1 são constantes desconhecidas que representam, respectivamente, a interceptação e a inclinação da reta que descreve o modelo linear. Essas constantes também são chamadas de coeficientes ou parâmetros do modelo.

Na prática, dado um conjunto (x_1, y_1), (x_2, y_2) ... (x_n, y_n) representando n pares de observações, dos quais, cada um consiste em um valor para X e outro para Y, o objetivo da regressão linear é calcular estimativas para os coeficientes β_0 e β_1, denotados por β_0' e β_1' respectivamente, de forma que o modelo linear da Equação 3-1 represente o

conjunto de dados da melhor forma possível, ou seja, de forma que $y_i \approx \beta'_0 + \beta'_1 x_i$, para todo i=1,...,n [30]. Existem diversas formas de se alcançar esse objetivo, porém, a mais comumente empregada, consiste em minimizar a soma residual dos quadrados, a qual é dada por:

$$RSS = e_1^2 + e_2^2 + \cdots + e_n^2, \tag{3-2}$$

onde $e_i = y_i - y'_i$ representa o i-ésimo resíduo, ou seja, a diferença entre o i-ésimo valor observado para Y (y_i) e o i-ésimo valor estimado pelo modelo (y'_i).

A acurácia de um modelo de regressão linear pode ser avaliada através de diferentes métricas, incluindo o erro padrão residual (RSE), a estatística R^2 e o erro relativo médio [30]. Neste trabalho, optamos por usar o erro relativo médio, calculado como a média dos erros relativos da regressão. Para um conjunto de n pares de observações (x_i, y_i), o erro relativo é calculado usando a Equação 3-3. Optamos por essa métrica, a fim de permitir a comparação com outros trabalhos da literatura que também a utilizam, a saber Piga et al. [45] e Zhang et al. [62].

$$\varepsilon = \frac{|y'_i - y_i|}{y_i} \tag{3-3}$$

Adicionalmente, a técnica de regressão linear pode ser estendida de forma a acomodar mais de uma variável de predição. Nesse caso, a técnica é normalmente denominada, regressão linear múltipla. Dadas p variáveis de predição, a regressão linear múltipla toma a seguinte forma:

$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p, \tag{3-4}$$

onde, $1 \leq i \leq p$, X_i representa a i-ésima variável de predição e β_i quantifica a associação entre a variável de resposta Y e X_i. Como ocorre na regressão linear simples, os coeficientes β_i são estimados minimizando a soma residual dos quadrados (RSS).

3.2 Modelo de sistema

Neste trabalho, o sistema alvo é um provedor de IaaS (*Infrastructure as a Service*) representado por um *data center* consistindo de vários servidores físicos heterogêneos. Os servidores são equipados com vários núcleos de processamento e são caracterizados pelo número de núcleos, a quantidade de memória RAM disponível e um parâmetro P_{TDP} (*thermal design power*) que especifica a potência máxima que o processador pode lidar.

Definimos os principais participantes desse ambiente como clientes, clientes IaaS e provedor IaaS. Clientes são os usuários finais dos sistemas Web hospedados no *data center*. Clientes IaaS são as companhias que alocam a infraestrutura do *data center* para prover um serviço online, enquanto o provedor IaaS são os proprietários desta infraestrutura.

A carga de trabalho é formada por aplicações Web, normalmente concebidas como sistemas multi-camadas (*multi-tier systems*). Cada camada fornece uma funcionalidade, incluindo apresentação de dados, lógica da aplicação e gerenciamento de dados. Os clientes interagem com tais aplicações através de uma interface Web ou navegador, onde as atividades no cliente local são medidas em unidades de páginas Web recuperadas. Uma página Web consiste em um arquivo HTML e vários objetos embutidos, como por exemplo imagens, áudio, vídeo, *applets Java*, etc. Para recuperar a página, o navegador emite uma série de solicitações HTTP para o servidor Web: a primeira solicitação para recuperar o arquivo HTML principal e as demais para obter os objetos embutidos.

O servidor Web e o servidor de aplicação estão hospedados no mesmo hardware e compartilham os mesmos recursos. A recuperação de uma página Web envolve o processamento de vários objetos menores. Se a recuperação requer acesso a dados, o servidor de aplicação encaminha a solicitação para a camada de dados, a qual executa um servidor de banco de dados responsável por manipular o acesso e o processamento de dados.

Referimos à combinação de todas as atividades de processamento no lado do servidor para entregar uma página Web para um cliente como uma **transação**. Isto inclui as operações para gerar o arquivo HTML principal, bem como para recuperar os objetos embutidos e executar consultas de banco de dados. Dessa forma, para cada tipo de página Web gerada por um sistema, existe um tipo de transação específica a ser executada para permitir a geração dessa página, criando, portanto um relacionamento 1 para 1 entre tipo de página Web e tipo de transação. O acesso do cliente a uma aplicação Web ocorre sob a forma de uma sessão, a qual consiste em várias transações individuais.

Para atender vários clientes IaaS ao mesmo tempo, os recursos de computação são dimensionadas em termos de máquinas virtuais (VMs). Para esse fim, o provedor IaaS oferece diferentes tipos de instâncias de VMs, cada uma delas sob diferentes combinações de núcleos de CPU e capacidades de memória RAM. Clientes IaaS escolhem o tipo de instância apropriada para as suas aplicações e estabelecem acordos de nível de serviço (SLAs), tais como limites mínimos para vazão e limites máximos para tempos de resposta. Como as camadas de um sistema Web têm necessidades de recursos de computação significantemente diferentes, cada camada é provisionada como uma VM a fim de satisfazer o SLA em comum acordo com o cliente.

A seguir, descrevemos nossa proposta para estimar o consumo de energia de

servidores hospedados em ambientes que seguem o modelo de sistema descrito acima. Como mencionado anteriormente, nossa proposta é centrada no processador, uma vez que esse componente é o responsável pela maior parte da energia consumida nos servidores típicos. Esta decisão também é baseada em um trabalho anterior [19], o qual mostrou que, ao executar cargas de trabalho Web, disco, rede e memória geram quase o mesmo consumo de energia que seria gerado se o servidor estivesse ocioso.

3.3 Demanda de CPU requerida por uma transação Web

Zhang et al. [62] propuseram um modelo analítico simples e acurado para modelar cargas Web usando informações no nível de transações. A ideia básica consiste em usar o conhecimento da demanda de CPU de cada tipo de transação que compõe um sistema Web para então compor a demanda de CPU de cargas de trabalho contendo diferentes combinações desses tipos de transações. Assumindo um sistema Web de n camadas é composto por N tipos de transações diferentes, os autores aplicam a Lei da Utilização [36, 29], obtendo a seguinte equação:

$$\sum_{i=1}^{N} \eta_i \times D_{i,j} = U_{CPU,j} \times T, \tag{3-5}$$

onde T é o tamanho da janela de monitoramento; η_i, $1 \leq i \leq N$, é o número de transações do tipo i observadas durante o período T; $U_{CPU,j}$, $1 \leq j \leq n$, é a utilização média de CPU na camada j durante o período de tempo T; e $D_{i,j}$ é o tempo médio de serviço de CPU requerido para processar uma transação do tipo i na camada j.

Entretanto, como é difícil obter tempos de serviço de forma acurada em ambientes de produção, os autores propõem uma aproximação e substituem $D_{i,j}$ pelo custo de CPU. Esse custo, denotado por $C_{i,j}$, quantifica a porcentagem de utilização de CPU requerida para processar uma transação do tipo i na camada j. Assim, para cada camada j, uma utilização média aproximada de CPU, denotada por $U'_{CPU,j}$, pode ser calculada como:

$$U'_{CPU,j} = \frac{\sum_{i=1}^{N} \eta_i \times C_{i,j}}{T}. \tag{3-6}$$

A Equação 3-6 nos permite concluir que, se observarmos as páginas Web processadas durante janelas de monitoramento T (e, por conseguinte, as transações processadas para gerar tais páginas), bem como a utilização $U_{CPU,j}$ em cada camada do sistema durante essas janelas, podemos aproximar $U_{CPU,j}$ de $U'_{CPU,j}$ e empregar uma técnica de regressão linear múltipla para estimar os valores $C_{i,j}$. Nas seções a seguir,

mostraremos como usamos essa informação para estimar o consumo de energia requerido por cada tipo de transação de um sistema Web, em cada camada do sistema.

3.4 Demanda de energia requerida por um servidor com múltiplos núcleos

Como mencionado no Capítulo 2, em circuitos CMOS, o consumo de potência deriva de duas fontes principais: uma estática, denominada P_{static}, e uma dinâmica, representada por $P_{dynamic}$. Para estimar a potência consumida por um processador com múltiplos núcleos, estendemos uma abordagem tradicionalmente empregada [28], a qual consiste em aproximar $P_{dynamic}$ de um fator α da potência ativa. Essa última, denotada por P_{active}, representa a diferença entre a potência máxima que o processador pode lidar (*thermal design power*), representado por P_{TDP}, e P_{static} ou seja:

$$P_{dynamic} \approx \alpha \times P_{active}. \tag{3-7}$$

$$P_{active} = P_{TDP} - P_{static}. \tag{3-8}$$

Tipicamente, os parâmetros P_{TDP} e P_{static} podem ser facilmente obtidos a partir de especificações do hardware e medições reais, respectivamente. Usamos a utilização média de CPU durante um intervalo de tempo T para descrever o fator de atividade α, normalizado por um fator que representa a utilização máxima do processador, ou seja:

$$\alpha = \frac{U_{CPU}}{MAX_{U_{CPU}}}. \tag{3-9}$$

Em nosso modelo de sistema, assumimos que os servidores são equipados com processadores com múltiplos núcleos. Dessa forma, modelamos a utilização média de um processador com m núcleos no intervalo T como sendo a soma da utilização média de cada núcleo (U_{core_k}) durante T, isto é:

$$U_{CPU} = \sum_{k=1}^{m} U_{core_k}. \tag{3-10}$$

Modelamos a utilização máxima de um processador ($MAX_{U_{CPU}}$) como o produto do número de núcleos (m) e a utilização máxima de cada núcleo (100). Portanto, usando as Equações 2-2, 3-7 e 3-9, modelamos o consumo total de energia de um servidor com um processador com m núcleos como:

$$E = T \times (P_{static} + \frac{\sum_{k=1}^{m} U_{core_k}}{100m} \times P_{active}). \tag{3-11}$$

Por simplicidade, assumimos que um sistema Web, com n camadas e formado por N tipos de transações, é implantado em um único servidor físico equipado com m núcleos de processamento. Assumimos também que cada camada j do sistema é implantada em uma VM diferente e cada VM é instanciada com um número fixo m_j de núcleos. Para capturar o padrão de consumo dos recursos computacionais ao longo do tempo, observamos o sistema durante janelas de monitoramento de tamanho T. Ao final de cada janela de monitoramento, gravamos as seguintes informações:

- η_i: o número de transações do tipo i processadas durante a janela de monitoramento, $1 \leq i \leq N$;
- $U_{CPU,j}$: a soma da utilização média de CPU de todos os núcleos de uma camada j durante a janela de monitoramento, $1 \leq j \leq n$;

3.5 Demanda de energia requerida para processar um tipo de transação

Usando a Equação 3-11, definida na seção anterior, e abstraindo a camada j como um servidor com m_j núcleos, podemos calcular a energia consumida pela camada j em uma janela de tempo T como:

$$E_j = T \times (P_{static,j} + \frac{\sum_{k_j=1}^{m_j} U_{core_{k_j}}}{100 m_j} \times P_{active,j}), \tag{3-12}$$

onde:

$$P_{static,j} = m_j \times \frac{P_{static}}{m}, \qquad P_{active,j} = m_j \times \frac{P_{active}}{m} \quad \text{e} \quad \sum_{k_j=1}^{m_j} U_{core_{k_j}} = U_{CPU,j}.$$

Assumindo que toda atividade realizada em camadas de um sistema Web é decorrência apenas do processamento de transações, podemos calcular a energia requerida para processar uma transação nessas camadas como:

$$E_{\tau_i,j} = T_{i,j} \times \frac{C_{i,j}}{100 m_j} \times P_{active,j}, \tag{3-13}$$

Os valores de $C_{i,j}$ e $T_{i,j}$ são a utilização de CPU e o tempo médio necessários para processar uma transação do tipo i na camada j, respectivamente. Como dentro de uma janela de monitoramento T o servidor processa η_i transações do tipo i, a energia consumida por esse tipo de transação na camada j durante T é $\eta_i \times E_{\tau_i,j}$. Logo, a soma da energia consumida por todas as transações, de todos os tipos na camada j durante

T, resulta na energia proveniente do consumo dinâmico dessa camada. Dessa forma, podemos reescrever a Equação 3-12 como:

$$E_j = T \times P_{static,j} + \sum_{i=1}^{N} \eta_i \times E_{\tau_{i,j}}. \tag{3-14}$$

Combinando as Equações 3-13 e 3-14, obtemos a Equação 3-15 para cada janela de monitoramento.

$$E_j = T \times P_{static,j} + \sum_{i=1}^{N} (\eta_i \times T_{i,j} \times \frac{C_{i,j}}{100 m_j} \times P_{active,j}). \tag{3-15}$$

Na Equação 3-15, os parâmetros $P_{static,j}$ e $P_{active,j}$ são calculados a partir dos parâmetros arquiteturais P_{static} e P_{active}. Como $U_{CPU,j}$ é gravado no final de cada janela de monitoramento, E_j pode ser estimado facilmente usando a Equação 3-12. Como η_i também é gravado no final de cada janela de monitoramento, $T_{i,j}$ e $C_{i,j}$ são as variáveis desconhecidas. Contudo, como gravamos a soma da utilização média de CPU de todos os núcleos de uma camada j ao final de cada janela de monitoramento ($U_{CPU,j}$), podemos facilmente calcular $C_{i,j}$ utilizando o arcabouço descrito na Seção 3.3.

Uma vez estimados os valores $C_{i,j}$, podemos substituí-los na Equação 3-15, para então aplicar um método de regressão linear múltipla e calcular os valores $T_{i,j}$. Finalmente, substituindo $C_{i,j}$ e $T_{i,j}$ por seus respectivos valores na Equação 3-13, podemos estimar a energia requerida para processar cada tipo de transação em cada camada de um sistema Web.

3.6 Considerações Finais

Este capítulo apresentou o modelo analítico proposto para estimar a energia consumida por sistemas Web no nível de transações que compõem o sistema. Primeiramente, abordamos os conceitos estatísticos necessários para a compreensão do modelo proposto. Em seguida, expomos o modelo de sistema considerado neste trabalho. Finalmente, apresentamos a formulação para o modelo proposto. No capítulo seguinte, avaliaremos o modelo proposto, comparando-o com dados reais coletados experimentalmente.

CAPÍTULO 4

Avaliação e Análise dos Resultados

Neste capítulo, apresentamos uma avaliação para a acurácia do modelo proposto no Capítulo 3. Primeiramente, descrevemos o ambiente de experimentação implantado para realizarmos a avaliação (Seção 4.1). Em seguida, apresentamos uma análise dos dados coletados (Seção 4.1.2). Por fim, apresentamos a avaliação da acurácia do modelo proposto (Seção 4.3).

4.1 Ambiente de experimentação

Nosso ambiente de experimentação é baseado no TPC-W [55], um *benchmark* que emula um *site* típico de comércio eletrônico seguindo uma arquitetura em multi-camadas. A Figura 4.1 ilustra o ambiente implantado, o qual inclui uma camada de aplicação (`APP VM`) baseada no servidor Apache Tomcat, uma camada de dados (`DB VM`) implementada pelo servidor MySQL e uma máquina virtual para o emulador de cargas (`Emulator VM`) contendo o módulo TPC-W que emula os clientes do *site*. Cada camada é implementada em uma VM diferente. As camadas de aplicação e dados foram hospedadas no mesmo servidor físico (`Servidor 1`), o qual possui um processador Intel Xeon E5-2620 v3 2.4 GHz com 16 GB de memória RAM. A camada de apresentação foi implantada em outro servidor (`Servidor 2`), equipado com um processador Intel Xeon E5-2420 v2 2.2 GHz com 32GB de memória RAM.

Neste trabalho, utilizamos uma implementação em Java de código aberto do TPC-W. Todas as VMs e servidores executam o sistema operacional Linux com a distribuição Ubuntu 14.04. O *hypervisor* instalado nos servidores é o Xen 4.4.1. As máquinas virtuais (`APP VM e DB VM`) e o Dom0 do `Servidor 1` também executam a ferramenta collectd [17] para realizar as coletas de dados, ocultamos o Dom0 na Figura 4.1 para melhor visualização. A Tabela 4.1 ilustra as principais configurações das máquinas virtuais empregadas. Para melhor emular o comportamento de um sistema real, elas seguem configurações de instâncias Amazon EC2 [1].

[1] https://aws.amazon.com/pt/ec2/instance-types/

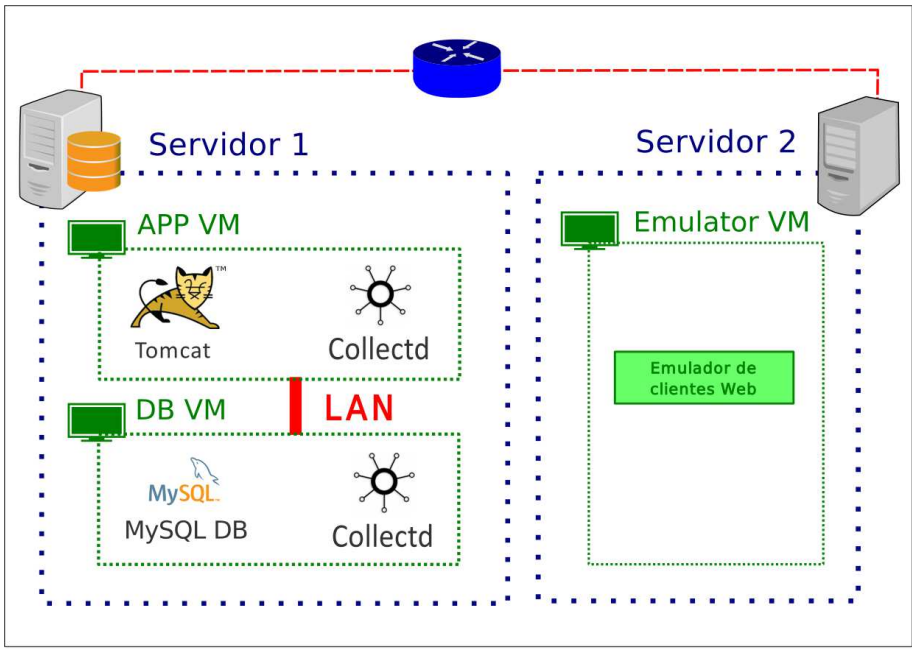


Figura 4.1: *Ambiente de experimentação.*

VM	SO	HD	RAM	CPU	Tipo EC2
APP	Ubuntu 14.04	50 Gb	4 Gb	2 vcpu	t2.medium
DB			4 Gb	2 vcpu	t2.medium
Emulator			2 Gb	1 vcpu	t2.small

Tabela 4.1: *Configuração de software e hardware das máquinas virtuais utilizadas nos experimentos.*

4.1.1 Benchmark TPC-W

Nosso ambiente de experimentação é baseado no *benchmark* TPC-W [55], uma iniciativa desenvolvida pelo *Transaction Processing Performance Council* (TPC). O TPC-W emula um *site* típico de comércio eletrônico seguindo uma arquitetura em múltiplas camadas.

O site é composto por 14 tipos de páginas Web (e, por conseguinte, 14 tipos de transações diferentes) que são classificadas como requisições relacionadas com a navegação de clientes no *site* ou como requisições de compras de produtos. A Tabela 4.2 mostra a lista de todas as páginas e suas respectivas classificações.

O TPC-W emula clientes Web através de emuladores denominados *Remote Browser Emulators* (RBEs). De acordo com as especificações do *benchmark*, durante um experimento, o número de RBEs que executam de forma concorrente é mantido constante e o TPC-W define para cada RBE, de forma estatística, o tamanho da sessão (em média 15 minutos), o tempo entre ações do usuário (*think time*, em média 7 segundos) e as consultas que o RBE realiza.

As páginas Web mostradas na Tabela 4.2 são combinadas para compor três tipos

Navegação	Compra
Home	Shopping Cart
New Products	Customer Registration
Best Sellers	Buy Request
Product detail	Buy Confirm
Search Request	Order Inquiry
Search Results	Order Display
	Admin Request
	Admin Confirm

Tabela 4.2: *Páginas do TPC-W de acordo com cada tipo de requisição.*

de carga de trabalho, a saber:

- *Browsing* (95% de navegação e 5% de compra)
- *Shopping* (80% de navegação e 20% de compra)
- *Ordering* (50% de navegação e 50% de compra)

Tipicamente, cargas Web não são limitadas por CPU, uma vez que sistemas Web tendem a executar um grande número de operações de entrada/saída [59]. Em geral, a maioria dessas operações ocorrem na camada de dados e, portanto, essa tende a ser a camada mais requisitada. A carga *Browsing* simula um cenário onde existem poucos compradores e várias requisições de busca ou pesquisa. Por outro lado, a carga *Ordering* simula um cenário com um número de pedidos e compras significantemente maior que a *Browsing*. *Shopping*, por sua vez, é uma carga de trabalho intermediária em relação às duas cargas anteriores.

4.1.2 Metodologia de experimento

Em nossos experimentos, utilizamos os três tipos de carga do TPC-W. Além disso, para emular intensidades de cargas diferentes, executamos um conjunto de experimentos variando o número de RBEs concorrentes em 100, 200, 300, 400 e 500. Variamos também o tamanho da janela de monitoramento, a qual assumiu valores de 1, 5, 10 e 15 minutos. Essas configurações nos permitiram avaliar o modelo proposto em diferentes cenários, os quais são apresentados na Tabela 4.3.

Cenários	Quantidade de RBEs	Janela de monitoramento
Browsing	100 - 500	1 minuto, 5, 10 e 15 minutos
Shopping		
Ordering	100 - 300	

Tabela 4.3: *Cenários dos experimentos.*

Como mostrado na Tabela 4.3, o número máximo de RBEs concorrentes utilizados com a carga *Ordering* é 300. A partir desse número de emuladores, o número de itens do banco de dados (10.000) não é suficiente para atender as requisições emitidas, forçando o servidor Web a abortar as requisições. Por esse motivo, o cenário *Ordering* utilizado nos nossos experimentos tem limite máximo com 300 RBEs. Para os cenários *Browsing* e *Shopping* observamos um comportamento semelhante para quantidades de RBEs maiores que 500.

Em cada cenário, executamos cada experimento por 5 horas e 40 minutos. Fixamos este tempo para facilitar a comparação dos nossos resultados com o trabalho de Zhang et al. [62] para o modelo de demanda de CPU. Além disso, os primeiros e últimos 20 minutos de cada experimento foram considerados períodos de aquecimento e resfriamento, respectivamente, portanto, esses períodos foram omitidos de nossa análise.

Durante a execução dos experimentos, coletamos e gravamos os valores $U_{CPU,j}$ e η_i a cada minuto. O primeiro valor foi obtido a partir da ferramenta `collectd` instalada em cada VM, enquanto o segundo foi extraído das requisições HTTP registradas no arquivo de *log* do Tomcat. Coletamos também a soma da utilização média de CPU de todos os núcleos do `Servidor 1` que não estão sendo usados por nenhuma camada do sistema ($U_{CPU,0}$), bem como o consumo médio de energia pelo processador durante a janela de monitoramento (E). Essa última métrica foi coletada usando a interface RAPL. É importante ressaltar que, embora $U_{CPU,0}$ e E não sejam parâmetros usados pelo modelo, suas coletas foram necessárias para a validação do modelo, como explicaremos futuramente.

Mantivemos a frequência de coleta da utilização de CPU e de η_i a cada minuto, enquanto a coleta de energia foi realizada a cada 15 segundos. Esse valor menor para a coleta de energia foi escolhido para evitar a sobrescrita dos contadores usados pela interface do RAPL. Assim, para obter a medição de energia dentro de uma janela de 1 minuto, agregamos 4 janelas consecutivas. Por outro lado, os dados para as janelas de montoramento maiores que 1 minuto foram obtidos agregando os dados da janela de 1 minuto. Por exemplo, os dados do cenário *Browsing*, 100 RBEs e janela de 5 minutos, foram obtidos agrupando 5 linhas consecutivas dos dados do cenário *Browsing*, 100 RBEs e janela de 1 minuto. Escolhemos a janela de um 1 minuto como sendo a menor janela, devido ao objetivo de realizar a comparação dos resultados obtidos entre o trabalho de Zhang et al.[62] com os resultados obtidos pelo nosso modelo. Além disso, estamos cientes que o QOS de aplicações Web está limitado a casa de milissegundos. Contudo, quanto menor a janela, maior a sobrecarga do sistema de monitoramento, consequentemente, maior o *overhead* do modelo.

Por fim, em cada cenário, executamos cada experimento duas vezes, resultando em 10 horas de execução após a remoção do período de aquecimento e resfriamento. Os

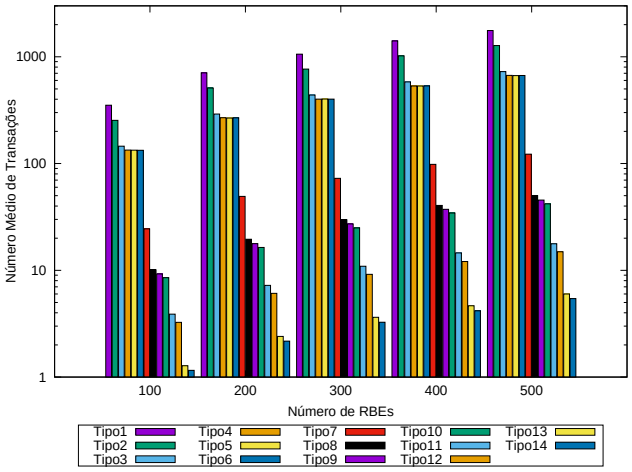
dados gerados pela primeira execução foram usados para treinar o modelo, enquanto os provenientes da segunda execução foram usados para os testes das regressões.

4.2 Análise de dados

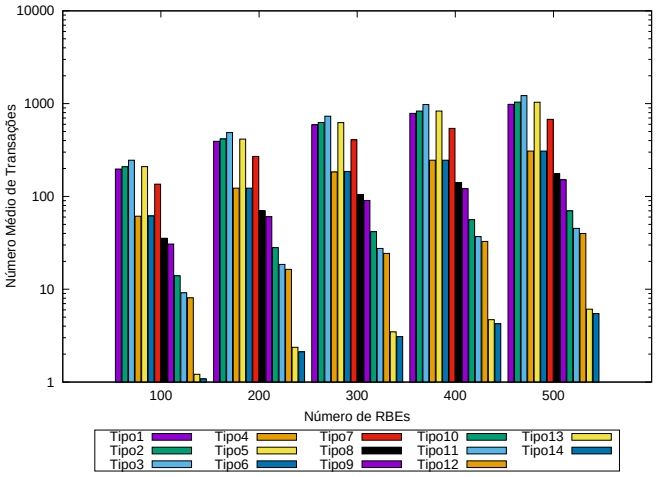
Analisamos os dados gerados pelos experimentos para compreender o comportamento dos dados coletados. Primeiramente, observamos a quantidade de transações do tipo i executada em uma janela de montioramento T, ou seja, o parâmetro η_i. As Figuras 4.2(a), 4.2(b) e 4.2(c) mostram a distribuição do número de requisições processadas para cada um dos 14 tipos de transação do TPC-W para os cenários *Browsing*, *Shopping* e *Ordering* de experimentação, respectivamente, para janelas de 1 minuto. Mostramos a distribuição por número de RBEs de cada experimento para cada cenário. Como esperado, dentro de um mesmo cenário, a distribuição do número de requisições por tipo de transação é a mesma, no entanto, a quantidade de requisições por tipo aumenta à medida que o número de RBEs aumenta. Como esperado também, para cada cenário, a distribuição do número de requisições por tipo de transação é diferente, uma vez que eles se diferem pelos tipos de cargas usadas nos experimentos, das quais são formadas por diferentes proporções de páginas do tipo `Compra` e `Navegação`.

Observamos também como a utilização de CPU varia na camada de aplicação e de dados à medida que o número de RBEs concorrentes aumenta. As Figuras 4.3(a) e 4.3(b) mostram essa variação para os três cenários, para as camadas de aplicação e dados, respectivamente. Nessas figuras consideramos dados dos cenários com uma janela de monitoramento de 1 minuto. Conforme esperado, a utilização de CPU é menor para a camada de aplicação. Na camada de dados, o uso de CPU é maior e apresenta maior variação. Podemos notar que o cenário que possui o maior o número de requisições do tipo `Compra`, demanda maior quantidade de CPU na camada de dados. Assim, nessa camada, o cenário *Browsing* requer uma demanda por CPU menor que o *Shopping*, que por sua vez, requer uma demanda menor que o *Ordering*. Para janelas maiores que 1 minuto, observamos comportamento semelhante. Portanto, apresentamos apenas o resultado para a janela de 1 minuto.

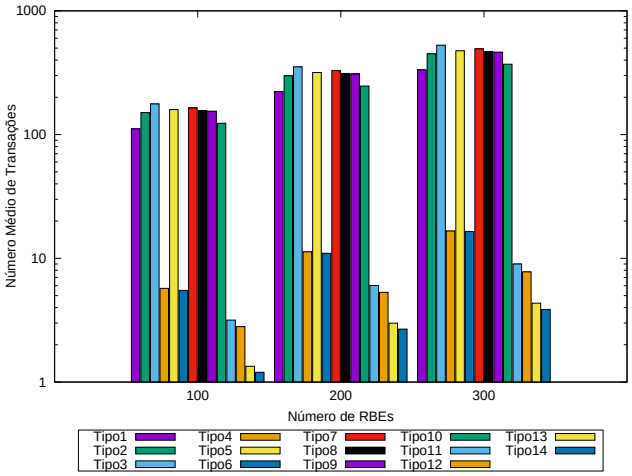
A Figura 4.4 ilustra melhor essa diferença de variação de utilização de CPU na camada de aplicação e dados. Nessa figura, ilustramos um cenário específico para a carga *Ordering*, 300 RBEs e uma janela de monitoramento de 1 minuto. Podemos observar que a camada de aplicação não utiliza mais que 10% da CPU ao longo de todo o experimento. Na camada de dados, a utilização de CPU varia entre 18% a 100%. Essa variação é explicada, na maioria das vezes, pelo fato do processamento da requisição na camada de dados realizar acessos ao disco. Assim, enquanto os dados são recuperados do disco, a utilização de CPU reduz, chegando próximo a 18% de utilização. Quando os dados



(a) *Browsing*

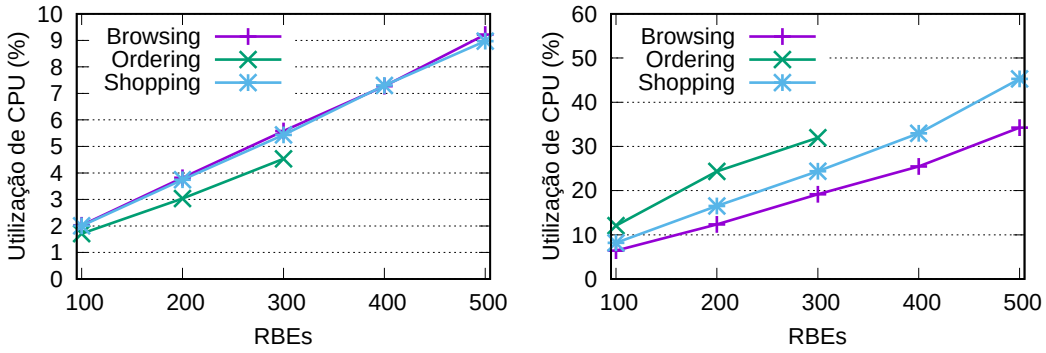


(b) *Shopping*



(c) *Ordering*

Figura 4.2: *Distribuição do número de requisições por tipo de transação nos três tipos de carga e para uma janela de 1 minuto.*



(a) *Camada de aplicação* (b) *Camada de dados*

Figura 4.3: *Média de utilização de CPU por experimento e por camada do sistema de todos cenários com janelas de 1 minuto.*

já estão disponível para processamento, a CPU é requisitada, elevando sua utilização. Finalmente, a diferença entre o uso de CPU por cada camada está relacionado com os tipos de requisições da carga Web. Esta carga tem características de requisições que demandam mais utilização de dados, ou utilização da camada de dados. Portanto, a camada de dados utiliza mais a CPU que a camada de aplicação.

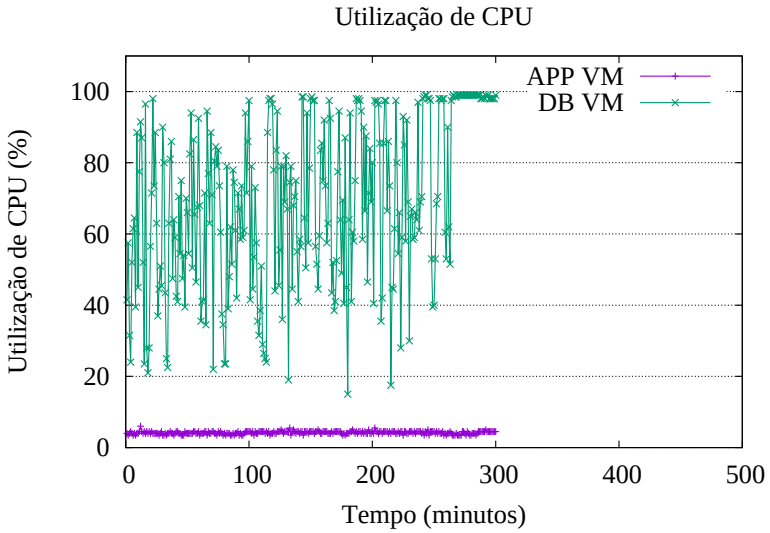


Figura 4.4: *Utilização de CPU para o cenário Ordering, 300 RBEs e janela de monitoramento de 1 minuto.*

Por fim, analisamos o consumo de energia no servidor. A Figura 4.5 ilustra a energia média consumida pelo processador do `Servidor 1` nos mesmos cenários avaliados na Figura 4.3. Podemos notar que as curvas de energia apresentam comportamento similar às curvas de CPU da camada de dados. Esse comportamento era esperado, uma vez que o consumo de CPU na camada de aplicação é pequeno. Podemos notar também que, embora as curvas de CPU (camada de dados) e energia sigam a mesma tendência, a

variação na curva de energia é menor. Isso é explicado pelo consumo de potência estática, o qual está presente independentemente da carga executada.

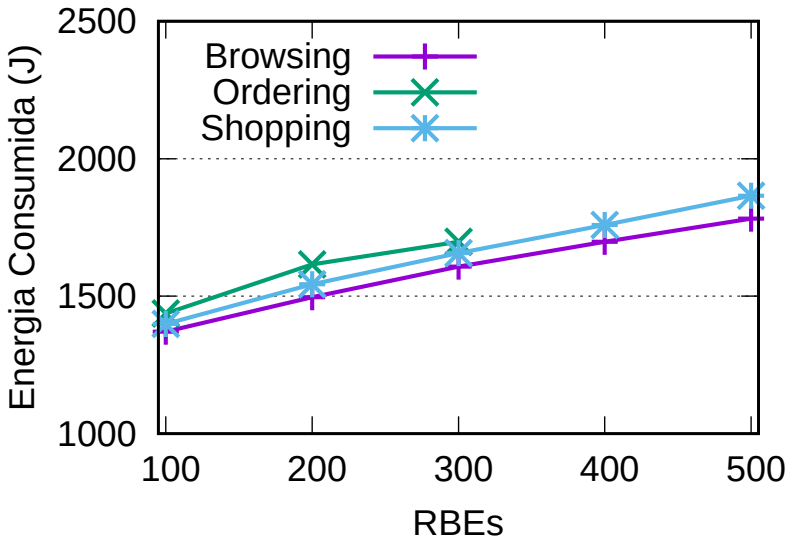


Figura 4.5: *Média do consumo de energia por RBE e para cada tipo de carga em uma janela de 1 minuto.*

Na seção seguinte, avaliamos o modelo depois de treinado, e analisamos quais os principais fatores que influenciam na acurácia do modelo.

4.3 Avaliação de acurácia

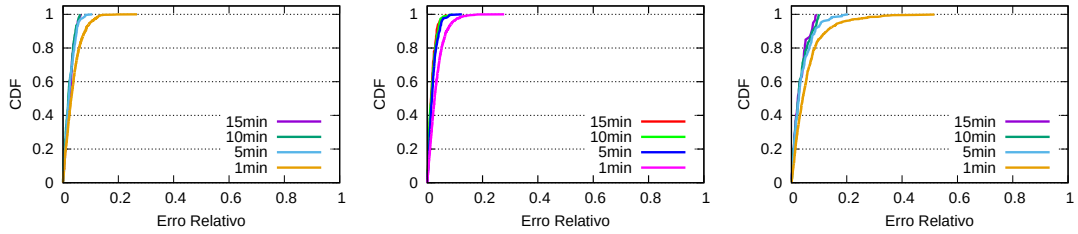
Nas subseções a seguir, os resultados dos experimentos são avaliados para diferentes tamanhos de janelas de monitoramento. Os valores das janelas bem como dos demais parâmetros usados no modelo estão ilustrados na Tabela 4.4.

Variável	T (mins)	m	n	N	m_j	m_0	P_{TDP} (Watt)	P_{static} (Watt)
Valor	1, 5, 10, 15	6	2	14	2	2	85	19,2

Tabela 4.4: *Valores dos parâmetros utilizados nos experimentos.*

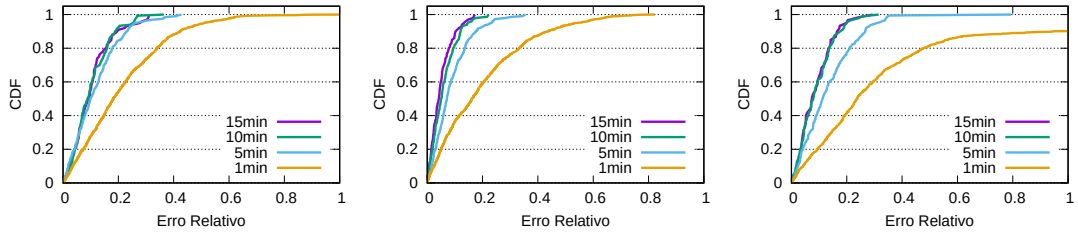
4.3.1 Resultados para a regressão da demanda de CPU

Inicialmente, avaliamos a acurácia do modelo para estimar a demanda de CPU para cada página Web em uma determinada camada, ou seja, os valores $C_{i,j}$. Para isso, utilizamos o erro relativo entre os valores $U'_{CPU,j}$ e $U_{CPU,j}$. O primeiro valor representa a utilização média prevista pelo modelo para uma camada j do sistema a partir dos valores estimados para $C_{i,j}$. O segundo valor representa a utilização média de CPU que de fato



(a) *Browsing* (b) *Shopping* (c) *Ordering*

Figura 4.6: *CDFs de* ε_{CPU} *para a camada de aplicação (APP VM).*



(a) *Browsing* (b) *Shopping* (c) *Ordering*

Figura 4.7: *CDFs de* ε_{CPU} *para a camada de dados (DB VM).*

foi observada na camada j durante os experimentos. O erro relativo é então obtido da seguinte forma:

$$\varepsilon_{CPU} = \frac{|U'_{CPU,j} - U_{CPU,j}|}{U_{CPU,j}} \tag{4-1}$$

As Figuras 4.6 e 4.7 ilustram as funções de distribuição acumulada (CDFs) de ε_{CPU} nas camadas de aplicação e de dados, respectivamente, para tamanhos diferentes da janela de monitoramento. Primeiramente, observamos que os resultados são melhores para a camada de aplicação, onde os valores médio de ε_{CPU} para os melhores cenários (janela de 15 minutos) são aproximadamente 2%, 1%, 3% para as cargas *Browsing*, *Shopping* e *Ordering*, respectivamente (Tabela 4.5). Isso ocorre porque nessa camada a utilização de CPU não tem grandes variações.

Observamos também que, em geral, janelas de monitoramento maiores resultam em erros menores, especialmente na camada de dados. Isso acontece porque quanto maior uma janela de monitoramento, maior o número de amostras usadas no cálculo da média de CPU e, consequentemente, menor a interferência que amostras de CPU com grandes variações têm sobre a média final. Para a camada de dados, os valores médios de ε_{CPU} da melhor janela (15 minutos) são aproximadamente 10%, 5%, 8% para cenários *Browsing*, *Shopping* e *Ordering*, respectivamente (Tabela 4.5).

Adicionalmente, observamos que os resultados aqui apresentados para a regressão de $C_{i,j}$ estão de acordo com os apresentados por Zhang et al. [62].

Camada	Janela	Shopping	Browsing	Ordering
Aplicação	1 minuto	0.033	0.0367	0.058
	5 minuto	0.020	0.024	0.037
	10 minuto	0.018	0.022	0.032
	15 minuto	0.017	0.024	0.030
Dados	1 minuto	0.198	0.213	0.356
	5 minuto	0.087	0.116	0.133
	10 minuto	0.060	0.101	0.089
	15 minuto	0.050	0.103	0.088

Tabela 4.5: *Valores médios de* ε_{CPU}.

4.3.2 Resultados para a regressão da demanda de energia

Em seguida, avaliamos a acurácia do modelo para estimar a demanda de energia para cada tipo de transação em uma determinada camada, ou seja, os valores $T_{i,j}$. Primeiramente, utilizamos os dados coletados com o RAPL como medida real do consumo de energia do servidor físico, e posteriormente calculamos a acurácia do modelo. Este tipo de medida real é conhecida como caixa preta, pois é uma medida global ao invés de local. Nosso modelo é uma medida local, pois estima o consumo energético a nível de transação, enquanto o RAPL estima o consumo de todo o processador. Portanto, utilizamos o RAPL somente para validação do modelo. Todavia, o RAPL não é capaz de reportar o consumo de energia por núcleos do processador para conseguirmos separar o consumo energético por cada camada Web. Diante desta limitação, nossa avaliação leva em consideração a energia consumida por todo o processador. O erro relativo é então calculado considerando-se E e E'. O primeiro valor representa a energia consumida pelo servidor em uma janela de monitoramento, sendo esse valor reportado pelo RAPL. O segundo valor representa a energia estimada para o servidor a partir dos valores estimados para E_j usando nosso modelo, bem como a energia consumida pelos núcleos que não estão sendo usados por nenhuma camada do sistema (E_0). Formalmente temos:

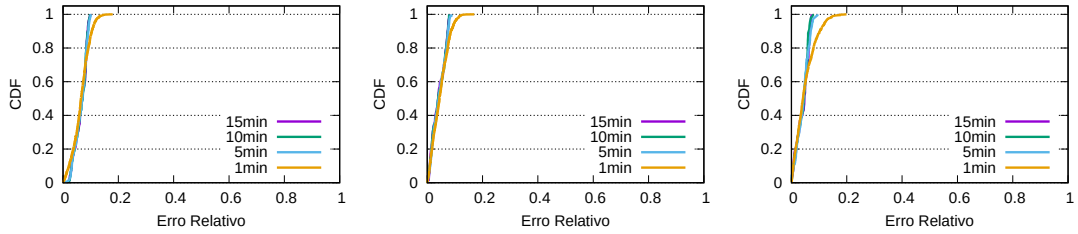
$$E' = E_0 + \sum_{j=1}^{n} E_j. \tag{4-2}$$

Assumindo que existem m_0 núcleos do servidor que não estão sendo usados por nenhuma camada no sistema, E_0 é dado por:

$$E_0 = T(P_{static,0} + \frac{\sum_{k_0=1}^{m_0} U_{core_{k_0}} P_{active,0}}{100 m_0}), \tag{4-3}$$

onde:

$$P_{static,0} = m_0 \frac{P_{static}}{m}, \qquad P_{active,0} = m_0 \frac{P_{active}}{m} \quad \text{e} \quad \sum_{k_0=1}^{m_0} U_{core_{k_0}} = U_{CPU,0}.$$



(a) *Browsing* (b) *Shopping* (c) *Ordering*

Figura 4.8: *CDFs de* ε_{energy} *para todos cenários.*

O erro relativo para a regressão de energia é então obtido da seguinte forma:

$$\varepsilon_{energy} = \frac{|E' - E|}{E} \tag{4-4}$$

A Figura 4.8 ilustra as CDFs de ε_{energy} para todos cenários de experimentação. A Tabela 4.6 mostra o valor médio de ε_{energy} em cada cenário ilustrado na Figura 4.8.

Janela	Shopping	Ordering	Browsing
1	0,046	0,049	0,065
5	0,042	0,042	0,065
10	0,041	0,040	0,065
15	0,041	0,041	0,065

Tabela 4.6: *Valor médio de* ε_{energy} *em cada cenário avaliado.*

Assim como na regressão de $C_{i,j}$, observamos que janelas de monitoramento de 1 minuto produzem resultados menos precisos que as demais janelas. Notamos também que a acurácia do modelo de energia é, em geral, melhor que a do modelo de estimação de CPU, especialmente quando comparada com o desempenho desse último na camada de dados. Isso ocorre por dois fatores: i) A validação do modelo de energia está levando em consideração todo o servidor e, consequentemente, o desempenho menos acurado na camada de dados está sendo amortizado pelo bom desempenho na camada de aplicação; ii) A pequena variação observada para os valores de energia à medida que o número de RBEs concorrentes aumenta. Como explicado anteriormente, essa pequena variação ocorre devido à componente estática do consumo de potência, a qual está presente independentemente da carga de trabalho utilizada.

Finalmente, observamos que nosso modelo, apesar de utilizar poucos parâmetro e informações facilmente obtidas em ambientes de produção, produz erros médios menores que 7% em qualquer cenário de utilização, o que de acordo com [48] são resultados com alta acurácia para modelos de energia. Alem disso, esses erros estão na mesma ordem de grandeza dos apresentados em Piga et al. [45].

Além de prover estimativas mais acuradas usando poucos parâmetros de entrada, um modelo construído no nível de transações permite compor a demanda energética de

cargas de trabalho que possuem diferentes combinações desses tipos de transações. Essa capacidade, por sua vez, possibilita estimar o consumo de energia de sistemas Web mesmo quando a carga de trabalho muda ao longo do tempo.

Para avaliarmos essa habilidade do modelo, criamos um conjunto de dados formado pela união dos três conjuntos de dados anteriores (*Browsing*, *Shopping* e *Ordering*) para cada janela de monitoramento. Denominamos esse novo conjunto de dados como o cenário *Mix*. Dessa forma, os dados coletados nos cenários *Browsing*, *Shopping* e *Ordering* com janela de monitoramento de 1 minuto formam do cenário *Mix* para esse tamanho de janela. A mesma lógica é aplicada para criar o conjunto de dados para o cenário *Mix* com outras janelas de monitoramento.

A Figura 4.9 mostra o resultado da regressão de $T_{i,j}$ quando o modelo é treinado com dados do cenário *Mix* e testado com dados do *Shopping*. Podemos observar que, com exceção da curva de 1 minuto, em todas as demais curvas o erro relativo (ε_{energy}) é menor ou igual a 10% em 100% das amostras. Nesse experimento, o valor médio de ε_{energy} foi de aproximadamente 5% para todas as curvas. Esses resultados demonstram que um modelo, baseado em tipos de transações, estima a demanda energética de cargas de trabalho, formadas por diferentes combinações dos mesmos tipos de transações, com elevada acurácia.

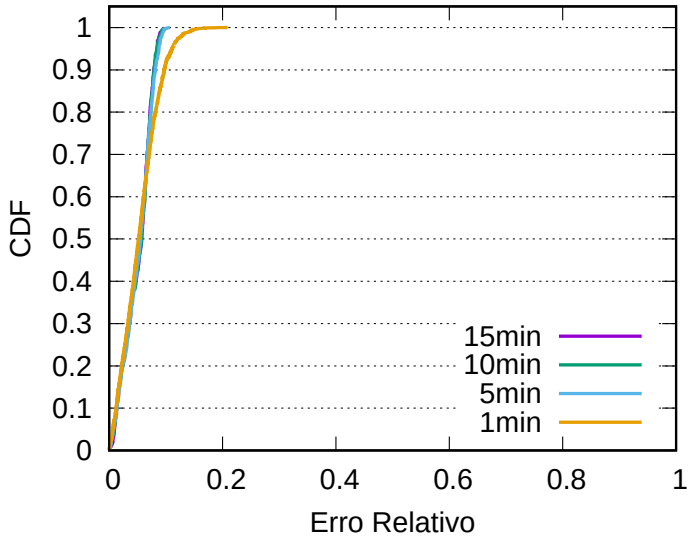


Figura 4.9: *CDF de ε_{energy} para um modelo treinado com a carga Mix e testado com a carga Shopping.*

4.4 Considerações finais

Este capítulo apresentou uma avaliação experimental do modelo analítico proposto para estimar o consumo de energia de sistemas Web no nível de transações. Es-

pecificamente, apresentamos o ambiente de experimentação implantado, a metodologia utilizada para realizar as comparações com um modelo de energia implementado em hardware, tomado como referência neste trabalho, e os resultados obtidos dessas comparações. No capítulo seguinte, apresentaremos um resumo das nossas conclusões e diretrizes para trabalhos futuros.

CAPÍTULO 5

Conclusão e Trabalhos Futuros

Conforme foi comentado anteriormente, eficiência energética se tornou uma questão importante em ambientes de computação tradicionalmente orientados a desempenho. Além disso, mensurações confiáveis são essenciais para melhorar a eficiência energética desses ambientes. Uma das abordagens promissoras, nesse contexto, consiste em modelos de estimação implementados em software. Embora existam várias propostas na literatura voltadas para ambientes de PAD, poucos trabalhos se dedicam à construção de modelos que estimam o consumo de energia para aplicações Web.

Neste trabalho, propomos e avaliamos um modelo com esse objetivo. A principal característica desse modelo é o fato de observar a demanda de serviço de componentes mais finos do sistema, como a utilização de CPU proveniente do processamento de apenas uma transação Web para estimar a energia consumida para toda a aplicação. Essa característica nos permitiu a construção de um modelo simples, devido a utilizar apenas uma métrica de desempenho: a utilização de CPU. Além disso, nosso modelo emprega apenas um tipo de técnica de análise de dados: a regressão linear múltipla.

As avaliações do modelo mostraram que ele é acurado, produzindo erros na faixa entre 6% a 8%, um limiar bem aceito pela literatura. Esses erros estão na mesma ordem de grandeza de outros trabalhos que propõem modelos mais complexos, como apresentado no Capítulo 2. A complexidade está relacionada com a quantidade de parâmetros e técnicas utilizadas pelos modelos. Quanto mais parâmetros, mais sobrecarga do sistema de monitoramento e quanto mais técnicas, maior é o custo computacional do modelo.

O desenvolvimento do modelo foi marcado por várias dificuldades superadas. Primeiramente, algumas no ambiente de experimentação, com o TPC-W e o RAPL. Em seguida, com execuções dos experimentos e finalmente, com a formulação do modelo. Destacamos a principais dificuldades a seguir:

- TPC-W: A primeira versão que encontramos não funcionou, depois de vários experimentos, observamos erros nos logs do TomCat. Depois, tivemos erros com a quantidade de itens do banco de dados, não conseguíamos executar mais de 300 RBEs, isso, depois de 5h de experimentos. Portanto, esta fase de testar o ambiente do TPC-W necessitou de muito tempo.

- RAPL: Primeiramente, não existe uma documentação simples, com referência da API, somente o manual do processador como um todo. Além disso, tivemos que testar várias APIs que retornassem a energia consumida pelo processador, somente depois de vários testes, conseguimos definir a API do PAPI. Após essa etapa, tivemos problema com o tempo de janela que queríamos, 1 minuto estava fazendo o RAPL sobrescrever os contadores e estávamos perdendo informações, dessa forma, tivemos que reduzir a janela para 15 segundos, o que gerou a necessidade de agregarmos dados de coleta do RAPL para obtermos a janela de 1 minuto que era igual a dos demais dados coletados.
- Experimentos: Primeiramente encontramos uma dificuldade de sincronizar as execuções, pois o Colectd era um serviço separado, o script do RAPL era outro e o script para leitura do log do tomCat era outro. Tivemos que testar formas de agendamento de execução dos scripts através de ferramentas internas do Ubuntu, além disso, tínhamos que levar em conta que o agendamento tinha que ser feito em 4 máquinas diferentes, um agendamento para o Dom0, um para APP VM, outro para DB VM e outro para Emulator VM. Além disso, tínhamos que repetir o processo para todos cenários de experimentação. Outra dificuldade foi o tempo, como os experimentos eram de cinco horas e quarenta minutos, caso alguma coisa desse errado, tínhamos que refazer tudo novamente, o que ocorreu várias vezes, pois caso uma camada falhasse, já tinha que refazer. Portanto, a etapa dos experimentos foi a que mais necessitou de tempo.
- Formulação: Não foi muito trabalhoso, mas tivemos que fazer um ajuste da primeira versão do modelo. Primeiramente, não estávamos separando o P_{static} somente para a camada, estávamos somando ele dentro do somatório do $\eta_{i,j}$. Outra parte que tivemos dificuldade, foi a quantidade de dados que deveriam ser lidos e iriam ser gerado. A organização destes dados, no início, não estava muito fácil de trabalhar, devido a quantidade de cenários, a tarefa de treinamento e teste do modelo era complexa, pois um diretório errado e podia dar erro na avaliação, sendo que cada cenário tinha pelo menos três diretórios e cada diretório, três diretórios com dados a serem lidos. Contudo, através de ajustes nos scripts do R para automatizar o processo, a tarefa ficou mais simples.

Este trabalho apresentou um modelo acurado, contudo, abre oportunidades para extensão e aperfeiçoamento do modelo proposto, sendo que as mais relevantes seguem a seguir:

- Extensão do modelo:
  - Adição de componentes: utilizar outras fontes de energia como: a memória RAM, disco e outros, para compor a energia total do servidor, contudo,

utilizando o mesmo componente fino.
  - Executar em tempo de execução: desenvolver componentes para alimentar o modelo em tempo de execução das aplicações Web.
  - Avaliar outras técnicas de estatísticas: treinar o modelo com outras técnicas além da regressão linear múltipla e avaliar se demonstram melhora na acurácia do modelo.
  - Criar gerenciador de recursos: adaptar o uso do modelo para alimentar um gerenciador de recursos computacionais com informações necessárias para tomada de decisão estratégica.

- Metodologia de avaliação:
  - Avaliar em arquiteturas diferentes: mesmo o modelo sendo compatível com a maioria das arquiteturas, avaliar o modelo em outras arquiteturas é uma oportunidade em aberto.
  - Avaliar perda com virtualização: não focamos na avaliação da virtualização, mas em avaliar a acurácia do modelo. Portanto, poderia ser realizado uma verificação do modelo em um ambiente não virtualizado e observado seus comportamentos em trabalhos futuros.

Referências Bibliográficas

[1] AGENCY, U. E. P. **Epa report on server and data center energy efficiency**. https://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Report_Exec_Summary_Final.pdf, 2007. Último acesso: fev-02-2017.

[2] AMD. **Amd opterontm 6200 series processors linux tuning guide.** http://amd-dev.wpengine.netdna-cdn.com/wordpress/media/2012/10/51803A_OpteronLinuxTuningGuide_SCREEN.pdf. Último acesso: aug-09-2016.

[3] ARMAN, S.; SMITH, S. J.; SARTOR, D. A.; BROWN, R. E.; HERRLIN, M.; KOOMEY, J. G.; MASANET, E. R.; HORNER, N.; AZEVEDO, I. L.; LINTNER, W. **United states data center energy usage report.** https://eta.lbl.gov/sites/all/files/publications/lbnl-1005775_v2.pdf, 2016. Último acesso: fev-02-2017.

[4] BARROSO, L.; HOLZLE, U. **The case for energy-proportional computing**. *Computer*, 40(12):33–37, 2007.

[5] BARROSO, L. A.; HOELZLE, U. **The Datacenter As a Computer: An Introduction to the Design of Warehouse-Scale Machines**. Morgan and Claypool Publishers, 1st edition, 2009.

[6] BEDARD, D.; LIM, M. Y.; FOWLER, R.; PORTERFIELD, A. **Powermon: Fine-grained and integrated power monitoring for commodity computer systems**. In: *Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon)*, p. 479–484, March 2010.

[7] BELOGLAZOV, A. **Energy-efficient management of virtual machines in data centers for cloud computing**. Phd thesis, Department of Computing and Information Systems, The University of Melbourne, Australia, 2 2013.

[8] BELOGLAZOV, A.; ABAWAJY, J.; BUYYA, R. **Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing**. *Future Gener. Comput. Syst.*, 28(5):755–768, 2012.

[9] BERTRAN, R.; GONZALEZ, M.; MARTORELL, X.; NAVARRO, N.; AYGUADE, E. **Decomposable and responsive power models for multicore processors using performance counters**. In: *Proceedings of the 24th ACM International Conference on Supercomputing*, p. 147–158, 2010.

[10] BOHRER, P.; ELNOZAHY, E. N.; KELLER, T.; KISTLER, M.; LEFURGY, C.; MCDOWELL, C.; RAJAMONY, R. **The case for power management in web servers**. In: *Power aware computing*, p. 261–289. Springer, 2002.

[11] BROWN, D. J.; REAMS, C. **Toward energy-efficient computing**. *Commun. ACM*, 53(3):50–58, 2010.

[12] BURROWS, P. **Apple says data centers now use 100% renewable energy**. https://www.bloomberg.com/news/articles/2013-03-21/apple-says-data-centers-now-use-100-renewable-energy, 2013. Último acesso: fev-02-2017.

[13] CHEN, F.; GRUNDY, J.; SCHNEIDER, J.-G.; YANG, Y.; HE, Q. **Automated analysis of performance and energy consumption for cloud applications**. In: *Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering*, p. 39–50, 2014.

[14] CHEN, M.; ZHANG, H.; SU, Y.-Y.; WANG, X.; JIANG, G.; YOSHIHIRA, K. **Effective vm sizing in virtualized data centers**. In: *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, p. 594–601, 2011.

[15] CHEN, Y.; DAS, A.; QIN, W.; SIVASUBRAMANIAM, A.; WANG, Q.; GAUTAM, N. **Managing server energy and operational costs in hosting centers**. In: *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, p. 303–314, 2005.

[16] CIOARA, T.; ANGHEL, I.; SALOMIE, I.; COPIL, G.; MOLDOVAN, D.; GRINDEAN, M. **Time series based dynamic frequency scaling solution for optimizing the cpu energy consumption**. In: *Intelligent Computer Communication and Processing (ICCP), 2011 IEEE International Conference on*, p. 477–483, 2011.

[17] **Collectd – the system statistics collection daemon**. https://collectd.org., 2016. Último acesso: oct-14-2015.

[18] DAVID, H.; GORBATOV, E.; HANEBUTTE, U. R.; KHANNA, R.; LE, C. **Rapl: Memory power estimation and capping**. In: *Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE International Symposium on*, p. 189–194, 2010.

[19] ECONOMOU, D.; RIVOIRE, S.; KOZYRAKIS, C.; RANGANATHAN, P. **Full-system power analysis and modeling for server environments.** In: *Workshop on Modeling, Benchmarking, and Simulation (MoBS) (June 2006).*, 2006.

[20] FAN, X.; WEBER, W.-D.; BARROSO, L. A. **Power provisioning for a warehouse-sized computer**. In: *Proceedings of the 34th Annual International Symposium on Computer Architecture*, p. 13–23, 2007.

[21] FLINN, J.; SATYANARAYANAN, M. **Managing battery lifetime with energy-aware adaptation**. *ACM Trans. Comput. Syst.*, 22(2):137–179, 2004.

[22] GANDHI, A. **Dynamic Server Provisioning for Data Center Power Management**. PhD thesis, Carnegie Mellon University, 2013.

[23] GONG, Z.; GU, X.; WILKES, J. **Press: Predictive elastic resource scaling for cloud systems**. In: *Network and Service Management (CNSM), 2010 International Conference on*, p. 9–16, 2010.

[24] GYU KIM, S.; CHOI, C.; EOM, H.; YEOM, H.; BYUN, H. **Energy-centric dvfs controling method for multi-core platforms**. In: *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*, p. 685–690, 2012.

[25] HAUGEN, D. **How wind energy helped Iowa attract Facebook's new data center**. http://midwestenergynews.com/2013/04/24/how-wind-energy-helped-iowa-attract-facebooks-new-data-center/, 2013. Último acesso: fev-02-2017.

[26] INTEL. **Intel 64 and ia-32 software developer manuals.** http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html. Último acesso: aug-09-2016.

[27] ISCI, C.; CONTRERAS, G.; MARTONOSI, M. **Live, runtime phase monitoring and prediction on real systems with application to dynamic power management**. In: *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, p. 359–370, 2006.

[28] ISCI, C.; MARTONOSI, M. **Runtime power monitoring in high-end processors: Methodology and empirical data**. In: *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, p. 93–, 2003.

[29] JAIN, R. **The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling**. John Wiley & Sons, 1990.

[30] JAMES, G.; WITTEN, D.; HASTIE, T.; TIBSHIRANI, R. **An introduction to statistical learning**, volume 6. Springer, 2013.

[31] JOSEPH, R.; MARTONOSI, M. **Run-time power estimation in high performance microprocessors**. In: *Proceedings of the 2001 International Symposium on Low Power Electronics and Design*, p. 135–140, 2001.

[32] KIM, K. H.; BELOGLAZOV, A.; BUYYA, R. **Power-aware provisioning of virtual machines for real-time cloud services**. *Concurr. Comput. : Pract. Exper.*, 23(13):1491–1505, 2011.

[33] LAROS, J. H.; POKORNY, P.; DEBONIS, D. **Powerinsight - a commodity power measurement capability**. In: *Green Computing Conference (IGCC), 2013 International*, p. 1–6, June 2013.

[34] LIN, H.; QI, X.; YANG, S.; MIDKIFF, S. **Workload-driven vm consolidation in cloud data centers**. In: *Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International*, p. 207–216, 2015.

[35] LIU, Y.; ALGHAMDI, M.; KU, W.-S.; ZHOU, Y.; TANEJA, S.; QIN, X. **Profiling energy usage of web-service applications on clusters**. In: *Networking, Architecture and Storage (NAS), 2016 IEEE International Conference on*, p. 1–5. IEEE, 2016.

[36] MENASCE, D. A.; DOWDY, L. W.; ALMEIDA, V. A. F. **Performance by Design: Computer Capacity Planning By Example**. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[37] METRI, G.; SRINIVASARAGHAVAN, S.; SHI, W.; BROCKMEYER, M. **Experimental analysis of application specific energy efficiency of data centers with heterogeneous servers**. In: *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, p. 786–793, 2012.

[38] MI, N.; CASALE, G.; CHERKASOVA, L.; SMIRNI, E. **Sizing multi-tier systems with temporal dependence: benchmarks and analytic models**. *Journal of Internet Services and Applications*, 1(2):117–134, 2010.

[39] MIFTAKHUTDINOV, R.; EBRAHIMI, E.; PATT, Y. N. **Predicting performance impact of dvfs for realistic memory systems**. In: *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, p. 155–165, 2012.

[40] MURUGESAN, S.; GANGADHARAN, G. R. **Green IT: An Overview**, p. 1–21. John Wiley & Sons, Ltd, 2012.

[41] NEUGEBAUER, R.; MCAULEY, D. **Energy is just another resource: Energy accounting and energy pricing in the nemesis os**. In: *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*, p. 67–, 2001.

[42] NVIDIA. **Nvidia management library (nvml).** https://developer.nvidia.com/nvidia-management-library-nvml. Último acesso: aug-09-2016.

[43] ON CLIMATE CHANGE, I. I. P. **Contribution of working group iii to the fifth assessment report of the intergovernmental panel on climate change.** https://www.ipcc.ch/report/ar5/wg3/, 2016. Último acesso: jul-25-2016.

[44] ORGERIE, A.-C.; ASSUNCAO, M. D. D.; LEFEVRE, L. **A survey on techniques for improving the energy efficiency of large-scale distributed systems**. *ACM Comput. Surv.*, 46(4):47:1–47:31, 2014.

[45] PIGA, L.; BERGAMASCHI, R. A.; RIGO, S. **Empirical and analytical approaches for web server power modeling**. *Cluster Computing*, 17(4):1279–1293, 2014.

[46] RANGANATHAN, P. **Recipe for efficiency: Principles of power-aware computing**. *Commun. ACM*, 53(4):60–67, 2010.

[47] RANGANATHAN, P.; LEECH, P.; IRWIN, D.; CHASE, J. **Ensemble-level power management for dense blade servers**. *SIGARCH Comput. Archit. News*, 34(2):66–77, 2006.

[48] RIVOIRE, S.; RANGANATHAN, P.; KOZYRAKIS, C. **A comparison of high-level full-system power models.** *HotPower*, 8:3–3, 2008.

[49] ROTEM, E.; NAVEH, A.; ANANTHAKRISHNAN, A.; WEISSMANN, E.; RAJWAN, D. **Power-management architecture of the intel microarchitecture code-named sandy bridge**. *IEEE Micro*, 32(2):20–27, 2012.

[50] SCARAMELLA, J.; MARDEN, M.; DALY, J.; PERRY, R. **The cost of retaining aging it infrastructure**. Technical report, International Data Corporation (IDC), Framingham, MA, feb 2014.

[51] SNOWDON, D. C.; LE SUEUR, E.; PETTERS, S. M.; HEISER, G. **Koala: A platform for os-level power management**. In: *Proceedings of the 4th ACM European Conference on Computer Systems*, p. 289–302, 2009.

[52] SU, B.; GU, J.; SHEN, L.; HUANG, W.; GREATHOUSE, J. L.; WANG, Z. **Ppep: Online performance, power, and energy prediction framework and dvfs space exploration**. In: *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-47, p. 445–457, 2014.

[53] SUBRAMANIAM, B.; FENG, W. C. **Enabling efficient power provisioning for enterprise applications**. In: *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*, p. 71–80, 2014.

[54] SUBRAMANIAM, B.; FENG, W.-C. **Towards energy-proportional computing for enterprise-class server workloads**. In: *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, p. 15–26, 2013.

[55] **Tpc-w benchmark.** http://www.tpc.org., 2016. Último acesso: oct-14-2015.

[56] VARGHESE, B.; CARLSSON, N.; JOURJON, G.; MAHANTI, A.; SHENOY, P. **Greening web servers: A case for ultra low-power web servers**. In: *Green Computing Conference (IGCC), 2014 International*, p. 1–8. IEEE, 2014.

[57] VENKATACHALAM, V.; FRANZ, M. **Power reduction techniques for microprocessor systems**. *ACM Comput. Surv.*, 37(3):195–237, 2005.

[58] VISWANATHAN, B.; VERMA, A.; DUTTA, S. **Cloudmap: Workload-aware placement in private heterogeneous clouds**. In: *Network Operations and Management Symposium (NOMS), 2012 IEEE*, p. 9–16, 2012.

[59] WANG, Q.; KANEMASA, Y.; LI, J.; LAI, C. A.; MATSUBARA, M.; PU, C. **Impact of dvfs on n-tier application performance**. In: *Proceedings of the First ACM SIGOPS Conference on Timely Results in Operating Systems*, p. 5:1–5:16, 2013.

[60] WEHNER, M.; OLIKER, L.; SHALF, J. **A real cloud computer**. *IEEE Spectr.*, 46(10):24–29, 2009.

[61] ZENG, H.; ELLIS, C. S.; LEBECK, A. R. **Experiences in managing energy with ecosystem**. *IEEE Pervasive Computing*, 4(1):62–68, 2005.

[62] ZHANG, Q.; CHERKASOVA, L.; SMIRNI, E. **A regression-based analytic model for dynamic resource provisioning of multi-tier applications**. In: *Proceedings of the Fourth International Conference on Autonomic Computing*, p. 27–, 2007.