Previsão de Carga para Economia de Energia em Aglomerados de Servidores *Web*

Carlos Henrique Sant' Ana[1], J.C.B Leite[1], Daniel Mossé[2]

[1]Instituto de Computação
Universidade Federal Fluminense – Niterói, RJ – Brasil

[2]Department of Computer Science
University of Pittsburgh – Pittsburgh, PA – EUA

{csantana, julius}@ic.uff.br, mosse@cs.pitt.edu

Abstract. *The complexity and requirements of web applications are increasing in order to meet more sophisticated business models (web services and cloud computing, for instance). For this reason, characteristics such as performance, scalability and security are addressed in web server cluster design. However, due to the energy crisis, the energy consumption in this type of environment became a main concern. Aiming to avoid this adverse scenario, this article shows how energy reduction can be applied in a web server clustered environment. The energy reduction policy used in this paper uses load forecasting combined with DVFS (Dynamic Voltage and Frequency Scaling) and dynamic configuration techniques. To validate this predictive policy, a web application running a real workload profile was submitted to a server cluster testbed. In addition a criteria for quality of service was used to evaluate system's performance.*

Resumo. *A complexidade e os requisitos das aplicações web vêm aumentando a fim de satisfazer modelos de negócios cada vez mais sofisticados (web services e computação em nuvem, por exemplo). Por esta razão, características como desempenho, escalabilidade e segurança são tratados no projeto de aglomerados de servidores web. Entretanto, devido à crise energética, o consumo de energia neste tipo de ambiente se tornou uma grande preocupação. A política de redução de energia em um aglomerado de servidores web utilizada neste trabalho emprega previsão de carga combinada com as técnicas de DVFS (Dynamic Voltage and Frequency Scaling) e de configuração dinâmica. Para validar esta política preditiva, uma aplicação web executando um perfil de carga real foi testada em um protótipo de cluster de servidores, e a energia consumida e a qualidade de serviço obtida foram usadas para avaliar o desempenho do sistema.*

1. Introdução

Com o passar dos anos, o consumo de energia elétrica vem tornando-se uma preocupação cada vez mais importante no desenvolvimento de novas tecnologias, pois sua produção representa uma importante questão ambiental e econômica. Segundo projeções feitas pela Agência Internacional de Energia [IEA 2008] (*International Energy Agency* ou IEA), a emissão de gases que ocasionam o efeito estufa vai dobrar no fim deste século, causando

dessa forma um aumento de 6° C na temperatura global. Como em muitos países a principal fonte de produção de energia elétrica são os combustíveis fósseis, medidas de largo alcance estão sendo tomadas para que essa crise energética seja superada.

A EPA (Agência de Proteção Ambiental dos Estados Unidos) divulgou recentemente um relatório [EPA 2007] no qual cita uma série de medidas necessárias para promover o aumento da eficiência nos *datacenters*. Também chamados de Centros de Processamento de Dados (CPD) ou fazenda de servidores, esta infra-estrutura reúne vários computadores para a execução de processamento, armazenamento, comunicação de dados, dentre outras funções. A justificativa para a escolha desse tipo de infra-estrutura como foco desse relatório são as projeções feitas em relação ao seu consumo de energia. Estima-se que o crescimento do consumo de energia pelas fazendas de servidores, somente nos Estados Unidos, irá efetivamente dobrar até 2011, passando de 61,4 para 124,5 bilhões de KWh gastos por ano.

No nível de hardware essa preocupação pode ser comprovada com a criação da ACPI [ACPI 2008]. Representando a união entre empresas tidas como referências no mercado (HP, Intel, Microsoft, Phoenix e Toshiba), elas estabeleceram uma especificação para gerenciamento de energia e temperatura em computadores de maneira geral. Outra iniciativa de destaque é o programa *Energy Star* da EPA [EPA 2009]. Este programa visa a adoção de práticas e produtos com maior eficiência energética tanto por empresas quanto pela população de maneira geral.

Seguindo esta tendência de pesquisa de novos métodos para a economia de energia, esse trabalho descreve uma política para redução no consumo energético em ambientes computacionais chamados de aglomerado de servidores *web*. Empregando mecanismos de previsão de carga (uma estimativa de um valor para um instante no futuro) no ajuste da configuração do sistema computacional (ou seja, ligando e desligando servidores e ajustando a velocidade de seus processadores), a escolha daquela mais eficiente para uma dada carga pode ser feito de forma pró-ativa e não reativa. Dessa forma, um incremento no ganho com relação a economia de energia pode ser obtido, e a qualidade de serviço (QoS) contratada pela aplicação ainda ser satisfatoriamente atendida.

Uma outra contribuição desse trabalho é a forma pela qual a manutenção da qualidade de serviço está relacionada aos atributos do sistema. Assume-se que a manutenção de um determinado nível de QoS baseia-se tão somente na utilização do processador dos servidores, limitando-a a um valor máximo; essa política torna-se aplicável em uma maior variedade de ambientes, visto que essa métrica é independente do tipo de serviços executados no sistema computacional em questão.

Na próxima Seção, trabalhos relacionados da área de economia de energia são discutidos. Em seguida, a Seção 3 apresenta a arquitetura do *cluster* considerado nos experimentos. A Seção 4 discute o modelo de previsão de carga empregado, e na Seção seguinte é feita uma avaliação do modelo tendo como base um *trace* derivado de uma aplicação real. A Seção 6 apresenta as conclusões e indica trabalhos a serem futuramente realizados.

2. Trabalhos Relacionados

Em função de sua crescente importância, a área de Economia de Energia em sistemas computacionais vem ganhando notoriedade e uma base mais abrangente de aplicações.

Por exemplo, em dispositivos móveis (*laptops* e PDAs), restrições referentes ao consumo de energia sempre obtiveram notoriedade. Em função de um crescente poder de processamento, que vem acompanhado de um maior consumo energético, eles precisam maximizar o tempo de vida de sua fonte de energia (bateria), já que durante boa parte de seu tempo de uso não possuem uma ligação com uma fonte fixa. Com esse tipo de restrição em vista, além da redução do consumo de energia obtida colocando-se alguns componentes de hardware em estado de baixo consumo (*display* e disco, por exemplo), trabalhos tais como [Flinn e Satyanarayanan 1999] procuraram adicionar a alguns tipos de aplicações móveis um perfil adaptativo em relação ao estado da capacidade de fornecimento de energia pela bateria.

Além do processador, outros tipos de componentes e sistemas de hardware também evoluíram na questão da economia de energia. Nessa área podemos citar trabalhos em dispositivos de armazenamento [Ganesh et al. 2007], sistemas de memória RAM [AbouGhazaleh et al. 2007, Tolentino et al. 2009], dispositivos de rede [Bertozzi et al. 2002], dentre outros. Contudo, como o processador possui um gasto energético substancialmente maior do que o de outros componentes de um computador, em função de sua utilização, [Bohrer et al. 2002] e outros trabalhos que ainda serão mencionados têm a CPU (*Central Processing Unit*) como seu principal foco.

Cenários envolvendo aplicações do tipo multimídia também foram explorados, como, por exemplo, no estudo apresentado em [Yuan e Nahrstedt 2003]. Este trabalho discute a construção de um escalonador para sistemas de tempo-real não críticos (*soft real-time*) que tem como objetivo economizar energia a partir da previsão dos ciclos demandados por uma aplicação multimídia. Outro estudo nesse mesmo contexto é apresentado em [Kim e Chang 2005], e descreve o uso de um algoritmo DVFS aplicado a um decodificador de áudio e vídeo.

Maiores oportunidades para economia de energia aparecem, contudo, em ambientes que utilizam grande quantidade de computadores. Devido à natureza das aplicações existentes na atualidade, ambientes tais como fazendas de servidores e aglomerados de computadores (daqui em diante *clusters* será utilizado como sinônimo) ganharam notoriedade. As técnicas desenvolvidas para esse tipo de sistema consideraram, inicialmente, somente *clusters* compostos por computadores homogêneos [Elnozahy et al. 2002, Pinheiro et al. 2001], onde o uso de configuração dinâmica (ligar e desligar máquinas do *cluster* de acordo com a utilização do mesmo) baseada em uma política de gerenciamento global obteve resultados significativos.

Uma das primeiras iniciativas utilizando um ambiente computacional heterogêneo foi feita por [Rusu et al. 2006]. Nesse trabalho, as técnicas de DVFS e configuração dinâmica são combinadas com o propósito de redução de energia associada à manutenção da qualidade de serviço provida para uma aplicação *web*. Nesse caso, o *cluster* considerado é composto por duas camadas, uma sendo o *front-end*, que distribui a carga e gerencia a configuração, e a outra abrangendo as camadas de aplicação e base de dados. O trabalho pressupõe que a maioria das requisições são atendidas a partir da *cache* de dados em memória, o que realça a importância do controle de energia em nível de processador (DVFS).

O estudo apresentado em [Horvath et al. 2006] discute uma infra-estrutura na qual

estão presentes três camadas: interface, negócios e armazenamento de dados. Minimizando o atraso total entre camadas, um modelo utilizando um controlador com realimentação é responsável pela execução do DVFS. Nesse trabalho não há menção de configuração dinâmica porque somente um *path* foi utilizado na avaliação. Por *path* os autores entendem uma máquina na segunda camada e uma máquina na terceira camada. Esse trabalho, contudo, é baseado em um modelo de filas, e só foi avaliado via simulação.

Também utilizando uma aplicação *web* de três camadas, o trabalho apresentado em [Bertini et al. 2007] mostra um modelo no qual a gerência de economia de energia leva em consideração os níveis QoS contratados pela aplicação, assumindo um sistema do tipo *soft real-time*. Com base em uma grandeza chamada de *tardiness*, um controlador consegue manter a qualidade de serviço acima de um patamar pré-estabelecido. Por *tardiness* os autores entendem a relação entre o tempo de resposta de cada requisição (medido durante a execução) dividido pelo seu prazo de execução (estabelecido nas especificações). O trabalho apresenta duas formas de modelar a função de distribuição dessa variável, e mostra como garantir que um dado percentil das requisições irá atender ao contratado. Posteriormente, em [Bertini et al. 2009], os mesmos autores estendem o trabalho anterior e realizam o controle de QoS e de consumo de energia sem fazer nenhuma hipótese sobre a forma da distribuição da variável *tardiness*, através do uso de aproximação estocástica.

Buscando um melhor aproveitamento e racionalização dos recursos providos por um *datacenter*, o emprego do mecanismo de virtualização tem obtido grande destaque. Além de prover grandes benefícios (escalabilidade, melhor gerenciamento e utilização do hardware), o uso deste tipo de técnica traz novos desafios para a implementação de redução no consumo de energia [Wang et al. 2008]. Apesar de não ser o escopo deste trabalho, a metodologia utilizada para realizar a previsão de carga pode servir como ferramenta para técnicas existentes de economia de energia em ambientes virtualizados [Kusic 2008, Petrucci et al. 2009].

O estudo desenvolvido aqui visa avançar o estado da arte, incorporando técnicas de previsão de carga às técnicas de economia de energia. Nesse caso, um sistema que consiga estimar a evolução da carga de trabalho submetida possibilita que medidas direcionadas tanto para economia de energia, quanto para manutenção da qualidade de serviço, possam ser tomadas. Ou seja, uma política de economia de energia baseada nesse paradigma passará de reativa a pró-ativa.

Além de fatores que influenciam o método de previsão, elementos característicos que ocorrem durante o tempo de execução de um sistema são mostrados e analisados (Seção 5). Diferentemente de outros trabalhos, muitos deles simplesmente baseados em simulação, nossa implementação ocorre em um protótipo de *cluster*, que é submetido a um perfil de carga real: um trecho do *workload* da Copa do Mundo de Futebol de 1998 [Arlitt e Jin 1999]. Apesar de o grau de previsibilidade desta carga de trabalho não ser dos mais difíceis [Baryshnikov et al. 2005], diversos desafios existem quando se faz a implementação em um ambiente real, principalmente pelo fato de que esse perfil não obedece a nenhuma distribuição conhecida, em particular.

3. Arquitetura do *cluster*

Na arquitetura utilizada para a construção do ambiente de testes existem três categorias de máquinas: o *front-end* (FE), os trabalhadores (*workers*) e as máquinas de suporte. A máquina *front-end* é aquela responsável por executar os procedimentos contidos no modelo, como o gerenciamento das máquinas trabalhadoras, o balanceamento de carga e a comunicação com as máquinas de suporte. Como estamos interessados em avaliar um esquema de relacionamento entre QoS e utilização do *cluster*, um sistema de duas camadas é aqui considerado.

Os *workers* são os responsáveis pelo processamento da carga enviada ao *cluster*, ou seja, pelo processamento de requisições *web*. Já as máquinas de suporte são aquelas que realizam funções que não estão relacionadas com as do FE e nem com as dos *workers*. Como indicado na Figura 1 elas representam o gerador de carga e o medidor de potência.

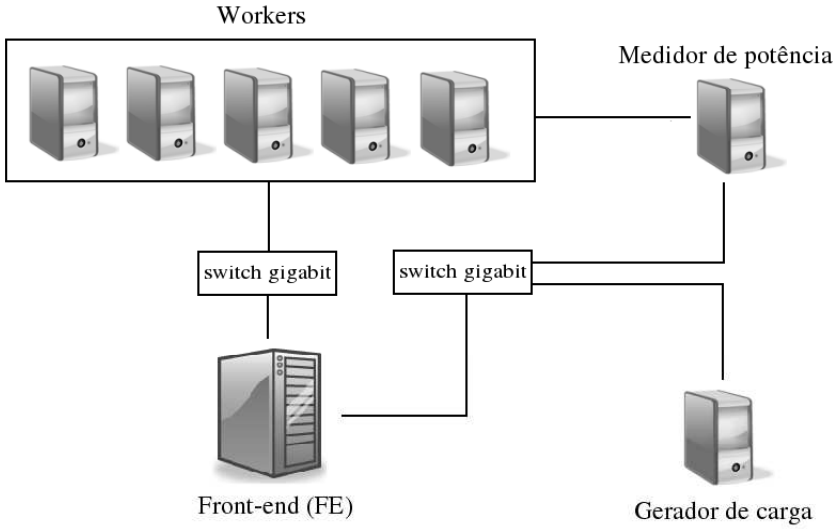

Figura 1. Organização e comunicação das máquinas do *cluster*

O FE e todos os *workers* executam Linux e servidores Apache, que são responsáveis pelo processamento dos pedidos feitos pelo gerador de carga. O que difere o FE de cada trabalhador são os módulos ativados em cada um dos servidores. Nos *workers* está ativado o módulo de processamento de requisições PHP e *worker_module*. A necessidade da utilização deste último módulo é devida a informações que estão contidas somente no escopo da aplicação *web* (como número de requisições pendentes em um determinado *worker*, por exemplo).

O *front-end* por sua vez possui o *frontend_module*, que é responsável por implementar todas as funções do FE, exceto o balanceamento de carga (feito pelo módulo *mod_proxy_balancer* [Apache 2009]). O lado cliente da aplicação é representado por uma versão modificada do gerador de carga *httperf* [Mosberger e Jin 1998], capaz de realizar a geração de requisições *web* de acordo com dados contidos em um aquivo de *trace*.

O medidor de potência é a única máquina que executa o sistema operacional Microsoft Windows XP, por conta do programa relacionado a medição de potência (LabView). Foram também instalados nessa máquina um servidor OpenSSH e um módulo

Tabela 1. Máquinas utilizadas na arquitetura e suas funções

Nome	Função	# Freq.	Freq. máx. (GHz)	Arquitetura
Watt	FE	–	–	AMD Athlon 64 3200+
Ampere	*worker*	3	2,0	AMD Athlon 64 X2 3800+
Coulomb	*worker*	5	2,4	AMD Athlon 64 3800+
Hertz	*worker*	5	2,2	AMD Athlon 64 3800+
Joule	*worker*	4	2,4	AMD Athlon 64 3500+
Ohm	*worker*	6	2,6	AMD Athlon 64 X2 5000+
Camburi	gerador de carga	–	–	Pentium 4 2,8GHz
Putiri	medidor de potência	–	–	Pentium 4 2,8GHz

VNC (*Virtual Network Computing*), para que a comunicação com o ambiente Linux ocorresse, respectivamente, via console ou *desktop* remoto. Outras características a respeito das máquinas empregadas nessa arquitetura podem ser vistas na Tabela 1.

4. Modelo para previsão de carga

Dentre as técnicas existentes para realizar previsão de séries temporais podem ser citadas as médias móveis, o amortecimento exponencial, as regressões, as redes neurais e modelos não lineares, tais como os da família ARCH (*Autoregressive Conditional Heteroskedasticity*). O previsor escolhido nesse trabalho foi o Método Linear de Holt [Spyros Makridakis e Hyndman 2002] (HLM ou *Holt Linear Method*). Esse modelo é definido a partir de um conjunto de três equações:

$$L_t = \alpha Y_t + (1-\alpha)(L_{t-1} + b_{t-1}) \quad (1)$$

$$b_t = \beta(L_t - L_{t-1}) + (1-\beta)b_{t-1} \quad (2)$$

$$F_{t+m} = L_t + b_t m \quad (3)$$

A Equação 1 mostra como é calculada uma estimativa do valor da série temporal, a partir de uma amostra Y_t obtida no instante t . A Equação 2 calcula b_t, isto é, uma noção de tendência positiva ou negativa apresentada pelos dados. Utilizando os valores definidos nas duas equações anteriores, a Equação 3 calcula a previsão para o instante $t + m$, onde m representa a janela de previsão. As variáveis α e β representam os coeficientes de amortecimento. Devido a existência desses coeficientes, o HLM é classificado como um método de amortecimento duplo. Esse previsor foi adotado pela simplicidade de implementação em um ambiente real, por precisar somente de um pequeno histórico para realizar previsões e por ser indicado na literatura como um previsor adequado para situações nas quais existe presença de tendência e ausência de sazonalidade. Outros métodos de previsão poderiam ser utilizados, tais como regressões ou redes neurais.

Após definido o método previsor, a escolha da grandeza alvo desse método é o próximo passo. Nesse caso, a carga a qual o sistema está sendo submetido foi selecionada. Ao longo do tempo, amostras do valor da carga (Y_t) são coletadas pelo FE a fim de fornecer os dados para alimentar o previsor HLM. Depois do tempo necessário para

coletar um número de amostras desejadas (**janela de decisão**) o previsor será invocado e uma nova previsão para o valor da carga calculada (F_{t+m}). A partir dessa estimativa, o *front-end* repassa essa valor para um modelo de otimização utilizando DVFS chaveado, como o indicado em [Bertini et al. 2010] (outros detalhes em [Bertini et al. 2008]).

Obter uma configuração do *cluster* para um determinado nível de carga é uma tarefa simples quando uma pequena quantidade de máquinas *workers* está envolvida. Para um número maior de *workers* (como 50 e 100, por exemplo) observou-se, no pior caso de execução de cada um dos programas de otimização utilizados [COIN-OR 2009, GLPK 2009, SCIP 2009], tempos superiores a 5 horas, indicando não ser prático executar esse processamento *on-line*. Sendo assim, a construção de uma tabela com valores de carga *vs.* configuração foi construída em tempo de projeto, para que somente uma consulta seja feita durante o funcionamento do sistema.

Para a construção da tabela de possíveis configurações para o *cluster* foi utilizado o seu tamanho total ao invés de valores pré-determinados para a carga. Essa abordagem permite que cada linha da tabela mostre a porcentagem da carga na qual haverá mudanças de frequência ou configuração do *cluster* (quais máquinas estarão ligadas). A Tabela 2 mostra um exemplo dessa construção para uma tabela que contém uma configuração para o aglomerado referente a variação de um milésimo no valor da carga máxima. Para os computadores que devem estar desligados o valor de frequência de operação igual a zero aparece na tabela.

Tabela 2. Exemplo de tabela de otimização gerada

Carga (%)	*ampere*	*coulomb*	*hertz*	*joule*	*ohm*
0,1	0	0	1000	0	0
0,2	0	0	1000	0	0
...	...	...	...	...	...
...	...	...	...	...	...
60	2000	0	1800	0	1971
...	...	...	...	...	...
100	2000	2400	2400	2200	2600

Tendo em vista que o processo descrito até aqui leva em consideração somente a minimização do consumo de energia do *cluster*, uma solução para atender a qualidade de serviço representa o próximo passo da política a ser implementado. Nesse trabalho, estamos definindo o tempo de resposta das requisições como a QoS a ser atendida. Para cada requisição pode-se associar um prazo (*deadline*), como em um sistema *soft real-time*. Por simplificação, somente um tipo de requisição é assumido, embora a extensão não seja difícil de implementar.

Experimentos preliminares indicaram que uma solução heurística, através da aplicação de uma parcela multiplicativa sobre a estimativa do valor futuro da carga, em função do atendimento ou não a uma QoS contratada, no momento de avaliação de uma nova configuração, pode ser uma solução adequada. Para isso, foi definido um fator de correção de previsão (chamado de γ, ou gama, daqui por diante).

A função desse fator é acionar uma configuração de maior desempenho de acordo com os níveis de QoS obtidos durante a execução do sistema. A fim de cumprir essa meta,

esse fator é definido como $\gamma = min(\gamma_{max}, (QoS_{alvo} \div QoS_{atual}))$, caso a QoS não esteja sendo atendida. Se a qualidade de serviço estiver de acordo com a contratada, o valor de γ não será calculado pela equação acima e a ele é atribuído o valor 1. Utilizando essa definição, o fator gama irá provocar a utilização de uma configuração de maior desempenho de acordo com o nível da qualidade de serviço atual do sistema e garantir que configurações com maior poder computacional serão chamadas gradualmente (de acordo com a parcela γ_{max}).

Adicionalmente, é assumido que limitando-se a utilização do servidor a um valor menor do que 100%, a QoS será mais adequadamente atendida. Vários estudos constataram que níveis de utilização superiores a 60-80% tornam o sistema suscetível a ser mais influenciado por variações súbitas de carga, piorando a QoS (e.g., [Rusu et al. 2006]). Assim, o valor de gama anteriormente obtido é dividido por uma valor de utilização limite (e.g., 80%), de forma a reduzir a citada influência.

Depois de calculado o valor de γ utilizando o resultado da previsão, uma nova configuração será escolhida e imposta ao *cluster*. Através de mensagens enviadas pela rede, o FE ajusta a frequência das máquinas que estão ligadas, liga as máquinas que possuem um valor de frequência maior que 0 na tabela e depois começa a desligar aquelas com frequência nula. A gerência desse processo é feita em intervalos regulares de tempo chamados de **janelas de controle**.

Com o intuito de impedir sucessivas operações de ligar e desligar máquinas em um intervalo muito curto, um novo intervalo de tempo foi adotado. A **janela de estabilização** é um período de tempo maior que outras as janelas definidas anteriormente, cuja a meta é deixar que o sistema estabilize após grandes mudanças sofridas na configuração (entrada e saída de *workers* no conjunto das máquinas ligadas).

No processo de ligar e desligar uma máquina, o tempo de *boot* (ou iniciação) é um fator que deve ser levado em consideração. A utilização do mecanismo de *Suspend-to-RAM* (em conjunto com *Wake-on-LAN*), em contraposição ao uso de *Suspend-to-disk*, faz esse tempo de *boot* diminuir cerca de 8 vezes (de 56s para 7,5s) de acordo com experimentos realizados em nosso ambiente de testes.

5. Avaliação

Para realizar a avaliação do modelo preditivo proposto na seção anterior, vários experimentos foram conduzidos reproduzindo o perfil da Copa do Mundo de Futebol de 1998. O período que serviu como base para gerar a carga de trabalho é o compreendido entre 11h30 e 24h00 do dia 29 de Junho. Com o intuito de aumentar o número de experimentos realizados e diminuir a probabilidade de interferências externas (e.g., falha no fornecimento de energia pela concessionária local), a taxa de reprodução desse *trace* foi acelerada em 3 vezes. Isto implica que a duração de cada um dos experimentos foi reduzida de 12 horas e meia para 4 horas e 10 minutos (ou 250 min). É importante observar que essa aceleração também permite atualizar a taxa de requisições para o que acreditamos seja mais próximo daquela dos CPDs modernos. A forma da carga submetida ao sistema pode ser observada na Figura 2. Adicionalmente, aqui foi considerado um único tipo de requisição, com prazo de execução limitado a 4s, de forma a tornar o atendimento à demanda realizável no pico da carga. Esse valor é típico das requisições definidas no *benchmark* TPC-W [TPC 2009].

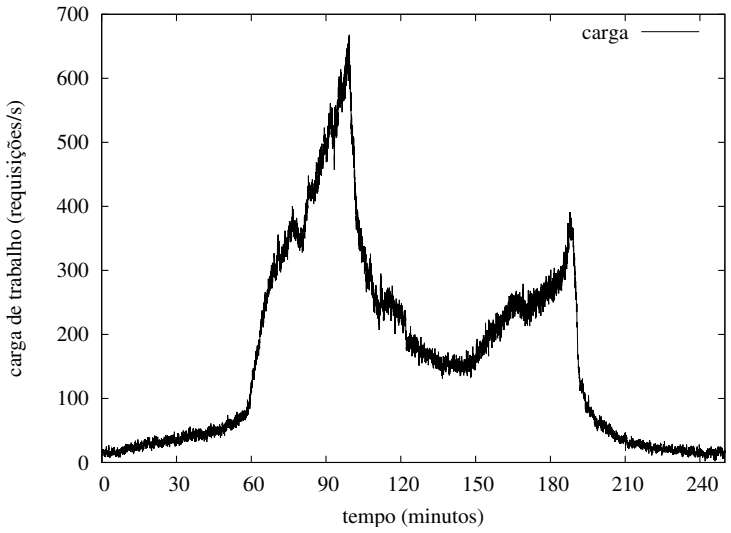

Figura 2. Perfil de carga de trabalho enviada ao *cluster*

As políticas de economia de energia utilizadas nos experimentos foram: *performance*, *ondemand*, OOO e HLM (proposta nesse trabalho e como será chamada daqui em diante). As duas primeiras são representadas por dois *governors* padrões do Linux, definidos, respectivamente, em [CPUFreq Governors 2009] e [Pallipadi e Starikovskiy 2006]. A política chamada de OOO (*On-Off* Otimizado) implementa configuração dinâmica e DVFS. A política HLM implementa *On-Off*, DVFS, previsão de carga, ajuste dinâmico dos parâmetros do previsor, além do fator de ajuste gama. Assim, para melhorar a qualidade da previsão nos experimentos, foi inserida na política HLM uma otimização de mínimos quadrados [Wuttke 2009] aplicados sobre os coeficientes de amortização (α e β), levando em consideração a minimização do erro médio quadrado entre os dados observados e a previsão feita em uma determinada faixa de tempo. Alguns dos parâmetros de configuração para o experimento estão indicados na Tabela 3. Esses parâmetros foram designados com base no tamanho do aglomerado ou de acordo com o *overhead* provocado no computador *front-end* (no caso de intervalos ou janelas de tempo).

Tabela 3. Variáveis utilizadas nos experimentos

Descrição	**Valor**
carga máxima do *cluster*	670 req/s
deadline das requisições	4 s
γ_{max}	1,3 e 1,4
janela de estabilização	2 min
janela de decisão	5 s
janela de controle	1 s

Buscando uma melhor exposição dos resultados, iremos primeiro comparar a política HLM com os *governors performance* e *ondemand*. Depois disso iremos comparar o HLM com o OOO. Essa divisão tem como objetivo mostrar a diferença entre políticas de economia de energia que não utilizam nem DVFS nem a configuração dinâmica (*performance*), que utilizam somente DVFS discreto (*ondemand*) e que utilizam tanto DVFS

(contínuo) quanto o processo de ligar e desligar máquinas do *cluster* (HLM).

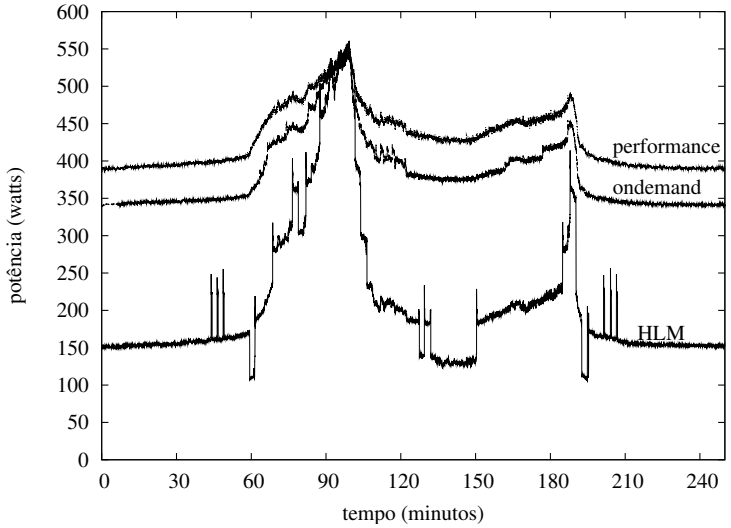

Figura 3. Potência gasta: HLM, *ondemand* e *performance*

A Figura 3 mostra a potência média gasta por segundo durante todo o tempo do experimento. Os resultados para consumo de energia registrado durante esse primeiro experimento foram: 1793 Wh para *performance*, 1600 Wh para *ondemand* e 862 Wh para o HLM. Apesar da ausência de perda de QoS nas outras políticas, o HLM obteve uma substancial economia de energia, ocorrendo somente 27 situações de QoS abaixo do nível especificado (95%). A Figura 4(a) mostra que a frequência mínima acumulada registrada no *cluster* foi de 7 GHz para a política *ondemand*. Este fato é explicado pela falta de configuração dinâmica das máquinas *workers* nessa política, ou seja, todos elas permanecem ligadas contribuindo ao menos com a sua própria frequência mínima. Já a Figura 4(b) mostra o estado de cada uma das máquinas do aglomerado durante o experimento utilizando o método HLM. Os estados de SHUT e de BOOT são estados intermediários que determinado *worker* pode assumir, significando, respectivamente, sua transição do estado ON para OFF e do estado OFF para ON.

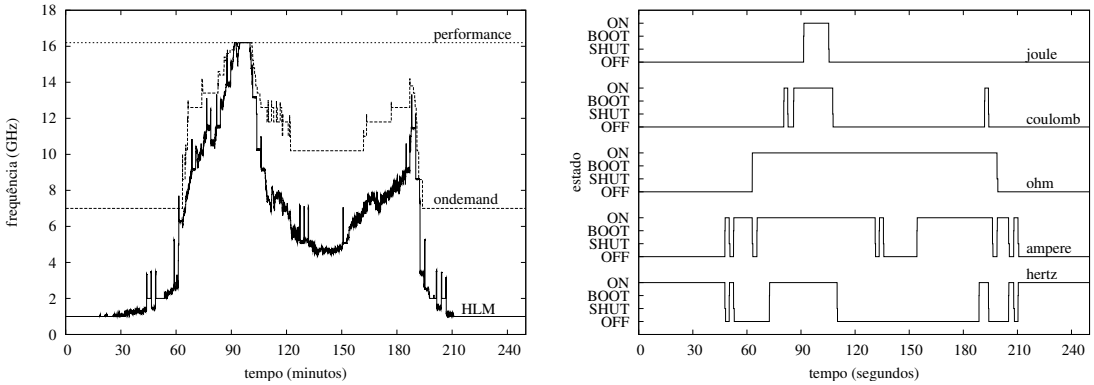

Figura 4. (a) Frequência acumulada nas políticas *performance*, *ondemand* e HLM; e, (b) estado das máquinas servidoras para HLM

Para realizar a comparação entre o método desenvolvido nesse trabalho e OOO foram usadas as seguintes métricas: nível de QoS em uma janela de controle, quantidade de energia utilizada durante todo o experimento, e número total de requisições que não terminam dentro do prazo durante o experimento. Esta última métrica difere do nível de QoS pelo fato de contabilizar requisições perdidas mesmo quando o nível de QoS está acima do especificado.

Tabela 4. Comparação de resultados: HLM *vs.* OOO

Métrica	OOO	HLM $\gamma_{max} = 1,3$	HLM $\gamma_{max} = 1,4$
requisições perdidas	9984	2096	942
ocorrências de QoS $< 95\%$	203	53	27
ocorrências de QoS $< 80\%$	63	34	14
ocorrências de QoS $< 65\%$	28	17	4
ocorrências de QoS $< 50\%$	10	8	3
energia gasta (Wh)	864	862	862

A Tabela 4 mostra os ganhos quantitativos entre as políticas HLM e OOO. Dentre as variantes do HLM, a que utiliza $\gamma_{max} = 1,4$ obteve melhores resultados em relação a QoS, devido a um melhor ajuste de configuração para um determinado instante de carga. É importante frisar que esse ajuste de gama é feito somente enquanto a QoS não atende ao especificado; assim que a nova configuração entra em vigor, gama retorna ao valor unitário, reduzindo seu impacto na energia consumida.

Quanto ao gasto energético, HLM e OOO apresentam valores que são praticamente idênticos. Diferentes execuções do *trace* levam a resultados distintos para o gasto energético, mas sempre em valores próximos ao indicado na tabela. Portanto o pior caso sempre foi utilizado em todas as políticas.5 Essa similaridade pode ser explicada pelo fato de, em vários instantes do experimento, OOO utilizar uma configuração que se mostra abaixo do suficiente para manter a qualidade de serviço dentro do padrão especificado. Dessa forma, uma menor quantidade de energia vai ser gasta pelos servidores.

Um outro efeito que pode ser destacado é o do papel do previsor de carga. Para ver a sua contribuição destacamos um trecho da execução, justamente quando a carga começa a ter um aumento significativo, em torno dos 63 minutos de execução do experimento. Esse cenário é mostrado na Figura 5. Detetando o incremento da carga, HLM vai aumentando gradualmente as frequências das máquinas ligadas, *ampere* e *ohm* (ver também Figura 4(b)). Já OOO, que não faz nenhuma previsão e procura sempre minimizar a potência gasta, manteve a frequência acumulada do *cluster* em torno de 5,1GHz. Como OOO só atua reativamente, próximo ao instante 64,3 minutos o *cluster* começa a perder QoS, e só altera as frequências de operação em torno do instante 64,5 minutos da execução. Já HLM atuou preventivamente para garantir a QoS, em detrimento do consumo energético, e manteve a QoS em 100%.

6. Conclusão

No trabalho desenvolvido nesse artigo, a utilização de uma política preditiva visando economia de energia foi aplicada em um ambiente real de um *cluster* de servidores. Empregando uma metodologia de previsão de carga baseada no Método Linear de Holt, ganhos

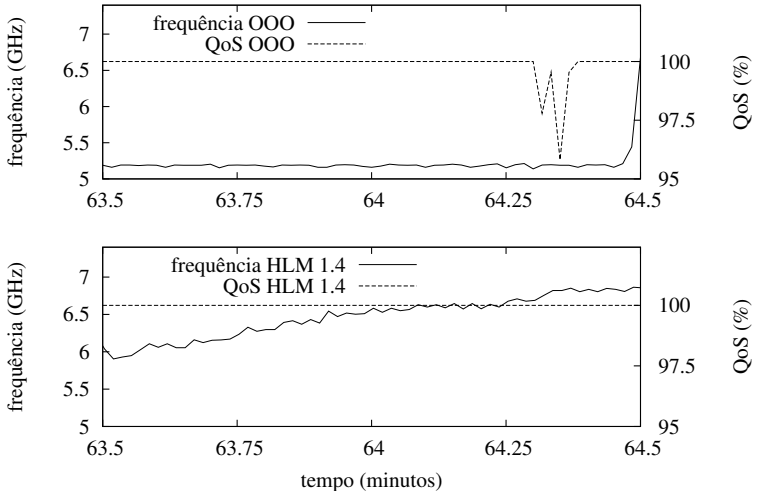

Figura 5. HLM *vs.* OOO: frequências e QoS

energéticos superiores a 51% foram obtidos quando HLM é comparado ao esquema *performance*, e de aproximadamente 46% quando comparado ao esquema *ondemand*. Foi observado que a economia de energia pode ser obtida respeitando-se contratos negociados para a qualidade de serviço. Quando comparada a um sistema otimizado (OOO), a técnica aqui apresentada mostrou que atinge os mesmos patamares de ganho energético, porém comportando-se melhor em termos de qualidade de serviço. A adoção de um método de previsão proporcionou ao sistema atuar de maneira pró-ativa, característica essencial para sistemas nos quais são exigidos altos níveis de qualidade de serviço.

Como trabalhos futuros, um primeiro será realizar experimentos para avaliar o impacto ocasionado por outras variáveis do modelo (janela de controle e $util_{max}$, por exemplo) e uma análise de sensibilidade mais ampla para os valores de γ_max. Um outro estudo a ser feito será a execução de testes com outros perfis de carga, para tentar avaliar características como sazonalidade e cargas com valores de autocorrelação mais baixos. Para tornar a metodologia exposta nesse trabalho aplicável em casos reais, quesitos de alta disponibilidade e segurança poderão ser discutidos e acrescentados em protótipos futuros. E finalmente pretendemos avaliar o impacto de outras técnicas de previsão, além dos métodos de amortecimento exponencial.

7. Agradecimentos

Os autores gostariam de agradecer às agências nacionais de pesquisa CNPq, Capes e Faperj por parcialmente financiarem esse trabalho. Um dos autores também agradece à NSF-USA.

Referências

AbouGhazaleh, N., Childers, B., Mossé, D. e Melhem, R. (2007). Near-memory caching for improved energy consumption. *IEEE Transactions on Computers*, 56(11):1441–1455.

ACPI (2008). Advanced configuration and power interface. `http://www.acpi.info/`.

Apache (2009). Módulo mod_proxy_balancer. http://httpd.apache.org/docs/2.2/mod/mod_proxy_balancer.html.

Arlitt, M. e Jin, T. (1999). Workload characterization of the 1998 Soccer World Cup web site. Relatório Técnico – Laboratórios Hewlett-Packard. http://www.hpl.hp.com/techreports/1999/HPL-1999-35R1.pdf.

Baryshnikov, Y., Coffman, E., Pierre, G., Rubenstein, D., Squillante, M. e Yimwadsana, T. (2005). Predictability of web-server traffic congestion. Em *International Workshop on Web Content Caching and Distribution*, pp. 97–103, Washington, DC, EUA.

Bertini, L., Leite, J. C. B. e Mossé, D. (2007). Statistical QoS guarantee and energy-efficiency in web server clusters. Em *19th Euromicro Conference on Real-Time Systems*, pp. 83–92, Pisa, Itália.

Bertini, L., Leite, J. C. B. e Mossé, D. (2008). Optimal dynamic configuration in web server clusters. Relatório técnico, IC-UFF.

Bertini, L., Leite, J. C. B. e Mossé, D. (2009). Generalized tardiness quantile metric: Distributed DVS for soft real-time web clusters. Em *21st Euromicro Conference on Real-Time Systems*, pp. 227–236, Dublin, Irlanda.

Bertini, L., Leite, J. C. B. e Mossé, D. (2010). Power optimization for dynamic configuration in heterogeneous web server clusters. *Journal of Systems and Software*, 83(4):585–598.

Bertozzi, D., Benini, L. e Ricco, B. (2002). Power aware network interface management for streaming multimedia. Em *IEEE Wireless Communications and Networking Conference*, volume 2, pp. 926–930, Orlando, FL, EUA.

Bohrer, P., Elnozahy, E. N., Keller, T., Kistler, M., Lefurgy, C., McDowell, C. e Rajamony, R. (2002). *Power aware computing*. Kluwer Academic Publishers.

COIN-OR (2009). Computational INfrastructure for Operations Research. http://www.coin-or.org.

CPUFreq Governors (2009). Linux Kernel Documentation. http://www.mjmwired.net/kernel/Documentation/cpu-freq/governors.txt.

Elnozahy, E. M., Kistler, M. e Rajamony, R. (2002). Energy-efficient server clusters. Em *Workshop on Power-Aware Computing Systems*, Cambridge, EUA.

EPA (2007). Report on server and data center energy efficiency. www.energystar.gov.

EPA (2009). Energy star. http://www.energystar.gov/index.cfm?c=prod_development.server_efficiency_study.

Flinn, J. e Satyanarayanan, M. (1999). Energy-aware adaptation for mobile applications. *ACM SIGOPS Operating Systems Review*, 33(5):48–63.

Ganesh, L., Weatherspoon, H., Balakrishnan, M. e Birman, K. (2007). Optimizing power consumption in large scale storage systems. Em *USENIX Workshop on Hot Topics in Operating Systems*, pp. 1–6, San Diego, CA, EUA.

GLPK (2009). Gnu linear programming kit. http://www.gnu.org/software/glpk.

Horvath, T., Abdelzaher, T., Skadron, K. e Liu, X. (2006). Dynamic voltage scaling in multitier web servers with end-to-end delay control. Em *IEEE Real-Time and Embedded Technology and Applications Symposium)*, pp. 444–458, CA, EUA.

IEA (2008). World energy outlook 2008 edition – executive summary. `http://www.iea.org/weo/2008.asp`.

Kim, B. I. e Chang, T. G. (2005). A power reduction technique based on the microscopic dynamic voltage scaling (DVS) of multimedia processors in wireless communication terminal. Em *The First IEEE and IFIP International Conference in Central Asia on Internet*, pp. 1–4, Bishkek, República do Quirguistão.

Kusic, D. (2008). *Combined Power and Performance Management of Virtualized Computing Environments Using Limited Lookahead Control*. PhD thesis, Drexel University, Philadelphia, PA, EUA.

Mosberger, D. e Jin, T. (1998). httperf: A tool for measuring web server performance. Em *Internet Server Performance Workshop*, pp. 59–67, Madison, WI, EUA.

Pallipadi, V. e Starikovskiy, A. (2006). The ondemand governor: past, present and future. Em *Linux Symposium*, pp. 223–238, Ottawa, Canadá.

Petrucci, V., Loques, O. e Mossé, D. (2009). A framework for dynamic adaptation of power-aware server clusters. Em *ACM Symposium on Applied Computing*, pp. 1034–1039, Honolulu, HI, EUA.

Pinheiro, E., Bianchini, R., Carrera, E. e Heath, T. (2001). Load balancing and unbalancing for power and performance in cluster-based systems. Relatório Técnico DCS-TR-440, Department of Computer Science, Rutgers University, EUA.

Rusu, C., Ferreira, A., Scordino, C. e Watson, A. (2006). Energy-efficient real-time heterogeneous server clusters. Em *IEEE Real-Time and Embedded Technology and Applications Symposium*, pp. 418–428. San Jose, CA, EUA.

SCIP (2009). Solving constraint integer programs. `http://scip.zib.de/`.

Spyros Makridakis, S. C. W. e Hyndman, R. J. (2002). *Forecasting: Methods and Applications*. Kluwer Academic Publishers.

Tolentino, M., Turner, J. e Cameron, K. (2009). Memory miser: Improving main memory energy efficiency in servers. *IEEE Transactions on Computers*, 58(3):336–350.

TPC (2009). Transaction Processing Performance Council. `http://www.tpc.org/`.

Wang, Y., Wang, X., Chen, M. e Zhu, X. (2008). Power-efficient response time guarantees for virtualized enterprise servers. Em *IEEE Real-Time Systems Symposium*, pp. 303–312, Barcelona, Espanha.

Wuttke, J. (2009). lmfit – A C/C++ routine for Levenberg-Marquardt minimization with wrapper for least-squares curve fitting. `http://www.messen-und-deuten.de/lmfit/`.

Yuan, W. e Nahrstedt, K. (2003). Energy-efficient soft real-time CPU scheduling for mobile multimedia systems. Em *ACM Symposium on Operating Systems Principles*, pp. 149–163, Bolton Landing, NY, EUA.