Um Modelo Analítico para Estimar o Consumo de Energia de Aplicações Web no Nível de Transações

Alex R. Ferreira[1], **Denner S. L. Vidal**[2], **Valéria Q. dos Reis**[2], **Sand Luz Correa**[1]

[1]Instituto de Informática – Universidade Federal de Goiás

[2]Faculdade de Computação – Universidade Federal de Mato Grosso do Sul

alexrabeloferreira@inf.ufg.br, denner.vidal@gmail.com

valeria@facom.ufms.br, sand@inf.ufg.br

Abstract. *Modern enterprise servers provide several features to manage the power consumption of their subsystems. In environments with high computational demand, these features can significantly reduce energy consumption. However, in order to make proper use of such resources, we need to understand how the applications that run in these servers consume the computational resources and the implications of such usage for the overall power and energy consumption. In this work, we propose an energy model to characterize the energy consumption of enterprise servers that host Web systems. Unlike other works in the literature, our model captures the consumption pattern of these applications at the level of transactions and for each tier of the system. In addition, our model depends solely on the CPU utilization and on server architectural parameters, which can be easily obtained in current production environments. We demonstrate the effectiveness of our model in a real experimental environment, based on the TPC-W benchmark. Results show that our model is able to estimate the energy consumption of Web applications with errors in the same order of magnitude as those presented by previous work.*

Resumo. *Servidores modernos oferecem diversos recursos para gerenciar o consumo de potência de seus subsistemas. Em ambientes com grande demanda computacional, esses recursos podem reduzir o consumo de energia de maneira significativa. No entanto, para fazer bom uso de tais recursos, é necessário entender como as aplicações que executam nesses servidores utilizam os recursos computacionais e as implicações desse uso para o consumo de potência e de energia. Neste trabalho, propomos um modelo analítico para estimar o consumo de energia de servidores que hospedam sistemas Web. 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. Resultados mostram que nosso modelo é capaz de estimar o consumo de energia de sistemas Web com erros na mesma ordem de grandeza que modelos mais complexos existentes na literatura.*

1. Introdução

O custo para alimentar um servidor empresarial típico durante seu tempo de vida já ultrapassa o custo de aquisição [Scaramella et al. 2014]. Como consequência, nos últimos anos, ambientes reconhecidamente orientados a desempenho, como *data centers* para computação em nuvem e *clusters* para processamento de alto desempenho (*High Performance Computing - HPC*), passaram a reconhecer a eficiência energética como outro requisito importante no projeto de sistemas computacionais de grande escala [Barroso and Hölzle 2007].

Servidores modernos oferecem diversos recursos para gerenciar o consumo de potência de seus subsistemas. Para ilustrar, a partir da tecnologia Sandy Bridge [Rotem et al. 2012], os processadores Intel proveem uma interface onde é possível limitar o consumo de potência em um determinado limiar, durante um período de tempo. Muitos processadores permitem também ajustar a voltagem e a frequência de sua operação de acordo com a carga de trabalho ou nível de utilização [Hsu and Feng 2005, Rountree et al. 2009], enquanto outros *chips* podem desativar ou colocar em estágio de dormência um subcomponente que não está em uso [Kaxiras and Martonosi 2008].

Em ambientes com grande demanda computacional, a aplicação dessas técnicas juntamente com controles no nível do sistema operacional, como por exemplo alocação de *threads* e políticas de escalonamento, podem reduzir o consumo de energia de maneira significativa [Isci et al. 2006a, Fan et al. 2007, Cochran et al. 2011]. No entanto, para fazer bom uso de tais técnicas, é necessário entender como as aplicações que executam nesses servidores utilizam os recursos computacionais disponíveis e as implicações desse uso para o consumo de potência e, por conseguinte, de energia. Além disso, como mostrado em [Isci et al. 2006b], quanto mais fino for o entendimento e o controle sobre os componentes controlados, maior a economia de energia. Esse controle fino se torna ainda mais importante em ambientes como *data centers* baseados em serviços de Internet, onde fenômenos de rajada e atrasos entre as diferentes camadas que compõem os serviços podem alterar o padrão de utilização dos recursos de maneira considerável e em um curto espaço de tempo [Mi et al. 2010]. A despeito 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 obter maior acurácia, muitos modelos existentes na literatura, como em [Piga et al. 2014], procuram monitorar um grande número de métricas e contadores de desempenho, para então realizar algum mecanismo de poda que retorne as métricas mais relacionadas com o componente observado. Essa abordagem, no entanto, torna os modelos de energia mais complexos, além de gerar sobrecarga relacionada à coleta e armazenamento 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. 2007], 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 pro-

cessar cada página individualmente. Avaliamos a efetividade do nosso modelo utilizando um ambiente de experimentação real e o *benchmark* TPC-W [TPC 2016]. 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.

O restante deste trabalho está organizado da seguinte forma. A Seção 2 introduz uma breve revisão de trabalhos relacionados. A Seção 3 apresenta o modelo de sistema considerado e descreve um modelo da literatura para estimar demandas de CPU em sistemas Web. A Seção 4 descreve o modelo de estimação de consumo de energia proposto. O ambiente de experimentação bem como o *benchmark* usado na avaliação são apresentados na Seção 5 juntamente com os resultados obtidos. Finalmente, a Seção 6 conclui o trabalho e apresenta as perspectivas de trabalhos futuros.

2. Fundamentos e Trabalhos Relacionados

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

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

onde E denota a energia, P denota a potência e T representa um intervalo de tempo específico. Em circuitos CMOS, o consumo de potência é proveniente de duas fontes: uma estática (P_{static}) e outra dinâmica($P_{dynamic}$) [Kaxiras and Martonosi 2008]. Logo, podemos reescrever a Equação 1 como:

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

O consumo estático é causado por "vazamentos" de correntes presentes em qualquer circuito ativo. Esse tipo de consumo não depende do cenário de utilização e está presente mesmo quando o componente está ocioso. O consumo dinâmico, por outro lado, é proveniente de atividades nos circuitos elétricos e está relacionado com o cenário de utilização, frequências do relógio e atividades de entrada e saída.

Em processadores, o problema de estimação de potência é definido como a tarefa de mensurar ou aferir a potência consumida por esses componentes quando um programa ou carga de trabalho são executados. Como a maioria dos processos de medição real para esse problema são intrusivos [Bedard et al. 2010, Ge et al. 2010], muitos trabalhos utilizam modelos de estimação em software. Nesse contexto, uma formulação comumente empregada consiste em estimar P_{static} através de parâmetros arquiteturais, enquanto $P_{dynamic}$ é estimada como uma função linear de n registradores específicos, denominados contadores de desempenho, que armazenam estatísticas sobre a atividade de diferentes subsistemas do processador [Joseph and Martonosi 2001, Isci and Martonosi 2003]. A potência total consumida é então calculada como a soma desses dois valores. No entanto, a maioria dos trabalhos que se baseiam em contadores de desempenho para estimar o consumo de potência/energia de processadores são voltados para ambientes de HPC, cujas cargas de trabalho diferem significativamente daquelas encontradas em servidores Web [Wang et al. 2013]. Além disso, a maioria desses modelos requerem mais de 15 contadores de desempenho, alguns dos quais não estão disponíveis na maioria dos servidores

comerciais típicos. Nosso modelo, ao contrário, baseia-se apenas na utilização de CPU e em parâmetros arquiteturais do servidor.

A proposta de [Bohrer et al. 2002] foi a primeira a tratar do problema de estimação de consumo de potência/energia em servidores e sistemas Web. Os autores apresentam uma ferramenta de simulação que estima o tempo de CPU e a energia consumida por uma requisição Web com base nos ciclos de CPU gastos para executar a requisição. Entretanto, o modelo de estimação leva em consideração apenas o servidor de aplicação e os autores assumem que as requisições ou são completamente servidas pela memória ou são completamente atendidas pelo disco. Nosso modelo, ao contrário, leva em consideração todos os objetos recuperados nas diferentes camadas (lógica e de dados) para atender as requisições. Consequentemente, nosso modelo é capaz de caracterizar a demanda de CPU e o consumo de energia requerido por cada página Web em cada camada.

Uma proposta para estimar o consumo de potência/energia de sistemas Web em diferentes estados de frequência/voltagem (*P-state*) de um processador é apresentado em [Piga et al. 2014]. Para derivar a relação entre consumo de potência e *P-state*, os autores monitoram a atividade do sistema através de um grande número de contadores de desempenho e métricas do sistema operacional. Entretanto, como o monitoramento dessas métricas é feito como uma caixa preta, ou seja, sem associá-las às requisições que estão sendo atendidas, para alcançar maior acurácia, os autores utilizam diferentes métodos de análise de dados, incluindo regressão linear, análise de *cluster* e técnicas de inteligência artificial. Dessa forma, os modelos apresentados atingem erros médios próximos a 2%. Como mostraremos na Seção 5.2, nosso modelo, apesar de usar apenas a técnica de regressão linear e um número menor de métricas, é capaz de atingir erros médios na mesma ordem de grandeza.

Existem também trabalhos que investigam os efeitos das técnicas de gerenciamento de energia no desempenho de servidores Web [Wang et al. 2013, Metri et al. 2012]. Esses trabalhos, entretanto, não proveem nenhuma formulação para o problema de estimação de consumo de potência/energia. Finalmente, existe uma vasta literatura em simulação que busca melhorar a eficiência energética em *data centers* de computação em nuvem pela consolidação de máquinas virtuais (VMs)(um *survey* abrangente pode ser encontrado em [Beloglazov and Buyya 2012]). Nosso modelo pode ser usado nesses simuladores para guiar a decisão dos algoritmos de otimização.

3. Demanda de CPU Requerida por uma Página Web

Tipicamente, sistemas Web são concebidos em múltiplas camadas. Cada camada, por sua vez, fornece um certa funcionalidade, incluindo apresentação de dados, lógica da aplicação e gerenciamento de dados. Clientes interagem com esses sistemas através de navegadores, os quais recuperam páginas Web. Uma página Web consiste em um arquivo HTML e vários objetos embutidos (imagens, áudio, vídeo, etc.). No lado do servidor, 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. Referimo-nos 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.

Zhang et al. [Zhang et al. 2007] 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 página Web para então compor a demanda de CPU de cargas de trabalho contendo diferentes combinações de páginas. Assumindo um sistema Web de n camadas e composto por η tipos de páginas Web diferentes, os autores aplicam a Lei da Utilização [Menasce et al. 2004], obtendo:

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

onde T é o tamanho da janela de monitoramento; η_i, $1 \leq i \leq \eta$, é o número de páginas Web 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 página Web do tipo i na camada j. Entretanto, como é difícil obter tempos de serviço de forma acurada, os autores propõem uma aproximação e substituem $D_{i,j}$ pelo custo de CPU (em termos de utilização) de $D_{i,j}$, denotado por $C_{i,j}$. Assim, para cada camada j, uma utilização média aproximada, denotada por $U'_{CPU,j}$, pode ser calculada como:

$$U'_{CPU,j} = \frac{\sum_{i=1}^{\eta} \eta_i \times C_{i,j}}{T}. \tag{4}$$

Portanto, se observarmos as páginas Web processadas durante janelas de monitoramento T, 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 para estimar os valores $C_{i,j}$. Esses valores representam a utilização média de CPU requerida por cada tipo de página Web em cada camada do sistema. Na seção a seguir, mostraremos como usamos essa informação para estimar o consumo de energia requerido por cada tipo de página em cada camada de um sistema Web.

4. Demanda de Energia Requerida por uma Página Web

Nossa proposta leva em consideração apenas a potência/energia consumida pelo processador, uma vez que esse componente é o responsável pela maior parte da energia consumida nos servidores típicos [Beloglazov and Buyya 2012]. Para estimar a potência do processador, usamos uma abordagem tradicionalmente empregada [Isci and Martonosi 2003], 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:

$$P_{dynamic} \approx \alpha \times P_{active}. \tag{5}$$

$$P_{active} = P_{TDP} - P_{static}. \tag{6}$$

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

$$\alpha = \frac{U_{CPU}}{MAX_{U_{CPU}}}. \tag{7}$$

Em nosso modelo de sistema, assumimos que os servidores são equipados com processadores com múltiplos núcleos. Seja U_{core_k} a utilização média do núcleo k durante a janela de tempo T. Modelamos a utilização média de um processador com m núcleos no intervalo T como sendo a soma da utilização média de cada núcleo durante T, isto é:

$$U_{CPU} = \sum_{k=1}^{m} U_{core_k}. \tag{8}$$

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, 5 e 7, modelamos o consumo total de energia de um servidor com um processador com m núcleos como:

$$E = T \times (P_{static} + \frac{\sum_{k=1}^{m} U_{core_k}}{100m} \times P_{active}). \tag{9}$$

Por simplicidade, assumimos que um sistema Web com n camadas e formado por η tipos de páginas Web 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 páginas Web do tipo i processadas durante a janela de monitoramento, $1 \leq i \leq \eta$;
- $U_{CPU,j}$: a soma da utilização média de CPU de todos os núcleos de uma camada j durante a janela de monitoramento, $1 \leq j \leq n$;

Denotando as potências estática e ativa do servidor por P_{static} e P_{active}, respectivamente, e abstraindo a camada j como um servidor com m_j núcleos, podemos aplicar a Equação 9 e calcular a energia consumida pela camada j em uma janela de tempo T como:

$$E_j = T \times (P_{static,j} + \frac{\sum_{k_j=1}^{m_j} U_{core_{k_j}}}{100m_j} \times P_{active,j}), \tag{10}$$

onde:

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

Sejam $C_{i,j}$ e $T_{i,j}$, respectivamente, a utilização de CPU e o tempo médio necessários para processar uma página Web do tipo i na camada j. Assumindo que toda atividade realizada nessa camada é decorrência apenas do processamento de transações, podemos calcular a energia requerida para processar uma página Web desse tipo nessa camada como:

$$E_{\tau_i,j} = T_{i,j} \times \frac{C_{i,j}}{100m_j} \times P_{active,j}, \tag{11}$$

Como dentro de uma janela de monitoramento T o servidor processa η_i páginas Web do tipo i, a energia consumida por esse tipo de página na camada j durante T é $\eta_i \times E_{\tau_i,j}$. Logo, a soma da energia consumida por todas as páginas de todos os tipos na camada j durante T resulta na energia proveniente do consumo dinâmico dessa camada. Dessa forma, podemos reescrever a Equação 10 como:

$$E_j = T \times P_{static,j} + \sum_{i=1}^{\eta} \eta_i \times E_{\tau_i,j}. \tag{12}$$

Combinando as Equações 11 e 12, obtemos a Equação 13 para cada janela de monitoramento.

$$E_j = T \times P_{static,j} + \sum_{i=1}^{\eta} (\eta_i \times T_{i,j} \times \frac{C_{i,j}}{100 m_j} \times P_{active,j}). \tag{13}$$

Na Equação 13, 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 10. 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. Uma vez estimados os valores $C_{i,j}$, podemos substituí-los na Equação 13, para então aplicar um método de regressão linear e calcular os valores $T_{i,j}$. Finalmente, substituindo $C_{i,j}$ e $T_{i,j}$ por seus respectivos valores na Equação 11, podemos estimar a energia requerida por cada tipo de página Web em cada camada do sistema.

5. Avaliação de Desempenho

Para realizar a avaliação do modelo proposto, preparamos um ambiente de testes com um sistema *Web* multicamadas, descrito na Seção 5.1. Os resultados obtidos em experimentos que comparam os valores estimados pelo modelo de energia proposto e os valores retornados por um modelo implementado em hardware são apresentados na Seção 5.2.

5.1. Infraestrutura de Testes e Carga de Trabalho

Nosso ambiente de experimentação é baseado no TPC-W [TPC 2016], um *benchmark* que emula um *site* típico de comércio eletrônico seguindo uma arquitetura em três camadas. A Figura 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 camada de apresentação (`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.

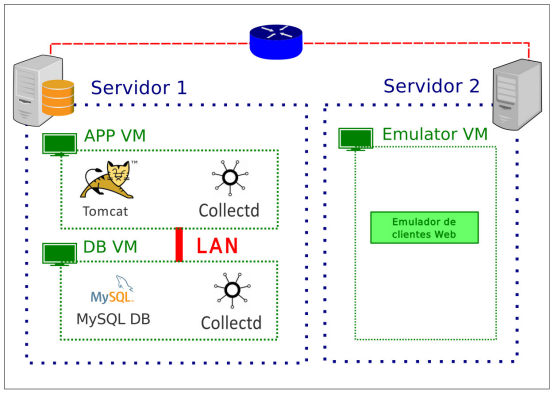


Figura 1. Arquitetura do ambiente de testes.

VM	Amazon EC2	VCPU	RAM
`APP`	t2.medium	2	4GB
`DB`	t2.medium	2	4GB
`Emulator`	t2.small	1	2GB

Tabela 1. Configurações das máquinas virtuais.

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`) também executam a ferramenta collectd [collectd 2016] para realizar as coletas de dados. A Tabela 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].

Como mostrado na Tabela 2, o TPC-W possui 14 páginas Web, classificadas como requisições relacionadas com a navegação de clientes no *site* ou como requisições de compras de produtos. Essas páginas Web são combinadas para compor três tipos de carga de trabalho, a saber: *Browsing* (95% de navegação e 5% de compra), *Shopping* (80% de navegação e 20% de compra) e *Ordering* (50% de navegação e 50% de compra).

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 2. Páginas do TPC-W de acordo com cada tipo de requisição.

Tipicamente, cargas Web não são limitadas por CPU, uma vez que sistemas Web tendem a executar um grande número de operações de entrada/saída [Wang et al. 2013]. 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

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

maior que a *Browsing*. *Shopping*, por sua vez, é uma carga de trabalho intermediária em relação às duas cargas anteriores.

O TPC-W simula sessões de clientes Web através de emuladores denominados *Remote Browser Emulators* (RBEs). Durante um experimento, o número de RBEs que executam de forma concorrente é mantido constante. Além disso, para cada RBE, o TPC-W define, 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.

Em nossos experimentos, utilizamos os três tipos de carga do TPC-W. Para emular intensidades de cargas diferentes, executamos um conjunto de experimentos variando o número de RBEs concorrentes em 100, 200, 300, 400 e 500. Executamos cada experimento por 5 horas e 40 minutos, uma vez que esse tempo de vida se mostrou adequado para gerar dados suficientes para o uso da técnica de regressão. 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. As Figuras 2 e 3 ilustram a média de utilização de CPU para cada valor de RBE e para as camadas de aplicação e dados, respectivamente. Nesse experimento, os dados foram coletados a cada 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 quanto maior for o número de requisições do tipo `Compra` em uma carga, maior a demanda por CPU na camada de dados. Assim, nessa camada, a carga *Browsing* requer uma demanda por CPU menor que a *Shopping* que, por sua vez, requer uma demanda menor que a *Ordering*.

A carga *Ordering*, diferentemente das demais, apresenta uma curva de utilização média de CPU que culmina em 300 RBEs. 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, a carga *Ordering* utilizada tem limite máximo de 300 RBEs. Para as cargas *Browsing* e *Shopping* observamos comportamento semelhante para RBEs maiores que 500.

A Figura 4 ilustra a energia média consumida pelo processador do `Servidor 1` durante a execução dos experimentos envolvendo os três tipos de cargas. Esse dado foi coletado usando a interface RAPL [David et al. 2010], um modelo em hardware de estimação de potência para todo o chip do processador e disponível nos processadores Intel mais recentes. Nesses experimentos, programamos a interface para realizar as coletas a cada 15 segundos. Esse valor menor foi escolhido para evitar a sobrescrita dos contadores usados pela interface. Para obter a medição de energia dentro de uma janela de 1 minuto, agregamos 4 janelas consecutivas. 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.

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

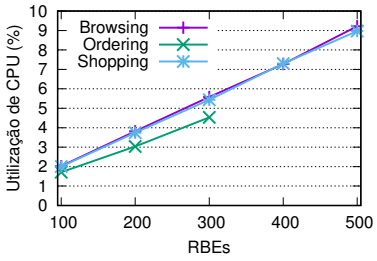


Figura 2. Utilização de CPU para a camada de aplicação.

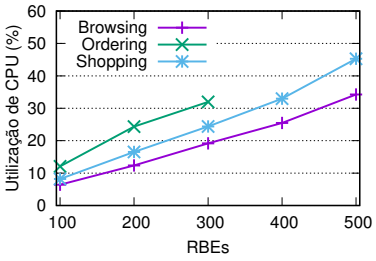


Figura 3. Utilização de CPU para a camada de dados.

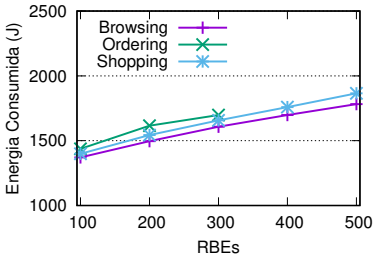


Figura 4. Consumo de energia durante os experimentos.

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). Embora esses dois últimos valores não são usados pelo modelo, eles foram necessários para sua validação. Durante os experimentos, foi utilizado o algoritmo de regressão linear múltipla disponível na ferramenta de computação estatística `R`. Inicialmente, cada carga de trabalho foi executada duas vezes, resultando em 10 horas de execução para cada carga 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 o teste da regressão.

5.2. Resultados

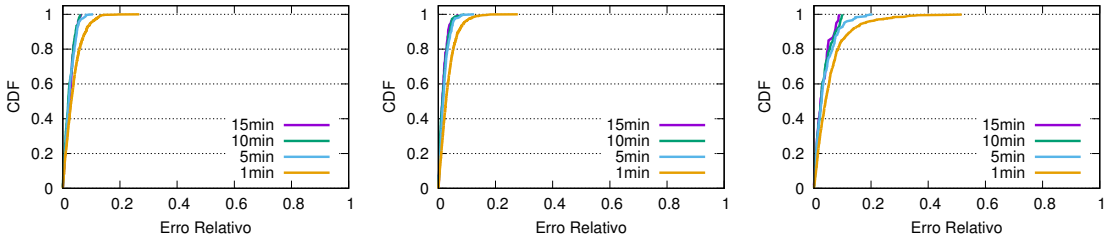
Nas figuras a seguir, os resultados dos experimentos são avaliados para diferentes tamanhos de janelas de monitoramento. Os valores das janelas bem como dos demais parâmetros usados no modelo estão ilustrados na Tabela 3.

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

Tabela 3. Valores dos parâmetros utilizados nos experimentos.

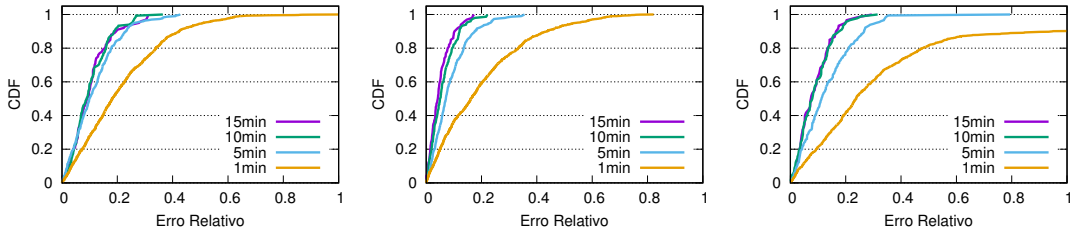
Inicialmente, avaliamos a acurácia do modelo para estimar a demanda de CPU para cada página Web em uma determinada camada, ou seja, os valores $C_{i,j}$. Para isso, utilizamos o erro relativo entre os valores $U'_{CPU,j}$ e $U_{CPU,j}$. O primeiro valor representa a utilização média prevista pelo modelo 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 foi observada na camada j durante os experimentos. O erro relativo é então obtido da seguinte forma: $\epsilon_{CPU} = \frac{|U'_{CPU,j} - U_{CPU,j}|}{U_{CPU,j}}$.

As Figuras 5 e 6 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. Isso ocorre porque nessa camada a utilização de CPU não tem grandes variações. Observamos também que, em geral, janelas de monitoramento



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

Figura 5. CDFs de ϵ_{CPU} para a camada de aplicação (APP VM).



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

Figura 6. CDFs de ϵ_{CPU} para a camada de dados (DB VM).

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} nos melhores cenários (janela de 15 minutos) são aproximadamente 10%, 5%, 8% para as cargas *Browsing*, *Shopping* e *Ordering*, respectivamente.

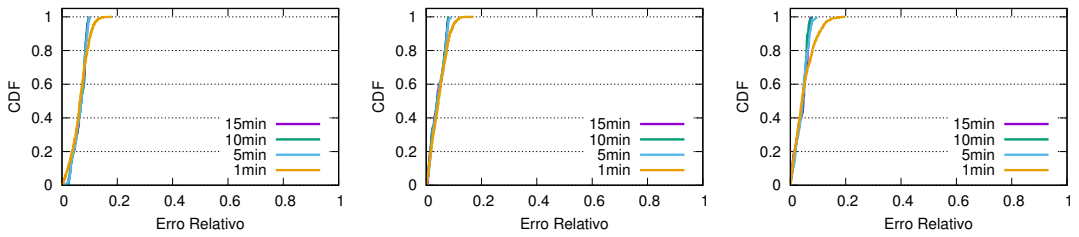
Em seguida, avaliamos a acurácia do modelo para estimar a demanda de energia para cada página Web em uma determinada camada, ou seja, os valores $T_{i,j}$. Como o RAPL não é capaz de reportar o consumo de energia por núcleos do processador, nossa avaliação leva em consideração a energia consumida por 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: $E' = E_0 + \sum_{j=1}^{n} E_j$. Assumindo que existam m_0 núcleos do servidor que não estão sendo usados por nenhuma camada no sistema, E_0 é dado por:

$$E_0 = T(P_{static,0} + \frac{\sum_{k_0=1}^{m_0} U_{core_{k_0}} P_{active,0}}{100 m_0}), \tag{14}$$

onde:

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

O erro relativo para a regressão de energia é então obtido da seguinte forma: $\epsilon_{energy} = \frac{|E'-E|}{E}$.



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

Figura 7. CDFs de ϵ_{energy} para diferentes cargas.

T (mins)	Browsing	Shopping	Ordering
1	0,065	0,046	0,049
5	0,065	0,042	0,042
10	0,065	0,041	0,040
15	0,065	0,041	0,041

Tabela 4. Valor médio de ϵ_{energy} em cada cenário avaliado.

A Figura 7 ilustra as CDFs de ϵ_{energy} para as cargas *Browsing*, *Ordering* e *Shopping* em diferentes tamanhos de janelas de monitoramento. A Tabela 4 mostra o valor médio de ϵ_{energy} em cada cenário ilustrado na Figura 7. 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 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. Esses erros estão na mesma ordem de grandeza dos apresentados em [Piga et al. 2014].

6. Conclusão

Este trabalho apresentou a proposta e avaliação de um modelo analítico para estimar o consumo de energia de sistemas Web no nível de transações. Utilizando o *benchmark* TPC-W, conseguimos mostrar através de experimentos reais, que o entendimento e o controle mais finos dos componentes do sistema ajudam a construir modelos simples e acurados para estimação de consumo de energia de ambientes com grande demanda computacional. Baseando-se em tais informações, nosso modelo foi capaz de estimar o consumo de energia com erros variando entre entre 4% e $6,5\%$. Esses valores são compatíveis com o estado da arte na área de eficiência energética em *data centers*. Nosso método de estimação de energia diferencia-se por: i) modelar o comportamento de transações Web em vez de sessões; ii) requerer poucas métricas, sendo essas de fácil obtenção; e iii) ser

baseado em um método de predição simples, como é o caso da regressão linear.

Como trabalhos futuros, pretendemos implementar o trabalho de Piga et al. com o intuito de compará-lo com o nosso trabalho sob as mesmas condições. Pretendemos também avaliar o impacto do uso de VMs sob o consumo de energia do servidor. Por fim, também temos como meta o estudo de técnicas estatísticas mais elaboradas para a predição do consumo energético de outras cargas de trabalho também comum em *data centers* de computação em nuvem.

Referências

Barroso, L. A. and Hölzle, U. (2007). The case for energy-proportional computing. *Computer*, 40(12):33–37.

Bedard, D., Lim, M. Y., Fowler, R., and Porterfield, A. (2010). Powermon: Fine-grained and integrated power monitoring for commodity computer systems. In *Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon)*, pages 479–484.

Beloglazov, A. and Buyya, R. (2012). Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurr. Comput. : Pract. Exper.*, 24(13):1397–1420.

Bohrer, P., Elnozahy, E. N., Keller, T., Kistler, M., Lefurgy, C., McDowell, C., and Rajamony, R. (2002). The case for power management in web servers. In *Power aware computing*, pages 261–289.

Cochran, R., Hankendi, C., Coskun, A. K., and Reda, S. (2011). Pack & Cap: Adaptive DVFS and Thread Packing Under Power Caps. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 175–185.

collectd (2016). Collectd – The system statistics collection daemon. `https://collectd.org`. Último acesso: oct-14-2015.

David, H., Gorbatov, E., Hanebutte, U. R., Khanna, R., and Le, C. (2010). Rapl: Memory power estimation and capping. In *Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design*, pages 189–194.

Fan, X., Weber, W.-D., and Barroso, L. A. (2007). Power provisioning for a warehouse-sized computer. In *Proceedings of the 34th Annual International Symposium on Computer Architecture*, pages 13–23.

Ge, R., Feng, X., Song, S., Chang, H.-C., Li, D., and Cameron, K. W. (2010). Powerpack: Energy profiling and analysis of high-performance systems and applications. *IEEE Transactions on Parallel Distributed Systems*, 21(5):658–671.

Hsu, C.-h. and Feng, W.-c. (2005). A power-aware run-time system for high-performance computing. In *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, pages 1–.

Isci, C., Buyuktosunoglu, A., Cher, C.-Y., Bose, P., and Martonosi, M. (2006a). An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 347–358.

Isci, C., Contreras, G., and Martonosi, M. (2006b). 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*, pages 359–370.

Isci, C. and Martonosi, M. (2003). Runtime power monitoring in high-end processors: Methodology and empirical data. In *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 93–.

Joseph, R. and Martonosi, M. (2001). Run-time power estimation in high performance microprocessors. In *Proceedings of the 2001 International Symposium on Low Power Electronics and Design*, pages 135–140.

Kaxiras, S. and Martonosi, M. (2008). *Computer Architecture Techniques for Power-Efficiency*. Morgan and Claypool Publishers, 1st edition.

Menasce, D. A., Dowdy, L. W., and Almeida, V. A. F. (2004). *Performance by Design: Computer Capacity Planning By Example*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Metri, G., Srinivasaraghavan, S., Shi, W., and Brockmeyer, M. (2012). Experimental analysis of application specific energy efficiency of data centers with heterogeneous servers. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 786–793.

Mi, N., Casale, G., Cherkasova, L., and Smirni, E. (2010). Sizing multi-tier systems with temporal dependence: benchmarks and analytic models. *Journal of Internet Services and Applications*, 1(2):117–134.

Piga, L., Bergamaschi, R. A., and Rigo, S. (2014). Empirical and analytical approaches for web server power modeling. *Cluster Computing*, 17(4):1279–1293.

Rotem, E., Naveh, A., Ananthakrishnan, A., Weissmann, E., and Rajwan, D. (2012). Power-management architecture of the intel microarchitecture code-named sandy bridge. *IEEE Micro*, 32(2):20–27.

Rountree, B., Lownenthal, D. K., de Supinski, B. R., Schulz, M., Freeh, V. W., and Bletsch, T. (2009). Adagio: Making DVS Practical for Complex HPC Applications. In *Proceedings of the 23rd International Conference on Supercomputing*, pages 460–469.

Scaramella, J., Marden, M., Daly, J., and Perry, R. (2014). The cost of retaining aging it infrastructure. Technical report, International Data Corporation (IDC), Framingham, MA.

TPC (2016). TPC-W Benchmark. `http://www.tpc.org`. Último acesso: oct-14-2015.

Wang, Q., Kanemasa, Y., Li, J., Lai, C. A., Matsubara, M., and Pu, C. (2013). Impact of DVFS on N-tier Application Performance. In *Proceedings of the First ACM SIGOPS Conference on Timely Results in Operating Systems*, pages 5:1–5:16.

Zhang, Q., Cherkasova, L., and Smirni, E. (2007). A regression-based analytic model for dynamic resource provisioning of multi-tier applications. In *Proceedings of the Fourth International Conference on Autonomic Computing*, pages 27–.