MPM:
Middleware para Gerência de Energia em Clusters Web

Dorgival Guedes , Wagner Meira Jr. , Diêgo Nogueira , Rodrigo Pereira

[1]Laboratório e-Speed
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Av. Antônio Carlos 6627 – 31270-010 – Belo Horizonte, MG

{dorgival,meira,diego,rpereira}@dcc.ufmg.br

Resumo. *Devido ao crescimento da World Wide Web, sistemas populares demandam o uso de servidores capazes de suportar demandas de pico que podem exceder em ordens de magnitude o tráfego médio de momentos de carga mais leve. Esse excesso em termos de capacidade de processamento se reflete em um consumo extra de energia. O desenvolvimento de aplicações capazes de gerenciar esse consumo requer formas eficientes de isolar os detalhes do hardware das demandas da aplicação. Este trabalho apresenta um middleware desenvolvido para esse fim que usa técnicas da teoria de controle para relacionar demandas de desempenho com o controle de energia. Para avaliar a eficácia da solução verificamos o seu funcionamento em um* cluster *de três servidores Web sob uma carga típica. A utilização do middleware permitiu reduzir em 27% o consumo de energia, sem afetar significativamente a latência média e a taxa de serviço.*

Abstract. *Considering the fast growth of the World Wide Web, servers for popular sites are designed to handle peak workload levels that may exceed by orders of magnitude average levels during moments of lighter load. This excess processing capacity leads to an extra energy comsumption. The development of applications capable of handling energy consumption levels require efficient solutions to isolate application demands, in terms of latency and throughput, from hardware details. This paper presents a middleware solution designed to achieve that. It uses control theory techniques to associate demands for satisfactory performance with energy saving goals. To verify the efficacy of the proposed solution we applied it to a three-server Web cluster under typical workloads. The use of the middleware produced energy savings of 27% without significant impacts on the average latency and service rates.*

1. Introdução

Com o crescimento do trafego Web, servidores de sites populares precisam ser implementados para suportar demandas em horários de pico que podem exceder em ordens de magnitude o tráfego médio de momentos carga mais leve. Esse excesso em termos de capacidade de processamento se reflete em um consumo extra de energia. O desenvolvimento de aplicações capazes de gerenciar a energia utilizada requer formas eficientes

de isolar os detalhes do hardware das demandas da aplicação em termos de desempenho e qualidade de serviço. Este trabalho apresenta um middleware desenvolvido para esse fim e utiliza técnicas da teoria de controle de sistemas para relacionar demandas de desempenho satisfatório com o objetivo de economia de energia.

Aplicações Web são caracterizadas por uma alta variação de carga, com valor máximos e médios de carga que podem diferir por ordens de magnitude. Isso significa que servidores projetados para lidarem com o pico de carga de um sítio Web terão uma capacidade de processamento muito além daquela necessária para tratar requisições durante a maior parte do tempo, quando a carga pode ser bem mais baixa. Essa capacidade computacional excessiva acaba representando um consumo de energia acima do necessário naqueles momentos [Chase et al., 2001].

Tendo em mente as preocupações atuais com conservação de energia, é altamente desejável que sistemas de grande porte sejam capazes de controlar seu consumo como uma estratégia para reduzir custos e aproveitar melhor os recursos do hardware. A observação de que o poder computacional excessivo continua consumindo energia em momentos de carga baixa indica um caminho para isso.

Se considerarmos que grandes servidores são hoje implementados como clusters de máquinas, é razoável considerar que durante períodos de carga mais baixa apenas uma fração das máquinas seria necessária para atender às requisições. Se pudéssemos reduzir o número de máquinas ligadas até aquela fração, o consumo de energia seria reduzido sem grande impacto sobre o desempenho do sistema [Bohrer et al., 2002]. Além da redução obtida pelo desligamento de máquinas, outras opções incluem a redução da velocidade da CPU e a escolha de políticas de escalonamento especiais, por exemplo [Flinn and Satyanarayanan, 1999, Weiser et al., 1994]. Tais recursos já estão disponíveis em máquinas portáteis e começam a ser encontrados também em sistemas desktop e servidores comerciais.

Dois desafios que precisam ser enfrentados nesse caso são a identificação das opções de controle de consumo oferecidas pelo hardware e a definição da forma de atuação sobre o mesmo. Devido à grande variedade de dispositivos e formas de economia de energia, a tarefa de desenvolver um sistema com suporte a várias soluções seria por demais onerosa. Determinar como o sistema deve atuar sobre os dispositivos a fim de obter o menor consumo possível mantendo um desempenho aceitável torna o problema ainda mais complexo. Atingir esse objetivo requer conhecimento específico sobre o hardware e sobre o comportamento do sistema. Com essas questões em mente é que decidimos implementar o middleware para gerência de energia aqui proposto, denominado MPM (*middleware for power management*).

Esse middleware garante isolamento entre as preocupações da aplicação e os detalhes do sistema. Os algoritmos e heurísticas utilizados para decidir quando mudar o perfil de consumo são implementados no núcleo do MPM, enquanto o desenvolvedor de aplicativos utiliza apenas uma interface de programação de alto nível através da qual deve expressar as demandas do sistema em termos de desempenho e níveis de serviço. O núcleo utiliza técnicas da teoria de controle para relacionar as demandas da aplicação com as restrições operacionais em termos de economia de energia. O middleware define pontos de controle que a aplicação pode usar para configurar o sistema e por onde o núcleo

pode notificá-la de alterações iminentes no perfil operacional do sistema. Dessa forma, o programador pode definir como a aplicação pode reagir e se preparar para uma alteração do ambiente.

Dessa forma, o middleware proposto simplifica o desenvolvimento de soluções economicamente eficientes e a utilização de sistemas com diferentes soluções para gerência de energia. Isso torna o MPM uma ferramenta importante no suporte ao desenvolvimento de serviços sensíveis a questões de economia de energia com alto grau de portabilidade.

Neste artigo apresentamos nas duas próximas seções os princípios internos de operação do middleware e sua arquitetura funcional, seguidas por um estudo de caso a partir da implementação disponível. Após os resultados desse estudo, apresentamos nossas conclusões e discutimos trabalhos futuros.

2. Aplicações Internet sensíveis ao consumo de energia

Como mencionado anteriormente, o desenvolvimento de serviços Web capazes de gerenciar seu consumo de energia não é uma tarefa trivial. Qualquer que seja a solução escolhida, ela deve ser capaz de controlar o consumo de energia sem prejudicar o desempenho percebido pelos usuários.

Em termos dos dispositivos, o controle do consumo de energia pode ser conseguido de várias formas. Em linhas gerais, maior desempenho implica em maior consumo de energia, mas a relação não é linear na maioria dos casos. Alguns processadores permitem a redução da freqüência de relógio; alguns trabalhos consideram o chaveamento entre discos de alto desempenho e discos mais lentos, mas de menor consumo, ou até mesmo seu desligamento temporário [Douglis and Krishnan, 1995]; outros, consideram o desligamento de recursos de *swap* da memória virtual para economizar discos, ao custo do espaço em memória disponível [Lebeck et al., 2000]. Finalmente, em clusters de alto desempenho, uma forma eficiente de se reduzir o consumo de energia é desligar parte das máquinas durante períodos de carga baixa[Bohrer et al., 2002].

Quando consideramos aplicações Web, seu desempenho é normalmente medido em termos do número de requisições atendidas por unidade de tempo (*throughput*), ou do tempo médio para atendimento de cada requisição (latência). No caso do *throughput*, como o número de requisições atendidas depende do de requisições recebidas, é comum considerarmos o *throughput* relativo, isto é, a razão entre as requisições atendidas e o total de requisições recebidas em um certo intervalo de tempo. Ações para reduzir o consumo de um sistema como as descritas anteriormente tendem a ter um impacto negativo sobre essas medidas. Isso se deve ao fato de que à medida que reduzimos o consumo de energia, a capacidade de processamento também é reduzida. Um relógio mais lento obviamente tem impacto sobre o desempenho, assim como a utilização de discos mais lentos. Se economizamos energia desligando máquinas pouco utilizadas em um cluster, o resultado final é um cluster menor, que pode ter uma capacidade de atendimento de requisições reduzida e ser sujeito a cargas maiores por máquina, aumentando o tempo de resposta [Pinheiro et al., 2002]. Um dos problemas mais difíceis do middleware proposto é como traduzir requisitos de desempenho das aplicações em limitadores para as decisões de economia de energia e vice-versa.

A fim de tornar a gerência de energia mais simples para os desenvolvedores de aplicações é importante permitir que as demandas da aplicação sejam expressas em termos das métricas apropriadas. Não faz sentido exigir que desenvolvedores definam que uma aplicação seja capaz de lidar com reduções de energia de 70%, por exemplo. A forma correta para isso é permitir que as aplicações indiquem sua intenção de tolerar reduções de consumo de energia desde que o tempo de resposta não ultrapasse um certo limiar para, por exemplo, 90% das requisições.

Assim sendo, o problema de gerência de energia pode ser expresso como "encontrar a melhor configuração em termos de economia de energia que ainda garanta à aplicação um desempenho dentro de limites aceitáveis". Esse objetivo deve ser atingido controlando os recursos de hardware (freqüência de relógio, tipo de disco, uso de memória, número de máquinas ativas, entre outros) a fim de minimizar o consumo de energia ao mesmo tempo em que certos níveis de desempenho são garantidos. O problema é que não há uma relação clara, simples e direta entre configurações de recursos que garantam uma certa economia de energia e o desempenho de cada aplicação. Essa relação é normalmente complexa e deve ser tratada no contexto do middleware, de forma a evitar que desenvolvedores de aplicações tenham que se preocupar com os detalhes.

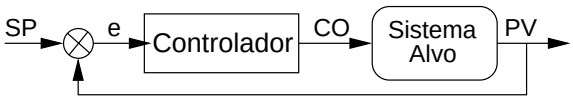

Figura 1: Controle com realimentação

O desenvolvedor deverá determinar os níveis de desempenho aceitáveis para o seu sistema (em termos de latência/*throughput*) e MPM encontrará a configuração com o menor consumo de energia que garanta tais níveis. A solução adotada no MPM é considerar um cluster Web como um sistema de controle com realimentação como ilustrado na Figura 1. O comportamento do sistema de controle pode ser descrito da seguinte maneira: o usuário define o comportamento esperado para o sistema em termos do valor desejado para uma grandeza mensurável (esse valor é denominado *setpoint*, SP) — por exemplo, a latência máxima aceitável; o sistema de controle (ou controlador) mede o valor real da variável escolhida (denominado variável do processo , PV) e calcula o erro, $e = SP - PV$. Com base nesse erro, o controlador deve calcular o valor de atuação sobre o sistema (saída do controlador, CO) que será aplicado a fim de tentar reduzir o erro, aproximando PV de SP [Franklin et al., 1997].

Essa solução é utilizada na teoria de controle quando a relação entre a variável de processo (PV) e a grandeza de atuação sobre o sistema (CO) não é clara. Nesse caso, o comportamento interno do controlador visa ajustar a entrada dinamicamente, em função da realimentação oferecida pelo erro, até que a saída desejada seja alcançada. A forma tradicional de se conseguir esse comportamento é através do que é chamado um controlador Proporcional-Integral-Derivativo (PID). Nesse caso, a saída do controlador e o erro calculado são relacionados pela equação:

$$CO = K_p e + K_i \int_0^t e.dt + K_d \frac{de}{dt}$$

As constantes PID, K_p, K_i e K_d definem o peso das três componentes do erro e

devem ser determinadas por um processo de sintonia realizado na instalação do sistema.

No MPM o objetivo é reduzir o consumo de energia tanto quanto possível, mantendo a qualidade do serviço dentro de níveis aceitáveis. Nesse caso, o erro será a diferença entre a latência (ou *throughput*) medida e o limite estabelecido. Se o valor medido é melhor que o limite estabelecido, isso indica que o sistema opera com recursos em excesso e o controlador pode decidir pela redução do consumo, reduzindo algum recurso (reduzindo a freqüência de algumas CPUs, desligando alguns dos servidores). Por outro lado, se o valor medido está pior do que limite estipulado o controlador deve adicionar recursos ao sistema a fim de melhorar o desempenho.

A decisão de qual ação tomar (alterar a freqüência de algumas CPUs ou o número de máquinas ligadas, por exemplo) depende da arquitetura de cada cluster. Quando é mais útil desacelerar cinco CPUs e quando é melhor desligar uma máquina, por exemplo? Para isso MPM deve considerar um grafo onde os nós representam todas as combinações possíveis de estados das máquinas de um cluster e as arestas representam transições possíveis, sendo rotuladas com o aumento/diminuição de consumo obtida pela mudança. A partir desse grafo é preciso determinar um caminhamento que garanta um aumento monotônico do nível de consumo de energia do sistema desde o estado com menor consumo até o estado de consumo máximo. Determinar esse grafo e o caminhamento a ser adotado é parte da configuração do MPM em um dado sistema. O controlador, com base na informação sobre o estado corrente do sistema e a saída computada pelo PID, deve decidir quantos estados devem ser percorridos no caminhamento a fim de atingir um novo estado, o qual espera-se que reduza o erro computado. Isso pode exigir a alteração de mais de uma variável na equação de energia (por exemplo, desligar uma máquina e reduzir a freqüência da CPU de duas das restantes).

A implementação do núcleo do middleware proposto em um sistema exige a regulagem do controlador pela determinação das constantes PID e a identificação dos estados de consumo de energia admissíveis no grafo, bem como o caminhamento monotônico a ser usado em mudanças das configuração. Todos esses elementos devem ser transparentes para o desenvolvedor da aplicação, que deve apenas indicar os limites de desempenho aceitáveis. Não há nenhuma necessidade de dotar a aplicação de conhecimento sobre níveis de consumo possíveis, ou como determinar quais alterações são necessárias para se fazer um ajuste de consumo. Entretanto, MPM deve notificar a aplicação antes de tornar efetivas quaisquer mudanças escolhidas, a fim de permitir que ajustes sejam feitos face à nova situação. A Seção a seguir discute em mais detalhes a arquitetura do controlador e a interação entre a aplicação e as demais partes do sistema.

3. Middleware para gerência de energia

A arquitetura geral da solução proposta é apresentada na Figura 2. Os três elementos principais são a aplicação propriamente dita, que inclui a interface de programação para gerência de energia, os processos controladores e o processo gerente.

A aplicação determina o comportamento desejado e responde a notificações dos controladores. Estes, por sua vez, recebem informações das aplicações e do sistema operacional de cada máquina do cluster para determinar como estão sendo atendidas as demandas de desempenho e gerência de energia naquele nó. Todos os controladores se

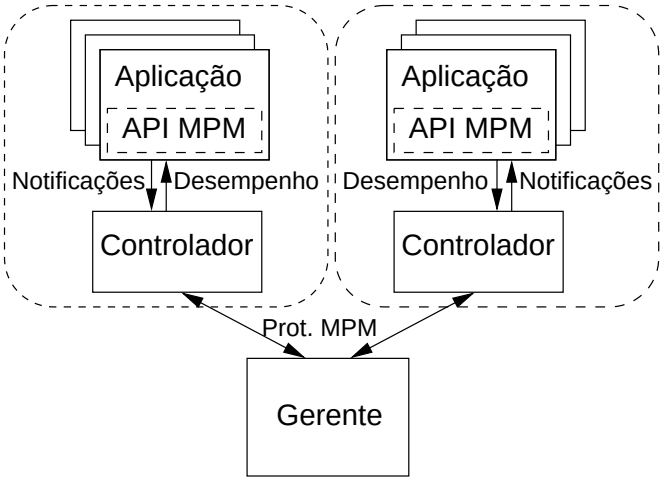

Figura 2: Arquitetura da solução proposta

comunicam periodicamente com um processo gerente responsável pela combinação das informações oriundas de todos os elementos do cluster e pelas tomadas de decisão. Essas decisões são por sua vez repassadas de volta aos controladores, que podem atuar sobre o sistema operacional de suas máquinas e notificar a aplicação de mudanças. As seções a seguir detalham cada um desses elementos.

3.1. Interface do usuário

A interface do usuário é embutida no código de cada aplicação na forma de um conjunto de primitivas disponíveis para o programador e definidas com um nível de abstração suficiente para isolar detalhes operacionais de equipamentos específicos para gerência de energia. Como discutido anteriormente, a aplicação deve ser capaz de exprimir suas demandas em termos de medidas de desempenho significativas para um serviço, como a latência das requisições ou o *throughput* do sistema. Assim sendo, a interface oferecida permite ao programador definir quais os níveis de desempenho mínimo desejáveis para o serviço em termos daquelas duas grandezas.

Um problema a ser resolvido no processo controlador é como determinar, por exemplo, o tempo de resposta de cada requisição atendida pela aplicação. Observações sobre o tráfego de rede e as várias conexões associadas a um determinado serviço são uma solução pouco confiável, uma vez que o controlador não tem conhecimento suficiente sobre a semântica da aplicação para identificar, no tráfego de rede, os momentos de início e fim de requisições no caso geral. Para solucionar esse problema, a interface inclui duas chamadas que devem ser usadas pela aplicação para notificar o controlador do início e fim de processamento de cada requisição.

Um identificador deve ser associado a cada requisição para permitir seu acompanhamento pelo controlador. Muitas aplicações já definem alguma forma de identificador para seu uso próprio e podem informar o controlador desse identificador. Caso tal identificador já não esteja disponível, o controlador define e retorna para a aplicação um identificador único para cada requisição através da primitiva de registro de requisições iniciadas. O identificador gerado de qualquer das duas formas deve ser usado pela aplicação quando da notificação de término do serviço de uma dada requisição. Dessa forma, a aplicação

tem controle sobre quais procedimentos devem ser observados e acompanhados pelo controlador para fins de determinação dos níveis de operação.

Devido a características próprias, certas aplicações podem não ser capazes de tolerar certas reduções de recursos para controle de energia. Por exemplo, um sistema pode depender da operação da memória virtual para acesso a arquivos (utilizando I/O mapeado em memória). Nesse caso, a aplicação deve ser capaz de indicar tais limitações ao sistema quando de sua configuração. Além disso, no caso em que o gerente do middleware decida pelo desligamento de uma máquina, a aplicação executando naquela máquina deve ser notificada da decisão a fim de garantir-lhe a opção de transferir informações importantes para outras instâncias executando em outras máquinas, ou para garantir que operações críticas sejam completadas antes do desligamento da máquina. A interface prevê então recursos do middleware para notificar a aplicação das decisões tomadas e para permitir que a aplicação tome precauções que julgue necessárias antes da ação de controle ser levada a cabo.

Notificações também são úteis para informar à aplicação quando recursos extras se tornam disponíveis, como no caso de ligação de máquinas ou aumento da velocidade de um processador. Nesse caso, uma aplicação mais complexa pode decidir pela migração de parte da carga de um nó mais carregado para o elemento que acaba de ser adicionado ao sistema.

3.2. Processo de controle

Em cada máquina do sistema um processo deve ser responsável pelo acompanhamento do comportamento das aplicações em execução e pela monitoração dos recursos do hardware e do sistema operacional de cada máquina. Todas essas informações devem ser combinadas pelo controlador e informadas ao processo gerente do middleware. A estrutura do processo de controle é apresentada na Figura 3.

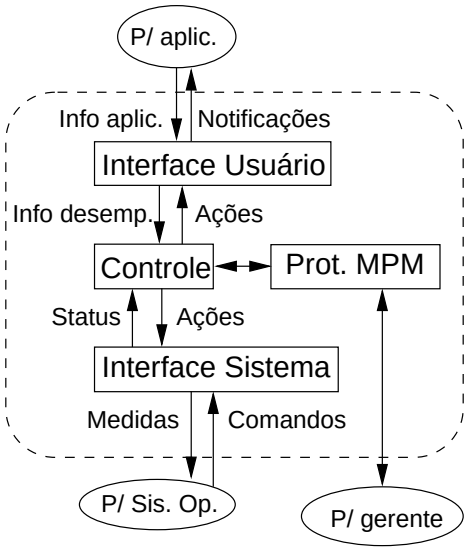

Figura 3: Estrutura do processo de controle

A interface com o usuário recebe as informações geradas a partir da API embutida na aplicação, controlando, por exemplo, as informações sobre início e fim de requisições

servidas. Essa informação é compilada e entregue ao núcleo do controlador na forma de medidas de desempenho consistentes com aquelas usadas pela aplicação para determinar níveis de serviço aceitáveis. Por outro lado, a interface com o sistema operacional é responsável por coletar dados como nível de utilização de CPU, memória e discos, bem como quaisquer indicadores de consumo de energia oferecidos pelo hardware em questão.

As informações das duas interfaces são processadas e periodicamente enviadas para o processo de gerência de energia através de um protocolo simples desenvolvido para esse fim. Comandos são por sua vez enviados do gerente para os controladores, por exemplo, para reduzir ou aumentar a velocidade da CPU, ou para desligar a máquina completamente. Esses comandos são notificados à aplicação através da interface do usuário, permitindo que ações específicas sejam tomadas pela aplicação antes da execução do comando. Após a notificação da aplicação o controlador atua sobre o sistema operacional para executar o comando informado pelo gerente.

3.3. Processo de gerência

O processo de gerência é responsável pela implementação da lógica de controle de energia, incluindo a implementação do algoritmo PID para o sistema. Sua função é coletar as informações dos diversos processos controladores, determinar se o sistema opera com excesso ou carência de recursos e atuar sobre o mesmo para corrigir as distorções verificadas. Como discutido anteriormente, com base no valor da diferença entre o desempenho especificado pela aplicação e o desempenho medido pelos controladores o gerente deve optar por adicionar ou remover recursos do sistema. Essa operação deve ser identificada através da informação embutida no middleware sobre as opções disponíveis para aumento ou redução da energia disponível, as quais devem ser detectadas e quantizadas durante a instalação do sistema.

Comandos de reconfiguração e desligamento de máquinas devem ser enviados para os controladores das mesmas. Caso a decisão seja pela adição de máquinas ao cluster o gerente deve iniciar o processo de ligação das mesmas, seja através de um mecanismo *wake-on-lan* ou utilizando hardware especialmente desenvolvido para esse fim. Em qualquer caso, o gerente deve acompanhar o processo para verificar a disponibilidade da nova máquina e para notificar o controlador na mesma sobre quaisquer operações de inicialização da aplicação que sejam necessárias. Normalmente, entretanto, tais operações deverão ser tomadas pelos processos da aplicação executando em outras máquinas, os quais serão notificados da disponibilidade de novos nós.

Em uma solução completamente distribuída o processo de gerência pode ser implementado de forma distribuída pelo cluster. Na implementação apresentada a seguir optou-se pela utilização de um processo central executando em um nó específico do sistema. Essa solução foi adotada para simplificar a implementação considerando o fato de que, em qualquer situação, pelo menos uma máquina do cluster deve permanecer ligada durante todo o tempo.

4. Estudo de Caso: *Cluster* de Servidores Web

Nesta seção apresentamos um estudo de caso onde o nosso middleware foi utilizado para gerenciar a energia de um cluster de servidores Web.

4.1. Arquitetura do Servidor Web

O servidor Web utilizado nos experimentos é um servidor seriado de documentos estáticos descrito em [Meira Jr. et al., 2002]. Este servidor responde apenas uma requisição a cada instante, mas realiza todo o processo de tratamento de uma requisição, do seu atendimento ao envio da página resposta, incluindo a leitura do recurso solicitado do disco do servidor.

A utilização desses servidores em clusters é bastante trivial, uma vez a natureza do serviço de provimento de páginas estáticas é facilmente paralelizável. Neste caso, basta adicionar novos servidores que contenham o conjunto de recursos que estejam sendo servidos para que tenhamos um novo servidor à disposição. Entretanto, a distribuição de requisições entre os vários servidores do cluster normalmente demanda um mecanismo mais elaborado [Cardellini et al., 2002], que leve em consideração aspectos como o recurso sendo solicitado, a sua origem ou a carga dos vários servidores, entre outros.

No caso dos experimentos apresentados nessa seção, utilizamos um mecanismo de distribuição de requisições baseado em chaveamento nível 4 por software implementado pelo LVS (*Linux Virtual Server Project*) [LVS, 2002], que, através de uma alteração do sistema operacional Linux, passa a se comportar como um servidor virtual, recebendo e redirecionando requisições para um grupo de servidores reais.

4.2. Caracterização da carga de Trabalho

De forma a gerar cargas de trabalho realistas nos experimentos descritos a seguir, realizamos a caracterização de uma carga de trabalho real observando os fatores mais relevantes para a atividade de gerência de energia. Esta carga de trabalho está disponível livremente e compreende as requisições submetidas ao Departamento de Computação da Universidade de Berkeley, EUA (www.cs.berkeley.edu).

O registro de acesso utilizado é do mês de dezembro de 2001 e compreende mais de 6 milhões de requisições gerando um tráfego superior a 165 TBytes. Essas requisições se destinam a aproximadamente 268 mil objetos que totalizam quase 48 TBytes de dados.

A Figura 4 mostra os gráficos de popularidade e de distribuição de frequência dos tamanhos de arquivos presentes na carga de trabalho. Em ambos os casos podemos observar que as distribuições são muito concentradas, indicando que tanto alguns arquivos específicos quanto arquivos de certos tamanhos têm uma maior probabilidade de serem requisitados.

A Figura 5 apresenta as distribuições de frequência cumulativas para os intervalos entre chegadas de requisições e a taxa de serviço atendida pelo servidor. Os intervalos entre chegadas de requisições estão em grande parte abaixo de 1 segundo. Podemos observar que as taxas de serviço atendidas pelo servidor raramente ultrapassaram 10 requisições por segundo. Mais ainda, essa taxa foi menor ou igual a 1 em mais de 90% dos casos.

Com base nessas observações podemos extrair algumas características relevantes para a nossa avaliação de gerenciamento de energia. A concentração em termos de popularidade de arquivos é contemplada no servidor, que escolhe os arquivos a serem lidos de acordo com essa popularidade. No caso do tamanho, ele poderia ser contemplado na carga de trabalho, entretanto, optamos por utilizar um tamanho fixo, de forma a melhor explorar os compromissos envolvendo taxa de demanda. Desta forma, a carga a ser

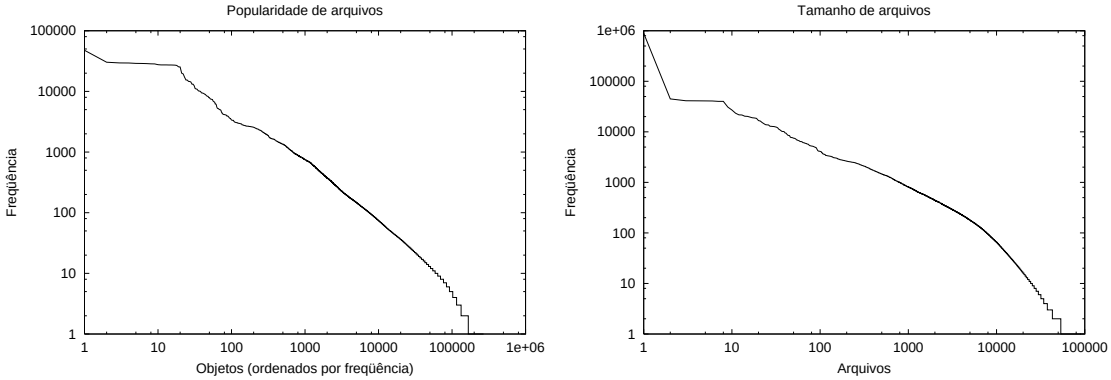

Figura 4: Popularidade e Distribuição de Frequência de Tamanhos de Arquivos

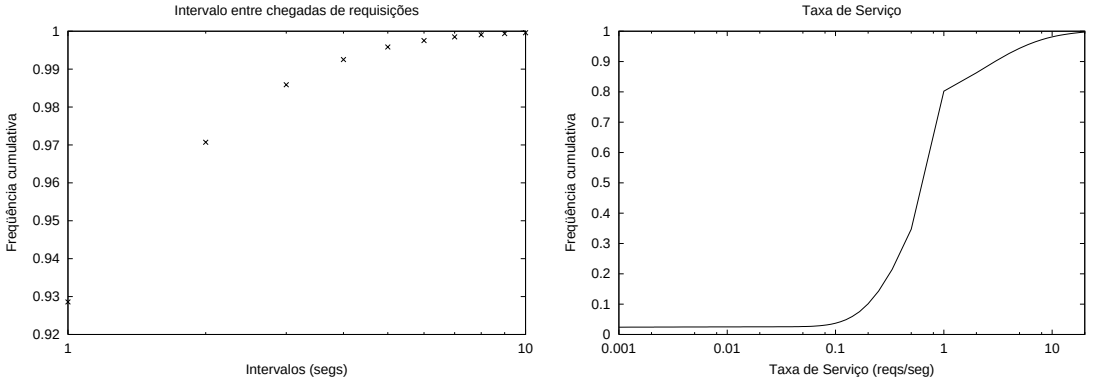

Figura 5: Intervalos entre Chegadas e Taxa de Serviço de Requisições

utilizada nos testes deve resultar em demandas variadas como conseqüência da taxa de requisições. Para refletir esse aspecto, a carga foi dividida em três fases. Na primeira fase a carga é monotonicamente crescente, iniciando com 3 e terminando com 15 requisições por segundo. A segunda fase é estável, mantendo a taxa de 15 requisições submetidas por segundo. A terceira fase possui tendência inversa à da primeira fase, quando a taxa reduz monotonicamente de 15 até 3 requisições por segundo.

4.3. Ambiente Experimental

O ambiente experimental consiste de sete máquinas conectadas por uma chave Fast Ethernet. Os servidores são Pentium 4 1,6Ghz, com 1GByte de memória. A máquina distribuidora de requisições é um AMD Athlon 1Ghz, com 512 MByte de memória. Os clientes são máquinas AMD Athlon 750Mhz, com 256 MByte de memória. Todas as máquinas executam sistema operacional Linux versão 2.4.20 e possuem discos IDE.

As máquinas servidoras possuem recursos de gerência de energia restritos e também limitaram a atuação do middleware. Mais especificamente, é possível apenas desligá-las e ligá-las (a partir do gerente) o que resulta em três configurações possíveis para o nosso cluster em termos de gerenciamento de energia, isto é, um, dois, ou três servidores atendendo requisições.

Em termos de software, cada servidor executa, além do servidor Web mencionado, o *daemon* de gerenciamento de energia, e, obviamente, o sistema operacional. Foi utili-

zada a versão 1.0.7 do módulo IPVS Netfilter na máquina distribuidora. Essa máquina também executa o programa de gerência de energia do cluster. A geração de carga pelos clientes utiliza o software `httperf`, versão 0.8, à execução do qual as máquinas clientes se encontram dedicadas.

As máquinas servidoras estão ligadas a um único *no-break* Engetron com interface de gerenciamento SNMP. Uma das informações providas pela MIB desta interface é o consumo de energia em Watts dos equipamentos a ele conectados. A medição de consumo foi então realizada através de requisições SNMP executadas a cada cinco segundos a partir de uma máquina de coleta.

4.4. Configuração do Middleware

Nesta seção discutimos a determinação de parâmetros para a operação do middleware e como esses parâmetros foram determinados no presente estudo de caso. Para determinar esses parâmetros, consideramos duas configurações estáticas que representam os dois extremos em termos de capacidade de processamento: um servidor e três servidores. A configuração com um servidor representa o menor consumo de energia, e também a menor capacidade de processamento, enquanto a configuração com três servidores representa a maior capacidade de processamento e também consumo de energia.

Utilizando a carga de trabalho descrita anteriormente, obtemos as medições de consumo de energia para cada configuração na Figura 6. Também podemos observar as taxas de serviço obtidas na Figura 7, onde fica clara a menor capacidade de processamento da configuração com 1 servidor. Finalmente, podemos observar a evolução do tempo de resposta para a configuração utilizando 3 servidores estáticos na Figura 8. Comparando estes resultados, podemos verificar que em ambas as configurações o tempo de resposta observado aumenta com a intensidade da carga de trabalho, embora a magnitude do aumento seja maior para a configuração com 1 servidor, como esperado. Por outro lado, o consumo de energia da configuração com 3 servidores é significativamente maior que a configuração com 1 servidor. Deve-se também notar que em ambos os casos há variações no consumo de energia mesmo para situações de manutenção de carga. Essas variações são explicadas pela ativação e desativação de mecanismos de refrigeração internos à cada maquina, dentre outros.

Uma informação fundamental para o funcionamento do middleware é o tempo de resposta a ser garantido pelo conjunto de servidores. Observando os gráficos e para fins de avaliação do middleware, vamos definir 0,2 segundos como o limite de latência, ou seja, atingir esse valor ativa novos servidores. Da mesma forma, tempos de resposta abaixo de 0,05 segundos causam a desativação de servidores. Na próxima seção apresentamos resultados utilizando esse limite.

4.5. Resultados Experimentais

Nesta seção apresentamos os resultados da utilização do MPM no cluster descrito. O experimento consiste em iniciar com uma configuração de apenas 1 servidor, e submeter a carga trifásica descrita. Apesar da sua simplicidade, essa carga vai permitir avaliar a velocidade e efetividade da reação tanto ao aumento quanto à redução de demanda de serviços, representada pela ativação e desativação, respectivamente, de servidores.

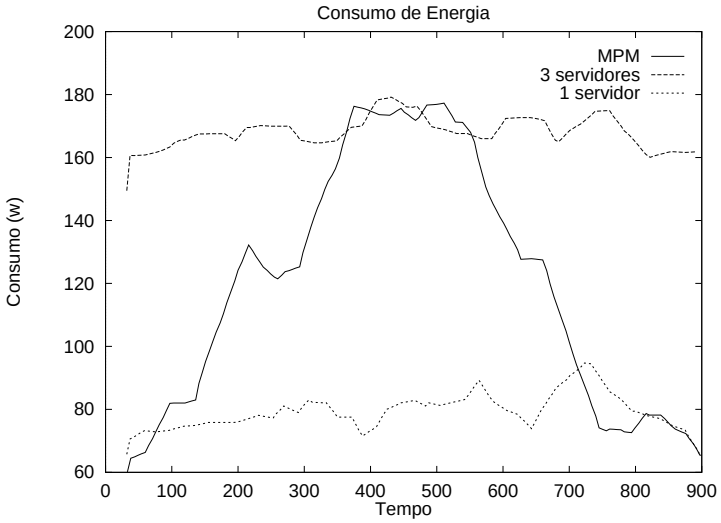

Figura 6: Consumo de energia em várias configurações

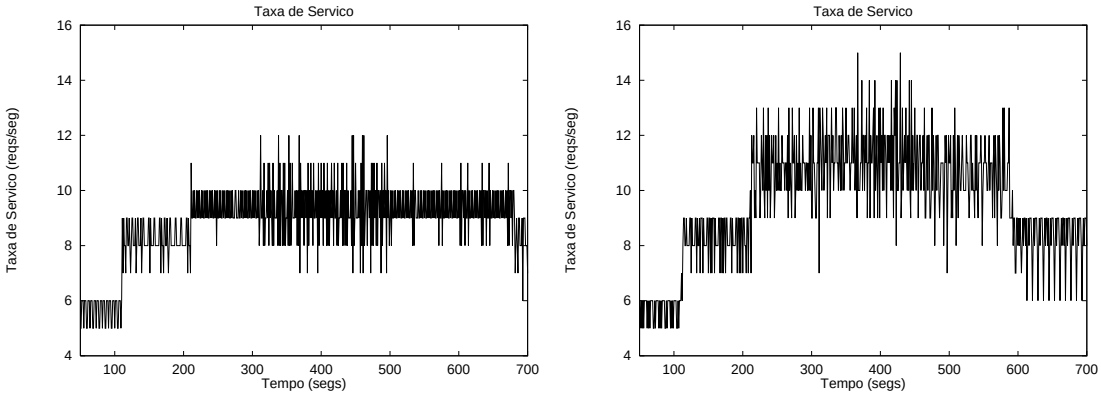

Figura 7: Taxas de Serviço para configurações com 1 e 3 servidores estáticos

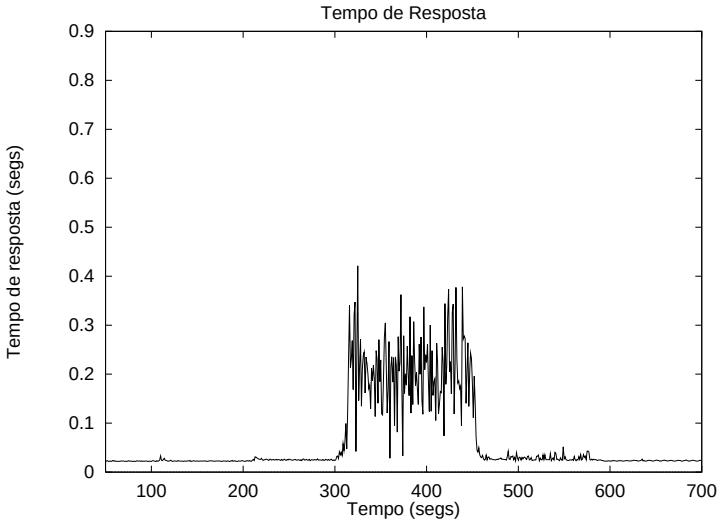

Figura 8: Tempo de resposta para configuração estática com 3 servidores

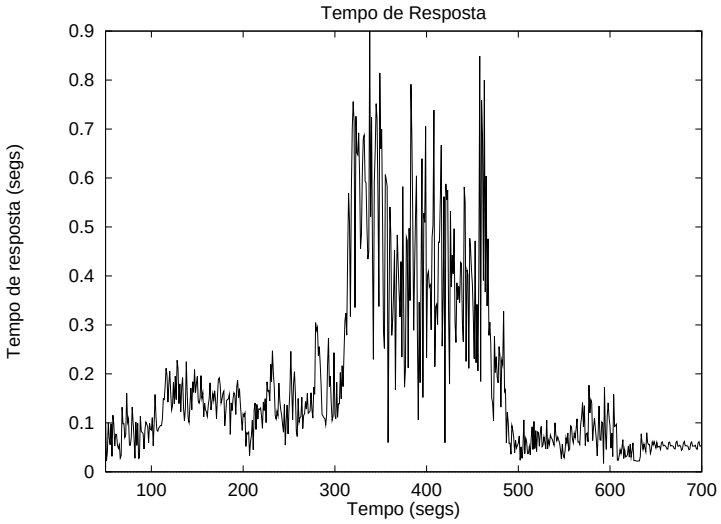

Figura 9: Tempo de resposta para configuração utilizando MPM

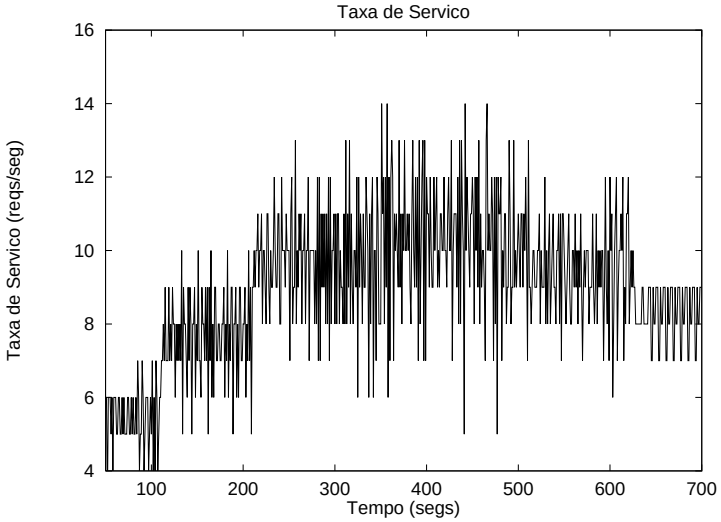

Figura 10: Taxas de Serviço para configuração utilizando MPM

Utilizando os limites definidos na Seção anterior, conseguimos os resultados apresentados nos gráficos das Figuras 6, 9 e 10. Em todos os casos podemos observar que a utilização do middleware representou um compromisso entre consumo e desempenho. Assim, a utilização do middleware permitiu reduzir em 27% o consumo de energia, que caiu de 43,52 Wh para 31,78 Wh. A latência média aumentou quando comparado à configuração que sempre utiliza três servidores, mas não a ponto de ser sensível ao usuário, passando de 0,067 para 0,174 segundos. Finalmente, a redução da taxa de serviço entre a configuração estática de três servidores e a utilizando MPM foi de apenas 11%, (de 8,24 para 7,40 requisições atendidas por segundo).

Esses resultados ilustram não apenas a eficácia do middleware proposto como também os ganhos significativos que podem ser atingidos em termos de redução de consumo de energia para aplicações Web.

5. Conclusões e trabalhos futuros

Neste artigo apresentamos MPM, um middleware para gerenciamento de energia de *clusters* de servidores Web. MPM cria uma camada de abstração para aplicações, evitando que programadores tenham que lidar com a complexidade de tarefas de controle de energia. MPM também propõe uma interface única com os recursos de gerenciamento de energia de hardware, os quais variam significativamente entre máquinas e fabricantes. Finalmente, MPM possui mecanismos para gerência cooperativa de grupos de máquinas, incluindo algoritmos extraídos da teoria de controle. Esses três componentes permitem que MPM faça um gerenciamento efetivo dos recursos computacionais com vistas a manter a qualidade dos serviços sendo realizados enquanto minimiza o consumo.

Com a finalidade de avaliar a eficácia da solução proposta verificamos o impacto da sua utilização em um *cluster* de três servidores Web submetidos a uma carga típica. A utilização do middleware permitiu reduzir em 27% o consumo de energia, sem afetar significativamente a latência média, que aumentou de 0,067 para 0,174 segundos, e a taxa de serviço, que diminuiu de 8,24 para 7,40 requisições atendidas por segundo.

Há diversas direções de trabalhos futuros. A avaliação do uso do MPM para outros serviços Web, como servidores de comércio eletrônico e máquinas de busca é interessante e necessária, tanto pela relevância dessas aplicações, quanto pela sua maior complexidade. A avaliação do MPM em plataformas que possuam mais recursos de gerência de energia, assim como o desenvolvimento de estratégias que permitam capturar mais facilmente os parâmetros de qualidade de serviço a serem atendidos pelo sistema que utiliza o middleware. O aperfeiçoamento dos mecanismos de controle utilizados é também de interesse, possivelmente utilizando equações mais elaboradas, como controladores PID auto-configuráveis que eliminam a necessidade de se definir as constantes K_p, K_i e K_d, ou considerando novos fatores relevantes ao problema de gerência de energia.

Agradecimentos

Este trabalho foi parcialmente financiado com recursos do CNPq, do programa Pronex, e recursos oriundos da Lei de Informática e geridos pelo MCT. O *no-break* utilizado nos experimentos foi gentilmente cedido pela empresa Engetron, e a sua utilização para fins de gerência de energia foi possível graças à colaboração do Prof. Antônio Otávio Fernandes, do DCC–UFMG, Marcos Pêgo de Oliveira e Wilton de Castro Padrão, da Engetron, e da equipe do Lecom–DCC–UFMG.

Referências

Bohrer, P., Elnozahy, E., Keller, T., Kistler, M., Lefurgy, C., and Rajamony, R. (2002). The case for power management in web servers. In Graybill, R. and Melhem, R., editors, *Power-Aware Computing*. Kluwer.

Cardellini, V., Casalicchio, E., Colajanni, M., and Yu, P. (2002). The state of the art in locally distributed web-server systems. *ACM Computing Surveys*, 34(2):1–49.

Chase, J. S., Anderson, D. C., Thakar, P. N., Vahdat, A. M., and Doyle, R. P. (2001). Managing energy and server resources in hosting centres. In Ganger, G., editor, *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP-01)*,

volume 35, 5 of *ACM SIGOPS Operating Systems Review*, pages 103–116, New York. ACM Press.

Douglis, F. and Krishnan, P. (1995). Adaptive disk spin-down policies for mobile computers. *Computing Systems*, 8(4):381–413.

Flinn, J. and Satyanarayanan, M. (1999). Energy-aware adaptation for mobile application. In *Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP'99)*, pages 48–63. Kiawah Island Resort, near Charleston, Sout Carolina.

Franklin, G. F., Powell, D. J., and Workman, M. L. (1997). *Digital Control of Dynamic Systems*. Addison-Wesley.

Lebeck, A., Fan, X., Zeng, H., and Ellis, C. (2000). Power aware page allocation. *ACM SIGPLAN Notices*, 35(11):105–116.

LVS (2002). Linux virtual server project. http://www.linuxvirtualserver.org.

Meira Jr., W., Murta, C., Campos, S., and Guedes, D. (2002). *Sistemas de Comércio Eletrônico: Projeto e Desenvolvimento*. Editora Campus.

Pinheiro, E., Bianchini, R., Carrera, E. V., and Heath, T. (2002). Dynamic cluster reconfiguration for power and performance. In Benini, L., Kandemir, M., and Ramanujam, J., editors, *Compilers and Operating Systems for Low Power*. Kluwer Academic Publishers.

Weiser, M., Welch, B., Demers, A., and Shenker, S. (1994). Scheduling for reduced CPU energy. In USENIX, editor, *Proceedings of the First USENIX Symposium on Operating Systems Design and Implementation (OSDI): November 14–17, 1994, Monterey, California, USA*, pages 13–23, Berkeley, CA, USA. USENIX.