Previsão de Carga para Configuração Dinâmica de Aglomerados de Servidores *Web*

Giulio D. Bottari[1], **Julius C. B. Leite**[1]

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

{gbottari,julius}@ic.uff.br

Abstract. *Nowadays, there is a growing awareness about green computing in computer systems. A major source of energy demand in these systems are web server clusters. This paper evaluates a policy to anticipate the load and configure the number of active servers so that a satisfactory quality of service is met, while reducing energy usage. Experiments are performed on a cluster prototype to validate the employed technique.*

Resumo. *Atualmente, há uma crescente conscientização sobre computação verde em sistemas computacionais. Uma das grandes fontes de demanda energética nesses sistemas são os aglomerados de servidores web. Este trabalho avalia uma política para antecipar a carga e configurar o número de servidores ativos de forma que seja mantido uma qualidade de serviço satisfatória, enquanto se reduz o uso de energia. Experimentos são feitos em um protótipo de cluster para validar a técnica empregada.*

1. Introdução

Redes sociais, motores de busca e aplicações de *e-commerce* caracterizam uma grande parte do que se tornou hoje a Internet. Muitas dessas aplicações demandam um processamento de dados contínuo, realizado em grandes aglomerados (*clusters*) de servidores.

Ao longo dos últimos anos, uma grande preocupação vem se formando em torno da demanda energética desses aglomerados. O tema tem um ponto de vista econômico e um ângulo ambiental, devido principalmente ao custeio da energia usada nesses centros e à emissão de gases causadores do efeito estufa, respectivamente.

A demanda de trabalho num aglomerado de servidores *web* varia ao longo do dia, em função da utilização dos usuários. Por esse motivo, *designers* de *datacenters* o dimensionam para condições de tráfego intenso, como medida de segurança. Isso significa que a maior parte do tempo o aglomerado se encontra subutilizado. Sendo assim, uma prática comum é desligar servidores inativos e religá-los no futuro quando houver necessidade de mais performance, economizando energia. Tal mecanismo é conhecido na literatura como *Configuração Dinâmica de Servidores* [Bianchini e Rajamony 2004] (que será identificado por *CDS*, daqui por diante).

Trabalhos como [Pinheiro et al. 2001, Rusu et al. 2006, Petrucci et al. 2011] utilizam uma proposta de CDS complementada por variação dinâmica de frequência do processador. Ao invés de se tratar o tema de economia de energia em servidores

num espectro mais amplo, aqui será abordado mais minuciosamente a previsão e antecipação de carga para CDS.

Trabalhos mais recentes como [Santana et al. 2008] e [Santana 2010] utilizam um esquema de CDS auxiliado por previsões, obtendo melhorias no atendimento da carga. Estes trabalhos também utilizam um fator de superestimação de carga para antecipar a reconfiguração do sistema. A proposta deste trabalho é separar estas abordagens em dois métodos: Preditivo e Reativo, explicados em detalhes mais adiante, e comparar as contribuições de cada um.

Na Seção 2 será caracterizada a abordagem escolhida para CDS. Posteriormente será apresentado o previsor utilizado, bem como a calibragem de seus parâmetros na Seção 3. As Seções 4 e 5 mostrarão os *traces* e o protótipo de *cluster* empregados nos testes. Em seguida, serão abordados a metodologia e resultados dos testes realizados na Seção 6. Finalmente, serão apresentados a conclusão e trabalhos futuros, na Seção 7.

2. Configuração Dinâmica de Servidores

A CDS tem como objetivo diminuir o gasto energético do aglomerado, ao minimizar o conjunto de servidores trabalhadores ativos. Porém, esse novo conjunto deve ter a capacidade de atender ao mesmo volume de requisições que o anterior de modo a não prejudicar a qualidade de serviço.

Pensando nisso, foi desenvolvida uma política conservadora com relação ao chaveamento de servidores, ao mesmo tempo em que busca otimizar o gasto energético do *cluster*. O seu funcionamento está resumido no Algoritmo 1, que deverá ser executado em *loop* com período configurável (nos experimentos foi usado um período de 20 segundos).

Os servidores disponíveis são colocados em uma lista, que deve ser ordenada por algum critério (e.g. eficiência energética). Um critério simples, adotado nesse trabalho, é ordenar as máquinas segundo a sua vazão máxima (ver Tabela 2). Ligando primeiro os servidores mais robustos evita-se que outros menos poderosos, que geralmente oferecem uma baixa taxa de performance por Watt, sejam escolhidos.

Uma convenção adotada foi que as máquinas do início da lista têm prioridade para serem ligadas, enquanto que as do fim têm prioridade de desligamento. Sempre que houver a necessidade de mais performance no *cluster*, um servidor da lista é ligado. Caso contrário, procura-se diminuir o número dos servidores ativos, dado que a reconfiguração não torne o aglomerado subdimensionado para a carga. A lista é fixa e igual para todos os experimentos. O objetivo deste trabalho é apenas comparar os métodos Reativo e Preditivo, para tanto não é necessário trabalhar com uma lista de servidores ótima em relação à eficiência energética.

O chaveamento de servidores é feito em função da carga (*load*), que é a média simples do número de requisições no intervalo passado. Além disso, pode-se utilizar o parâmetro $\gamma \in [0, 1]$ (linhas 1, 6 e 17 do Algoritmo 1) para *antecipar* a reconfiguração do aglomerado. A utilização desse recurso será chamado de método Reativo.

A necessidade de se antecipar reconfigurações em sistemas de CDS surge ao

Algoritmo 1 Atuação da Configuração Dinâmica

```
1: if load > throughput(currConfig) · γ then
2:    {Obter o servidor mais eficiente que, uma vez ligado, será capaz de suprir a carga.}
3:    candidates ← (serverPoll − currentConfig) {máquinas que ainda podem ser ligadas}
4:    for server in candidates do
5:       newConfig ← (currentConfig ∪ server)
6:       if throughput(newConfig) ≥ load · γ then
7:          return newConfig
8:       end if
9:    end for
10:   {Nenhum servidor é capaz de suprir load, então o mais robusto será ativado.}
11:   return currentConfig ∪ mostRobustServer(candidates)
12: else
13:   {Tenta-se reduzir o número de máquinas que estão ligadas.}
14:   candidates ← reverse(currentConfig) {máquinas que podem ser desligadas, na ordem inversa}
15:   for server in candidates do
16:      newConfig ← (currentConfig − server)
17:      if throughput(newConfig) · γ ≥ load then
18:         return newConfig
19:      end if
20:   end for
21:   return currentConfig
22: end if
```

se considerar o tempo de ligamento destes servidores, operação que leva de 6 à 10 segundos. Quando não se consegue ativar um servidor a tempo de tratar um aumento de carga, tende-se a saturar o sistema.

Como pode-se observar, o Algoritmo 1 irá ligar ou desligar no máximo uma máquina de cada vez. Pretende-se evitar que máquinas sejam ligadas quando há grandes variações de carga que se mantêm por um curto período. Nesse caso, quando o nível de requisições aumenta abruptamente, o *overhead* energético do ligamento de um servidor não compensa o benefício de ligá-lo. Um efeito análogo também ocorre quando a carga decresce bruscamente, nesse caso em relação a possibilidade de perda de QoS.

3. Previsor Holt

O Método Linear de Holt, utiliza dois componentes s_t e b_t, que representam o valor suavizado da carga e a estimativa da tendência para o instante t, respectivamente. Por ser capaz de detectar tendências nas séries temporais, julga-se que este previsor é adequado para abordar o problema.

O Sistema 1 mostra o conjunto de equações deste modelo. O Método de Holt possui dois parâmetros que controlam a suavização: α e $\beta \in [0,1]$. A variável F_{t+m} dá a previsão para m instantes no futuro, em função da carga atual.

$$
\begin{aligned}
s_1 &= load_0 \\
b_1 &= load_1 - load_0 \\
s_t &= \alpha \cdot load_t + (1-\alpha) \cdot (s_{t-1} + b_{t-1}) \\
b_t &= \beta \cdot (s_t - s_{t-1}) + (1-\beta) \cdot b_{t-1} \\
F_{t+m} &= s_t + m \cdot b_t
\end{aligned}
\tag{1}
$$

O Algoritmo 1 considera a carga atual para realizar a configuração do sistema, como no método Reativo. No caso do método Preditivo, esse valor serve de entrada para o método de Holt, retornando o valor da carga prevista ($load_{pred}$). Porém, não é vantajoso utilizar o valor de $load_{pred}$ para configurar diretamente o aglomerado. Caso a previsão aponte para uma queda do volume de requisições, não se deve correr o risco de afetar a QoS desligando servidores no momento atual, sendo que a demanda ainda não diminuiu. Sendo assim, o *cluster* é configurado utilizando o maior valor entre a carga prevista e a atual, i.e. $max(load_{pred}, load_{curr})$. Em testes preliminares foi verificado que esse método é melhor ou equivalente ao uso de $load_{pred}$ para configurar o *cluster*.

3.1. Otimização de parâmetros

Os valores dos parâmetros do previsor Holt interferem drasticamente no resultado das previsões. Neste trabalho não serão abordados métodos de variação dinâmica para otimizar esses parâmetros. Ao invés disso, serão calculados valores fixos utilizados por toda a duração de um experimento.

Uma técnica apropriada para calibrar estes parâmetros é a minimização de alguma métrica de erro entre a previsão e o dado observado, como MSE (*Mean Square Error*) ou MAE (*Mean Absolute Error*) [Spyros Makridakis e Hyndman 2002]. Além desses critérios clássicos, elaborou-se um terceiro chamado *Híbrido*. Este tem o intuito de produzir parâmetros que evitem a subestimação do previsor em relação à carga. Combinando as outras duas métricas, procura-se penalizar mais severamente uma configuração insuficiente do que uma superdimensionada. Um resumo dos critérios utilizados pode ser visto na Tabela 1. Nela, A_t denomina o valor observado, enquanto que F_t indica o valor previsto no instante t.

Tabela 1. Critérios de minimização

Erro	Fórmula
MSE	$\frac{1}{n} \cdot \sum_{t=1}^{n} (A_t - F_t)^2$
MAE	$\frac{1}{n} \cdot \sum_{t=1}^{n} \lvert A_t - F_t \rvert$
Híbrido	$\frac{1}{n} \cdot \sum_{t=1}^{n} \begin{cases} (A_t - F_t)^2 & \text{se } A_t < F_t \\ \lvert A_t - F_t \rvert & \text{se } A_t \geq F_t \end{cases}$

Para realizar a minimização dos erros, foi utilizado um método simples. Gerou-se todas as combinações possíveis de parâmetros α e β no intervalo [0,1] com um passo de 10^{-2}. Para cada combinação, calculou-se o erro entre a estimativa

e o resultado real para todo o *trace*. Ao final, foram escolhidos os parâmetros que resultam no menor erro em questão.

As principais vantagens desse método são a simplicidade de implementação e estabilidade (o método sempre convergirá). Em contrapartida, o processo pode se tornar custoso ao levar em conta o tamanho do *trace* e número de parâmetros a serem pesquisados (no caso, apenas dois).

4. *Traces*

Para obter um perfil de carga realístico foram utilizados *traces* de requisições feitas a servidores reais, como mostra a Figura 1. O primeiro deles (*Linear*) foi criado sinteticamente para ser fácil de ser previsto usando o método de Holt. Dessa forma, pode-se ter uma base de comparação sobre o potencial desse previsor. Em seguida, temos um trecho da demanda submetida aos servidores durante a Copa do Mundo de Futebol de 1998 (*WC 98*) e outros três realizados aos servidores da NASA.

Os *logs* nos quais os *traces* foram baseados estão disponíveis em [LBNL 2011]. Para construir os *traces*, foi feita a contagem de requisições que foram atendidas a cada segundo nestes *logs*, e em seguida estes valores são dimensionados para a capacidade máxima do *cluster* de testes.

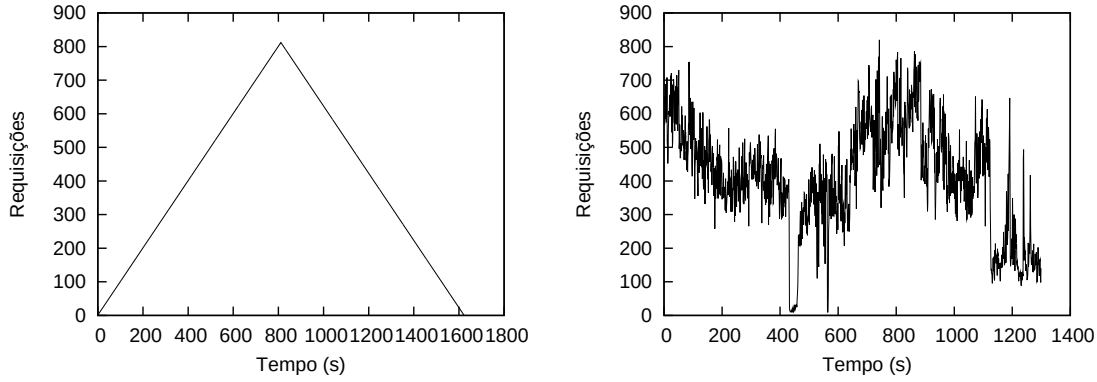
5. Ambiente de testes

O ambiente de testes consiste de um aglomerado de servidores trabalhadores (*workers*) heterogêneos rodando o servidor *web* Apache versão 2.2.10, sob o SO Gentoo Linux. Um gerador de carga envia requisições durante o experimento para um *frontend*, que distribui a carga entre os trabalhadores através do módulo Apache `mod_proxy_balancer`. A carga é balanceada de acordo com a vazão de cada servidor (Tabela 2), buscando-se obter uma utilização de CPU uniforme entre os mesmos. A Figura 2 ilustra essa topologia.

Durante o experimento, cada servidor pode ser chaveado entre dois estados: *ativo* e *suspenso*. No primeiro estado o servidor se encontra pronto para atender requisições, enquanto que no segundo a máquina salva seu contexto em memória *RAM* e entra em um modo econômico de energia (chamado de *Suspend-to-RAM*), que não possibilita o atendimento de requisições. Para transitar entre os estados *suspenso* e *ativo* é utilizado o recurso *Wake-on-LAN*, disponível na placa de rede.

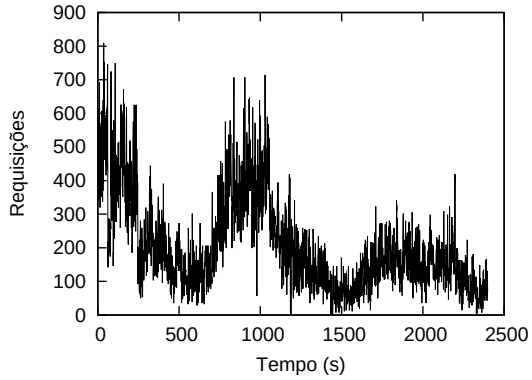
O tráfego é gerado nesse sistema através da ferramenta de *benchmark* `httperf` [Mosberger e Jin 1998]. A carga consiste de apenas um tipo de requisição, intensiva em CPU, implementada através de um *script* PHP que calcula números primos. Além disso, ao longo do experimento as máquinas utilizam o *governor* `performance` do Linux, mantendo o seu *clock* máximo durante todo o tempo. Quando o sistema não se encontra saturado, cada requisição demora cerca de 6 ms para ser atendida.

A Tabela 2 mostra a medida de performance utilizada para os servidores do experimento na sua frequência de operação máxima: a vazão (*throughput*) de requisições. Além disso, foi medida a potência dos servidores no estado ocioso e ocupado. No estado ocioso nenhuma requisição está sendo atendida, enquanto que no segundo, o processador se encontra cerca de 100% ocupado.

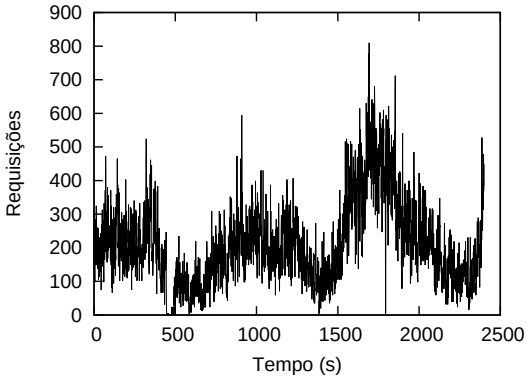


(a) Linear

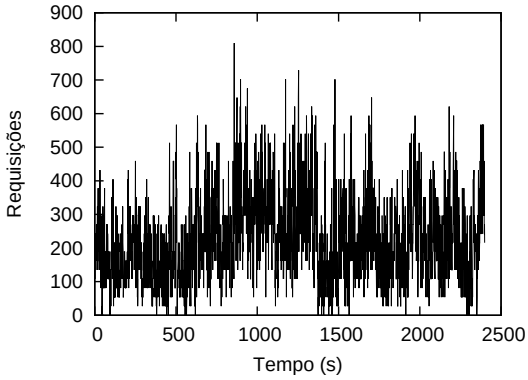
(b) WC 98



(c) NASA 1



(d) NASA 2



(e) NASA 3

Figura 1. Traces do experimento

6. Experimentos

Os experimentos realizados foram divididos em duas etapas: simulações e experimentos no *cluster*. Na etapa de simulação procura-se observar o impacto de diversos parâmetros no sistema e os critérios de escolhas de parâmetros de previsão. A etapa seguinte irá analisar o comportamento no ambiente real das propostas preditiva e reativa.

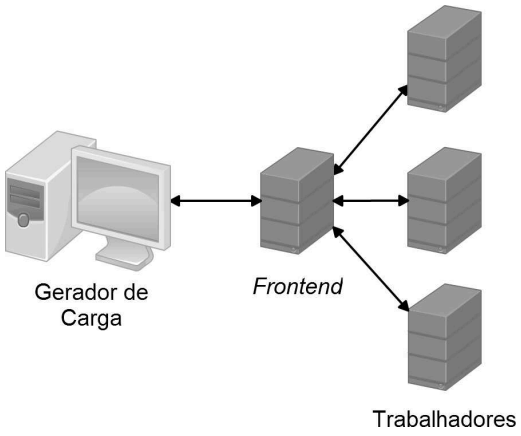


Figura 2. Topologia

Tabela 2. Características dos servidores utilizados

Nome	Arquitetura (AMD)	Potência ociosa (W)	Potência ocupada (W)	Vazão (req/s)
ampere	Athlon 64 X2 3800+	72,7	109,8	199,6
coulomb	Athlon 64 3800+	75,2	107,7	122,3
hertz	Athlon 64 3800+	71,6	103,2	122,8
joule	Athlon 64 3500+	80,0	110,6	111,4
ohm	Athlon 64 X2 5000+	76,9	140,1	255,2

6.1. Simulações

Para comparar as políticas Reativa e Preditiva, foi montada uma série de experimentos de simulação com os *traces* da Seção 4. A título de comparação, será utilizado também um previsor "visionário", que conhece exatamente o valor da carga em qualquer instante de tempo. As simulações assumem que o tempo de ligamento ou desligamento de um servidor é instantâneo.

6.1.1. Efeito das previsões

Nesta Seção será investigada a capacidade do previsor de reduzir o número de subestimações: ocasiões em que a configuração apontada não é suficiente para suprir a demanda. Também será levado em conta o caso contrário, quando a configuração é superdimensionada. Essas métricas estão relacionadas com a performance da aplicação e com o gasto energético, respectivamente. Em todos os experimentos será usado $\gamma = 1$.

Na Figura 3 é possível observar que o previsor Visionário superestima a carga algumas vezes. Isto ocorre porque o Algoritmo 1 só desliga uma máquina por vez, não importa quão baixa esteja a carga. Os experimentos começam com todos os servidores ligados, fazendo com que esse efeito ocorra apenas no início.

Excluindo o Visionário, o método Preditivo com critério Híbrido apresenta o menor índice de subestimações, mas às custas de um alto número de superestimações em todos os *traces*. Pode-se esperar em experimentos reais no *cluster* que esse

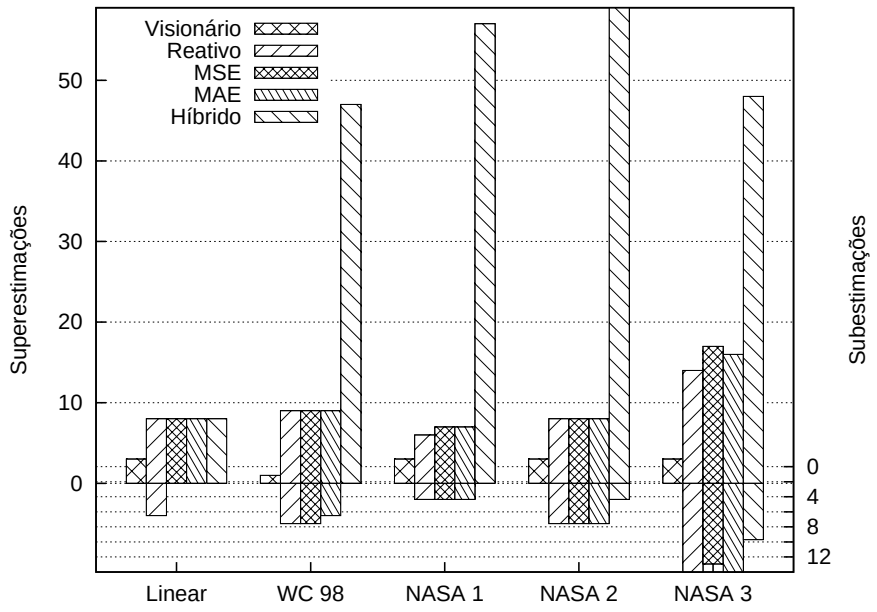


Figura 3. Efeito das previsões

comportamento resulte em um melhor nível de QoS, acompanhado de um gasto maior de potência.

Os demais métodos possuem resultados equivalentes, sem conseguir reduzir muito o número de subestimações em comparação com o método Reativo, exceto no caso do *trace* Linear, onde a linearidade do perfil de carga propicia o bom funcionamento do previsor Holt.

6.1.2. Reduzindo subestimações

Na Seção anterior notou-se que apenas o método Preditivo Híbrido é capaz de oferecer alguma melhoria significativa sobre o método Reativo, no caso do subdimensionamento. Por esse motivo, a análise seguirá apenas com esse critério.

Nos experimentos que seguem, foi variado o parâmetro γ a fim de se obter um resultado com zero subestimações com o método Reativo. Na Figura 4, o número que acompanha cada barra é o valor de γ utilizado.

Conforme a Figura 4, o método Reativo é capaz de obter zero subestimações em todos os perfis de carga, diferentemente do Preditivo Híbrido. De fato, é possível obter sempre um valor de γ tal que número de subestimações é zero. Dado que o *cluster* está dimensionado corretamente, $\gamma = 0$ significa que todos os servidores permanecerão ligados, sem causar nenhuma subestimação. Além disso, este método consegue menos superestimações para um valor igual ou menor de subestimações nas simulações em todos os casos, a não ser pelos *traces* Linear e WC 98.

A Figura 5 mostra as (re)configurações feitas ao longo de dois *traces*. Nela, as linha de Reativo e Preditivo correspondem a vazão máxima do *cluster* naquele momento. Note que o método Preditivo é mais estável, pois requer menos reconfi-

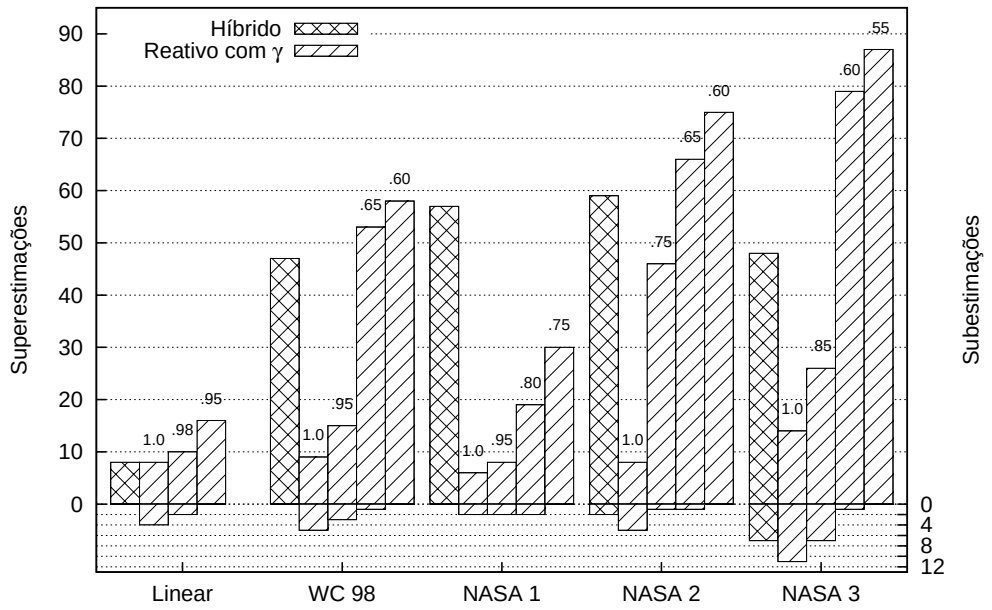
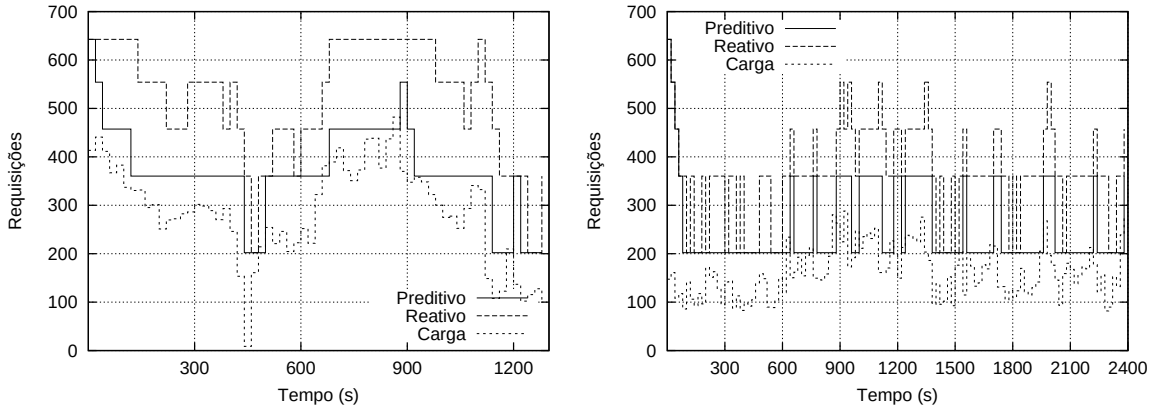


Figura 4. Reduzindo subestimações

gurações. Por sua vez, o método Reativo é mais robusto, pois acompanha melhor o perfil da carga.



(a) WC 98 (b) NASA 3

Figura 5. Configurações reativas e preditivas

6.2. Experimentos no protótipo de *cluster*

As simulações cumpriram seu papel de prover uma análise inicial do sistema de CDS desenvolvido e para calibrar uma série de parâmetros, como intervalo de atuação, α, β e γ. Nesta seção alguns experimentos selecionados da Seção 6.1 serão rodados no ambiente de testes.

Ao invés de medir o número de configurações subestimadas ou superestimadas, serão medidas QoS e potência consumida pelo *cluster*. A QoS instantânea é calculada a cada segundo, e foi definida em função das requisições como $1 - \frac{perdidas}{atendidas}$ (note que requisições *perdidas* se referem àquelas atendidas fora do prazo). Obtêm-se

os valores de requisições atendidas, QoS e tempo de atendimento, através da análise do arquivo `acess_log` do Apache.

O valor para o *deadline* dessas requisições foi ajustado para 20 vezes o valor do seu tempo de resposta médio quando o aglomerado não se encontra saturado (no caso, 120 ms). Essa é uma recomendação do TPC-W [TPC 2011] para requisições com *deadline* curto.

Experimentos realizados em outros trabalhos [Li et al. 2009] demonstram que pode-se aproximar por uma relação linear a utilização do processador e a potência gasta por um servidor quando a carga é intensiva em CPU. Por isso, os valores de potência apresentados foram aproximados através das medidas de potência ociosa e ocupada (ver Tabela 2) em função da utilização dos processadores.

6.2.1. Comparação entre parâmetros otimizados

Na Seção 6.1, foi argumentado a favor do método Preditivo Híbrido em relação aos outros métodos preditivos. Resta saber se resultados semelhantes podem ser obtidos em experimentos no *cluster*. A Figura 6.2.1 mostra os resultados de potência e QoS para o método Preditivo com todos os critérios da Tabela 1. Repare que o *trace* Linear não foi mostrado, pois os parâmetros resultantes dos três critérios são os mesmos ($\alpha = \beta = 1$). No eixo vertical esquerdo temos o QoS médio, enquanto que no direito está representada a potência média.

Em alguns casos, os experimentos reais não acompanharam os resultados da Seção 6.1. Por exemplo, os valores de QoS nos *traces* NASA 2 e 3 são diferentes nos critérios MSE e MAE apesar de obterem o mesmo número de superestimações em alguns casos. No entanto, na Figura 3 eles apresentam o mesmo número de subestimações. Repare que a métrica utilizada era baseada na contagem de subestimações e não no quanto foi subestimado nem por quanto tempo, possibilitando essa diferença. Esse efeito mostra que experimentos reais são muito importantes para validar trabalhos como este.

Os três primeiros *traces* indicados na Figura 6.2.1 mostram uma melhor QoS com a utilização do método Híbrido. Porém, tal melhoria vem acompanhada de um maior gasto energético, como esperado. Nos experimentos com o *trace* NASA 3, o critério MAE apresenta vantagem em QoS e potência consumida em relação aos demais.

6.2.2. Comparação entre Reativo e Preditivo

Tendo em vista que os critérios MAE e Híbrido tiveram melhor desempenho na seção anterior, serão comparados agora com o método Reativo. Em termos de QoS, o método Reativo é superior ou fica próximo do Preditivo Híbrido em todos os casos representados na Figura 6.2.2. No segundo caso, a potência consumida pelo método Preditivo Híbrido não justifica a sua pequena vantagem em QoS.

O método Preditivo MAE se coloca numa posição interessante nos três pri-

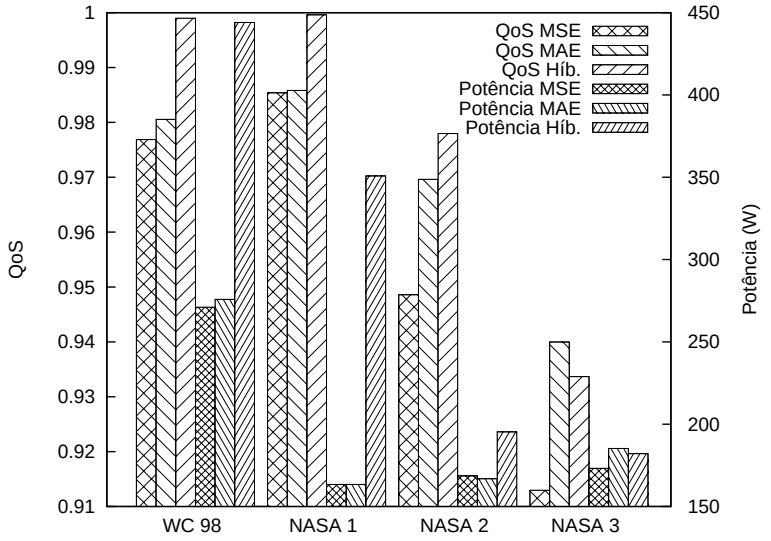


Figura 6. Experimentos no cluster: variações do método Preditivo

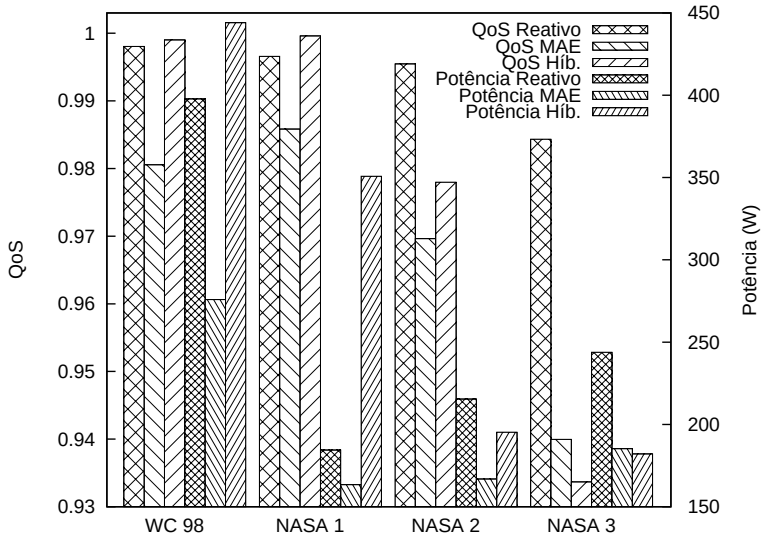


Figura 7. Experimentos no cluster: Preditivo vs Reativo

meiros *traces*, oferecendo um bom compromisso entre potência e QoS. Em compensação, quando uma garantia de QoS mais elevada for uma característica mais valiosa em aplicações *web*, o método Reativo é mais vantajoso pela sua flexibilidade.

7. Conclusão e trabalhos futuros

Este trabalho analisou uma política simples e eficaz para CDS. Foi comparado o uso de duas abordagens que visam evitar a saturação do aglomerado: a reativa e a preditiva. Foi apresentado o previsor Holt e a otimização de seus parâmetros através da minimização de três critérios: MSE, MAE e Híbrido. Através dos experimentos conduzidos, estudou-se o impacto de políticas simples em QoS e potência consumida.

Em face dos resultados obtidos, pode-se concluir que a abordagem mais recomendada é a Reativa, com uso do parâmetro γ. Em testes adicionais (não mostrados

aqui), foi observado que abordagem preditiva com o uso de γ é equivalente ou inferior à Reativa. O uso deste parâmetro permite um controle mais simples e intuitivo sobre a superestimação de carga e uma flexibilidade superior ao prover níveis de QoS ajustáveis. Em contrapartida, o uso da política Preditiva por si só não é capaz de realizar o mesmo. Por exemplo, na Figura 6.2.1, não é possível obter um QoS de 95% em NASA 3, caso isso seja um requerimento da aplicação.

O passo seguinte desta pesquisa será o controle de QoS através de um sistema adaptativo. Nesse sistema os parâmetros α e β seriam minimizados periodicamente enquanto γ seria manipulado para se ajustar ao alvo de QoS fixado. Espera-se obter um controle simples que possa antecipar a carga de maneira eficaz. Além disso, futuramente será importante incorporar políticas de variação de frequência para otimizar o gasto energético e obter um controle fino de QoS.

Em trabalhos futuros serão analisados outros previsores como o Método de Holt-Winters, capaz de detectar sazonalidade em séries temporais. Esta é uma característica presente em alguns *traces* que não foi explorada neste trabalho. Além disso, neste trabalho as previsões têm um período curto de 20 segundos. É necessário analisar períodos maiores, onde previsões de longo prazo podem ser mais vantajosas que o esquema reativo.

Referências

Bianchini, R. e Rajamony, R. (2004). Power and energy management for server systems. *IEEE Computer*, 37(11):68–74.

LBNL (2011). The Internet traffic archive. `http://ita.ee.lbl.gov/index.html`.

Li, L., RuiXiong, T., Bo, Y. e ZhiGuo, G. (2009). A model of web server's performance-power relationship. In *International Conference on Communication Software and Networks*, pp. 260–264, Macau, China. IEEE Computer Society.

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

Petrucci, V., Carrera, E. E. V., Loques, O., and J.C.B. Leite, Mossé, D., Leite, J. e Mossé, D. (2011). Optimized management of power and performance for virtualized heterogeneous server clusters. In *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid*, pp. 23–32.

Pinheiro, E., Bianchini, R., Carrera, E. V. 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. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pp. 418–428. San Jose, CA, EUA.

Santana, C. (2010). Previsão de carga em aglomerado de servidores web aplicada a economia de energia. Dissertação de Mestrado, Universidade Federal Fluminense, Niterói, RJ, Brasil.

Santana, C., Bertini, L., Leite, J. C. B. e Mossé, D. (2008). Applying forecasting to interval based DVS. In *Brazilian Workshop on Real-Time Systems*, pp. 13–20, Rio de Janeiro, RJ, Brasil.

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

TPC (2011). Transaction Processing Performance Council. `http://www.tpc.org/`.