Escalonamento com Prioridade na Alocação Ciente de Energia de Máquinas Virtuais em Nuvens

Daniel G. Lago[1], Edmundo R. M. Madeira[2], Luiz Fernando Bittencourt[3]

Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Av. Albert Einstein, 1251, Campinas – SP – Brazil – 13083-852

lago[1]@lrc.ic.unicamp.br, {edmundo[2], bit[3]}@ic.unicamp.br

Resumo. *Para cumprir requisitos de QoS, um provedor de nuvem pode priorizar determinadas cargas de trabalho dentro de um data center. Por outro lado o provedor deve reduzir o consumo energético dos recursos. Neste trabalho apresentamos um algoritmo para submissão de cargas com prioridades. Incorporamos o algoritmo proposto a outro apresentado em trabalho anterior, baseado em conceitos de Computação Verde, capaz de reduzir substancialmente o consumo de energia, especialmente em data centers heterogêneos. Resultados obtidos através de simulações mostraram redução no tempo de execução de cargas com alta prioridade, sem comprometer significativamente o consumo de energia.*

Abstract. *To meet QoS requirements, a cloud provider can prioritize predefined workloads in a data center. On the other hand the provider must reduce the energy consumption of resources. In this paper we present an algorithm for submission of loads with priorities. We incorporated the presented algorithm into a previous work algorithm which was able to substantially reduce energy consumption, especially in heterogeneous data centers. Results from simulations showed a reduction in execution time of loads with high priority, without compromising power consumption significantly.*

1. Introdução

Uma das mais notáveis tecnologias emergentes da Internet nos dias de hoje é a computação em nuvem. Esse conceito se refere ao uso das capacidades de processamento, memória e armazenamento de computadores e servidores compartilhados através da internet, seguindo os princípios da computação em grade.

Em [Lago et al. 2011], projetamos e avaliamos um algoritmo para escalonamento de tarefas em nuvens, usando o conceito de Computação Verde. O algoritmo focou na minimização do consumo de energia em *data centers* e, consequentemente, na redução da poluição potencial gerada. Para atingir esse objetivo foram usados princípios de gerência distribuída de energia, ajuste dinâmico de tensão e frequência, migração de cargas virtuais e desligamento de servidores subutilizados. Também aplicamos o conceito de controle de refrigeração ativa, originalmente usado para redução de ruído, para minimizar o consumo de energia; além de usarmos também o conceito de eficiência energética, obtida pela razão entre MIPS e potência consumida. Como resultado conseguimos um algoritmo competitivo com os melhores encontrados na literatura para

data centers homogêneos, sendo capaz de superá-los em *data centers* heterogêneos.

No presente trabalho criamos um segundo algoritmo, para ser executado simultaneamente com o primeiro, no processo de escalonamento de computação em nuvem. Enquanto o algoritmo trabalhado anteriormente foca na alocação de máquinas virtuais, o algoritmo apresentado neste trabalho lida com a submissão das cargas de trabalho para essas máquinas.

No algoritmo proposto adicionamos características de qualidade de serviço ao processo de escalonamento, permitindo a redução do tempo de processamento de cargas com alta prioridade, sem comprometer significativamente o consumo de energia. Em outras palavras podemos dizer que neste trabalho buscamos resolver o seguinte problema: dado um conjunto composto por M máquinas virtuais e C cargas de trabalho, tendo cada carga C_i uma prioridade P_i associada, efetuar o escalonamento de C em M minimizando os tempos de execução das cargas com maior prioridade P_i e mantendo o consumo.

Para atingir esse objetivo, trabalhamos para que as cargas de maior prioridade fossem executadas em máquinas virtuais mais aptas para seu processamento. Com essa política obtivemos melhoras substanciais nos tempos de execução das cargas de trabalho com altas prioridades, sem prejudicar criticamente as cargas com baixa prioridade e praticamente sem aumento do consumo de energia no escalonamento.

Este artigo está organizado da seguinte maneira: na Seção 2 apresentamos alguns trabalhos relacionados, enquanto na Seção 3 mostramos conceitos básicos. Na Seção 4, o algoritmo proposto é descrito; e na Seção 5 validamos o algoritmo e seus resultados obtidos através de simulação. Na Seção 6 apresentamos as conclusões.

2. Trabalhos Relacionados

Escalonamento em nuvem pode ser dividido segundo dois pontos de vista: do usuário da nuvem e do provedor da nuvem. Do ponto de vista do usuário, um escalonador deve ter como objetivo a minimização tanto do tempo de execução quanto do custo para o usuário [Bittencourt e Madeira 2011]. Por outro lado, do ponto de vista do provedor, um escalonador de tarefas deve melhorar a utilização de recursos reduzindo a manutenção e os custos de energia [Dasgupta et al. 2011]. O presente trabalho foca no escalonamento do ponto de vista do provedor.

[Beloglazov e Buyya 2010] propõem um sistema de administração de recursos para consumo eficiente de energia que reduz os custos operacionais e provê qualidade de serviço. A economia de energia é obtida através da consolidação contínua de máquinas virtuais de acordo com a utilização de recursos, topologias de redes virtuais estabelecidas através das máquinas virtuais e estados de temperaturas dos nós. Resultados obtidos por simulação mostraram 83% de economia de energia comparado a um sistema sem gerência de energia.

[Binder e Suri 2009] propõem um algoritmo de alocação e despacho de tarefas que minimiza o número de servidores ativos requeridos, gerenciando o ambiente para atingir um consumo de energia inversamente proporcional ao número de *threads* que executam as cargas de trabalho.

[Srikantaiah et al. 2008] estudam como obter a consolidação da eficiência energética baseada no inter-relacionamento entre consumo de energia, utilização de

recursos e desempenho das cargas de trabalho consolidadas. É mostrado que existe um ponto ótimo na operação entre esses parâmetros baseado no problema do empacotamento aplicado ao problema da consolidação.

[Xu et al. 2009] propõem estratégias de escalonamento com múltiplas restrições de *QoS* para múltiplos *workflows* em computação em nuvem, aumentando a taxa de sucesso para tarefas de múltiplos *workflows* com diferentes requisitos de *QoS*.

3. Conceitos Básicos

Nesta seção são apresentados os conceitos usados nos algoritmos deste trabalho.

3.1. Migração de Máquinas Virtuais

Máquinas virtuais são componentes básicos de *data centers*, principalmente devido às suas capacidades de isolamento de cargas de trabalho, consolidação e migração; recursos esses que permitem a um *data center* servir múltiplos usuários de um modo seguro, flexível e eficiente. Como resultado, essas infraestruturas virtualizadas são consideradas componentes-chave para conduzir o emergente paradigma da Computação em Nuvem [Buyya et al. 2009].

A migração de cargas virtuais procura aperfeiçoar a gerência, desempenho e tolerância a falha de sistemas. Mais especificamente, as razões que justificam a migração de máquinas virtuais em um sistema de produção incluem: a necessidade do balanceamento de carga e a necessidade de desligar servidores para manutenção após a migração de suas cargas de trabalho para outros servidores [Voorsluys et al. 2009].

Neste trabalho usamos o conceito de migração de máquinas virtuais com o objetivo de migrar as cargas de servidores subutilizados e, em seguida, desligá-los, desse modo a maximizar a economia de energia.

3.2. Dimensionamento Dinâmico de Tensão e Frequência

Dimensionamento Dinâmico de Tensão (DVS – *Dynamic Voltage Scaling*) [Rabaey 1996] é uma técnica de gerência de energia em uma arquitetura computacional onde a tensão de um determinado componente pode ser alterada de acordo com parâmetros específicos. Fabricantes de processadores se beneficiam dos recursos de DVS para minimizar o consumo de energia de seus dispositivos e, também, reduzem a frequência de tais equipamentos, permitindo-os permanecer estáveis.

O conceito de Dimensionamento Dinâmico de Tensão e Frequência (DVFS) é aplicado nas tecnologias *Intel Speed Step* e *AMD Coll'n'Quiet*, que ajustam automaticamente, em hardware, a tensão e frequência dos processadores de acordo com suas cargas de trabalho. Se há altas cargas de trabalho, o processador aumenta os níveis de tensão e frequência, caso contrário diminui esses níveis. Os estados possíveis de frequência e tensão são chamados *P-States*.

Aplicamos DVFS nos algoritmos que propomos, usando de forma inteligente a energia do processador, adaptando-o às cargas que ele necessita processar.

3.3. Fan Control

Fan Control é um dos nomes dados para representar a tecnologia que monitora a temperatura dos computadores e, mediante condições térmicas, regula a intensidade dos

dissipadores ativos. Várias fabricantes de placas-mãe possuem diversos modos de operação dessa tecnologia, cada uma com seus próprios nomes, por exemplo: AOpen: SilentTEK; ASUS: Q-Fan; MSI: Core Center; Abit: μGuru; Gigabyte: EasyTune 6; Intel S478: Active Monitor / Desktop Control Centre; Intel S775: Desktop Utilities e Dell Inspiron/Latitude/Precision: I8kfanGUI. Essas tecnologias controlam a intensidade do dissipador ativo, através do sistema operacional ou diretamente da BIOS, de acordo com o aquecimento do processador.

Apesar dessa tecnologia originalmente ter sido desenvolvida para redução do ruído dos dissipadores ativos, ela possui uma consequência direta: a redução do consumo de energia. Por exemplo, quando uma ventoinha que inicialmente é alimentada por 12 volts passa a ser alimentada por 5 volts, ela consumirá 83% menos energia. Neste trabalho nós usamos esse mecanismo para otimizar o consumo de energia.

4. Os Algoritmos

Em um *data center* operando numa nuvem, um dos servidores deve realizar a função de *broker*, que lidará com eventos tais como: requisições de recursos, criação de máquinas virtuais, retorno do processamento das cargas virtuais e finalização das cargas e máquinas virtuais. Então, o *broker* é a entidade responsável por executar os algoritmos apresentados nesta seção.

Em nosso trabalho anterior propusemos um algoritmo que age na alocação de servidores para a criação de máquinas virtuais [Lago et al. 2011]. O algoritmo proposto no presente trabalho age na submissão de cargas de trabalho para as máquinas virtuais previamente criadas, adicionando qualidade de serviço ao escalonamento em nuvens. Este algoritmo otimiza o escalonamento ponderando as cargas em vários níveis de prioridade, por exemplo: prioridades alta, média e baixa.

Os algoritmos mostrados neste trabalho foram projetados para lidar com o escalonamento em nuvens computacionais de *data centers* não-federados, portanto assume-se que todos os servidores estão na mesma nuvem. Também é considerado que as tecnologias de DVFS e *Fan Control* estão ativadas para todos os nós.

Nosso escopo inclui *data centers* homogêneos, constituídos por todas as máquinas com iguais configurações de hardware, e *data centers* heterogêneos, que possuem agrupamentos de máquinas com configurações distintas de hardware. Além disso partimos do princípio que não é possível conhecer ou estimar os pesos das cargas de trabalho que serão alocadas nas máquinas virtuais do *data center*. Em decorrência disso os algoritmos foram projetados para adaptarem-se dinamicamente às necessidades de processamento, otimizando o consumo de energia *on-the-fly*. Uma vez que todas as máquinas virtuais tenham sido criadas, é iniciado o processo de submissão das cargas.

4.1. Alocação de Máquinas Virtuais

Nesta seção descrevemos os principais passos do algoritmo de alocação de máquinas virtuais *Lago Allocator* [Lago et al. 2011], que utilizamos em paralelo com o algoritmo que propomos neste trabalho.

Para cada máquina virtual a ser alocada no *data center*, é verificado entre todos os servidores, quais são candidatos para receber a nova máquina virtual, ou seja, quais são os que possuem recursos (processador, RAM, largura de banda, etc.) suficientes para alocá-la. Para cada servidor candidato, a sua eficiência em energia é verificada,

através da razão entre MIPS e potência consumida pelo servidor. O algoritmo então classifica o servidor que possui a melhor eficiência em energia para alocar a máquina virtual.

Um empate pode ocorrer, especialmente em ambientes homogêneos onde as máquinas possuem MIPS e consumo de potência iguais. Neste caso, uma comparação de desempate é realizada da seguinte maneira: partindo do princípio que o sistema possui DVFS ativado, juntamente com *Fan Control*, haverá provavelmente algum servidor consumindo menos energia que os demais. Então os servidores serão confrontados um contra os outros e o que consumir menor potência será escolhido para a máquina virtual.

Ainda assim poderá acontecer um empate. Nessa hipótese o servidor candidato com a maior utilização de CPU será escolhido. Essa escolha se baseia no fato de que se a CPU está processando maior carga, ela levará mais tempo para atingir seu limiar de migração de máquinas virtuais.

Caso ainda haja um empate, o servidor com maior capacidade de processamento é escolhido, pois ele é potencialmente mais apto para receber um número maior de cargas de trabalho. Na circunstância de outro empate um servidor será escolhido aleatoriamente.

4.2. Submissão de Cargas de Trabalho

Neste trabalho propomos a adição de qualidade de serviço em um escalonamento de tarefas para computação em nuvens ciente de energia, além de estudarmos seu comportamento. A qualidade de serviço que propomos baseia-se na adição de prioridades às cargas que serão processadas pelas máquinas virtuais, minimizando o tempo de execução das cargas que possuírem maior prioridade.

Nosso algoritmo atua nas cargas de trabalho que não possuem vínculo com alguma máquina virtual específica, ou seja, ele pode determinar em qual máquina virtual cada carga de trabalho será alocada. Faz parte do nosso objetivo que a adição da qualidade de serviço traga vantagens, impactando o mínimo possível no consumo de energia, no sentido de não aumentá-lo. Descrevemos a seguir algumas funções usadas pelo algoritmo de submissão de cargas que propomos.

A função *sort_decreasing_by_free_MIPS(vmList)* ordena um vetor *vmList* contendo objetos que representam máquinas virtuais em ordem decrescente de MIPS disponíveis; *getPriority(c)* retorna a prioridade de uma carga de trabalho *c*. A função *send(c, vm)* encaminha uma carga de trabalho *c* para ser processada em uma máquina virtual *vm*. A função *resort_decreasing_by_free_MIPS(vmList, i)* reordena a máquina virtual no índice *i* de uma lista de máquinas virtuais *vmList* em ordem decrescente de MIPS disponíveis, aplicada no caso de um vetor que possua apenas um elemento fora de posição. As constantes *MAX_PRIORITY* e *MIN_PRIORITY* são valores inteiros que representam as prioridades máxima e mínima possíveis para uma carga virtual qualquer.

Apresentamos no Algoritmo 1 uma versão geral desse algoritmo. No Algoritmo 2 apresentamos uma versão otimizada para o caso em que o número de máquinas virtuais é igual ao de cargas de trabalho. Assim como no [CloudSim 2009], representamos uma *cloudlet* como uma tarefa submetida às máquinas virtuais do *data center*.

Algoritmo 1: submitCloudlets()

1 **Vm[]** vmList // Lista contendo as máquinas virtuais que podem ser alocadas para

```
    cloudlets
2   Cloudlet[] cloudletList // Lista de cloudlets que deverão ser alocadas nas máquinas
    virtuais
3
4   sort_decreasing_by_free_MIPS(vmList)
5   for int current_priority := MAX_PRIORITY to MIN_PRIORITY do
6     foreach cloudlet in cloudletList do
7       if (getPriority(cloudlet) == current_priority) then
8         send(cloudlet, vmList[0])
9         resort_decreasing_by_free_MIPS(vmList, 0)
10      end if
11    end foreach
12  end for
```

Esse algoritmo inicialmente ordena uma lista composta pelas máquinas virtuais em ordem decrescente de MIPS disponíveis (linha 4). Após isso é efetuada uma varredura para todas as prioridades disponíveis, da maior para a menor (linha 5). A cada iteração são verificadas todas as cargas que fazem parte daquela prioridade (linha 6) e as cargas que possuírem a prioridade da iteração corrente são submetidas para a máquina virtual com maior MIPS disponível (linha 8), seguido pela reordenação da lista de máquinas virtuais (linha 9).

Se em um determinado instante não existirem máquinas virtuais aptas para receberem novas cargas de trabalho, estas serão postergadas. Note que o Algoritmo 1 funciona bem, mas ele pode ser otimizado. Observando nuvens onde cada carga de trabalho é processada por uma máquina virtual distinta, ou seja, o número de cargas é igual ao de máquinas virtuais, podemos remover a função *resort_decreasing_by_free_MIPS*, conforme descrito no Algoritmo 2:

Algoritmo 2: submitCloudlets()

```
1   int vmIndex // Índice da máquina virtual que está sendo processada
2   Vm[] vmList // Lista contendo as máquinas virtuais que podem ser alocadas para
    cloudlets
3   Cloudlet[] cloudletList // Lista de cloudlets que deverão ser alocadas nas máquinas
    virtuais
4
5   sort_decreasing_by_free_MIPS(vmList)
6   vmIndex := 0
7   for int current_priority := MAX_PRIORITY to MIN_PRIORITY do
8     foreach cloudlet in cloudletList do
9       if (getPriority(cloudlet) == current_priority) then
10        send(cloudlet, vmList[vmIndex])
11        vmIndex++
12      end if
13    end foreach
14  end for
```

Com todas as cargas submetidas, o *broker* passará a verificar os servidores e desligar aqueles em que todas as máquinas virtuais tenham terminado de processar suas cargas, além de verificar quais servidores estão sendo utilizados abaixo de um limiar

predefinido para migrar as máquinas virtuais destes para desligá-los.

5. Simulações

Para uma simulação eficiente que analisa diversos cenários, é fundamental a escolha de um simulador robusto. O [CloudSim 2009] é um simulador relevante para computação em nuvem, reconhecido academicamente com citações em centenas de trabalhos publicados. Ele torna possível elaborar simulações de ambientes de computação em nuvem e avaliar algoritmos de aprovisionamento de recursos, além de efetuar análises de duração de simulações e consumo de energia. Por esses motivos constatamos que o *CloudSim* é capaz de suprir as necessidades deste trabalho e, portanto, foi escolhido como simulador.

5.1. Regras de Simulação

Para avaliar os algoritmos apresentados e suas interações, as métricas relevantes para as simulações foram o consumo de energia, o *makespan* e o tempo de conclusão das *cloudlets* considerando suas prioridades.

Para realizar comparações apropriadas entre os algoritmos, algumas definições são apresentadas: seja A um algoritmo de escalonamento para computação em nuvem; seja C um conjunto de configurações que são usadas pelo algoritmo, a saber: desligamento de servidores não utilizados, migração de cargas virtuais, DVFS e *Fan Control*; e seja P um conjunto de parâmetros para simulações: número de servidores, número de máquinas virtuais e milhões de instruções que cada *cloudlet* executa.

EC(A, C, P) é o consumo médio de energia de um algoritmo A com o conjunto de configurações C e parâmetros P; *ECI(A, C, P)* é o intervalo de confiança em 95% de *EC(A, C, P)*. Um algoritmo A_1 com configurações C_1 e parâmetros de simulação P é considerado melhor em consumo de energia do que um algoritmo A_2 com configurações C_2 e parâmetros P, $A_1 \neq A_2$, se a desigualdade mostrada em (1) for verdadeira.

TC(A, C, P) é o tempo médio para realização de todas as tarefas (*makespan*) do algoritmo A com o conjunto de configurações C e parâmetros P; e *TCI(A, C, P)* é o intervalo de confiança em 95% de *TC(A, C, P)*. Um escalonamento realizado por um algoritmo A_1 é considerado intoleravelmente mais lento do que A_2 para os parâmetros de simulação P se, para um mesmo conjunto de configurações C e parâmetros de simulação P, a desigualdade (2) for satisfeita.

$$EC(A_1, C_1, P) + ECI(A_1, C_1, P) < EC(A_2, C_2, P) - ECI(A_2, C_2, P) \quad (1)$$

$$TC(A_1, C, P) + TCI(A_1, C, P) > 2 \times (TC(A_2, C, P) - TCI(A_2, C, P)) \quad (2)$$

Para descrever os algoritmos e configurações usadas para cada conjunto de simulações, usaremos as seguintes siglas: MPD = *Minimum Power Diff* [CloudSim 2009]; LA = *Lago* Allocator; LAQ = *Lago Allocator* com Qualidade de Serviço (*QoS*) ativada. Siglas para as configurações do algoritmo: N = *Non Power Aware*; P = *Power Aware*; D = DVFS; T = *Threshold* (limiar de migração de máquinas virtuais); F = *Fan Control*. Por exemplo, o *Lago Allocator* com configurações *Power Aware*, DVFS, *Threshold* e *Fan Control* será referenciado como LA PDTF. Para todos os valores de medição de energia apresentados nas próximas seções a métrica utilizada é *quilowatts * hora* e, para medidas de tempo, *segundos*.

5.2. Configurações do Data Center

Um *data center* é um conjunto físico de máquinas conectadas por uma rede disponíveis para receber máquinas virtuais e, consequentemente, cargas de trabalho (*cloudlets*). As simulações para avaliar o algoritmo proposto foram conduzidas com *data centers* pequenos (10 servidores, 20 máquinas virtuais e 20 *cloudlets*), médios (100 servidores, 200 máquinas virtuais e 200 *cloudlets*) e grandes (1000 servidores, 2000 máquinas virtuais e 2000 *cloudlets*), possuindo servidores homogêneos e heterogêneos.

5.3. Configurações dos Servidores

Define-se as configurações comuns dos servidores para *data centers* homogêneos e heterogêneos da seguinte forma: cada servidor possuirá uma entidade de processamento (o processador), 1 TB de espaço em disco, 24 GB de RAM e *ethernet gigabit*. Assume-se que os servidores possuem arquitetura x86, tendo como sistema operacional GNU/Linux; e usam como monitor de máquina virtual o Xen, assim como as próprias máquinas virtuais.

Cada *cloudlet* usará uma entidade de processamento e cada *cloudlet* submetida ao *data center* possui 300 bytes antes e após o processamento (padrão dos modelos CloudSim). As *cloudlets* no *data center* possuem 100.000, 150.000, 200.000, 250.000 e 300.000 milhões de instruções em uma distribuição *round-robin.* Cada *cloudlet* pode assumir uma entre três prioridades: baixa, média e alta; sendo que a distribuição dessas prioridades também será *round-robin* com prioridades alta, média, média e baixa (mantendo-se 25% de cargas com baixa prioridade, 50% com prioridade média e 25% alta prioridade). A Tabela 1 exemplifica como será essa distribuição para um conjunto de *cloudlets*. CID representa o ID da *cloudlet* e MI milhões de instruções. Pretendemos efetuar a avaliação de casos mais específicos em trabalhos futuros.

As máquinas virtuais do *data center* possuem capacidade de processamento de 500, 750 e 1000 MIPS em uma distribuição *round-robin*. Por exemplo, em uma simulação onde 20 máquinas virtuais são geradas, 7 máquinas virtuais serão criadas com 500 MIPS, 7 máquinas virtuais terão 750 MIPS e 6 máquinas virtuais ficarão com 1000 MIPS. Além disso adotamos que cada máquina virtual possui 128 MB de RAM, 2.500 Kbps de largura de banda; cada máquina virtual tem tamanho de imagem de 2.500 MB.

Tabela 1: Exemplo de Distribuição de Cloudlets e Prioridades

CID	***C1***	***C2***	***C3***	***C4***	***C5***	***C6***
MI	100 k	150 k	200 k	250 k	300 k	100 k
Prioridade	Baixa	Média	Média	Alta	Baixa	Média

Nas simulações com DVFS ativado, foi assumido que quando o servidor estiver trabalhando no nível mínimo de processamento consumirá 70% de sua potência máxima e com o crescimento de sua carga de trabalho adotaremos um modelo de crescimento linear para calcular a potência consumida. Por exemplo, quando um servidor de 250 watts estiver trabalhando a 0% de carga ele consumirá 175 watts; quando estivar a 50% da capacidade consumirá 212 watts e quando estiver trabalhando na capacidade total consumirá 250 watts.

Para simulações com *threshold* ativado, tenta-se migrar as máquinas virtuais dos servidores que estejam operando abaixo de 80% de suas capacidades de processamento, seguido pelo desligamento destes.

Para as simulações, nos baseamos no *cooler stock* que acompanha o Intel i5 que trabalha com 0,6 ampères e tensão máxima de 12 volts. Em outras palavras ele pode consumir até 7,2 watts de potência. Apesar dos mecanismos de refrigeração ativa permitirem o completo desligamento do *cooler* em boas circunstâncias, nós decidimos manter uma margem de segurança para garantir que a simulação seja válida, então o *cooler* nunca desligará. Portanto adotamos que o consumo mínimo das ventoinhas será quando elas estiverem em baixa rotação, na tensão de 5 volts.

O aquecimento dos processadores depende de muitos fatores, incluindo temperatura, tipo de dissipador, arquitetura e litografia. Dado que não existe um padrão de aquecimento entre todos os processadores existentes, um modelo válido para todos é muito difícil de estimar. Partindo do princípio de que a temperatura do processador é diretamente proporcional à potência dissipada e que aumenta proporcionalmente de acordo com a frequência do processador e ao quadrado de sua tensão de operação, adotamos que a potência consumida pelo *cooler* será proporcional à potência consumida pelo processador com DVFS ativado, adotando um modelo linear para tal.

5.4. Configurações para Data Centers Homogêneos

Para as simulações em *data centers* homogêneos, as seguintes configurações foram adotadas: cada servidor em um *data center* homogêneo possui uma entidade de processamento (processador contendo um núcleo) com 2.000 MIPS e o consumo de potência de cada servidor é definido como 250 watts.

5.5. Configurações para Data Centers Heterogêneos

Para simulações em *data centers* heterogêneos, as seguintes configurações foram adotadas: cada servidor em um *data center* heterogêneo possui uma única entidade de processamento, contendo as seguintes MIPS em uma distribuição *round-robin*, 1.000, 1.500, 2.000, 2.500, 3.000; e cada servidor em um *data center* heterogêneo possui o seguinte consumo de potência em uma distribuição *round-robin*, 200, 250, 300 e 350, como exemplificado na Tabela 2.

Tabela 2: Exemplo de Configuração Inicial de Servidores em um Data Center Heterogêneo

Servidor	***h1***	***h2***	***h3***	***h4***	***h5***	***h6***	***h7***
MIPS	1.000	1.500	2.000	2.500	3.000	1.000	1.500
Potência (watts)	200	250	300	350	200	250	300

5.6. Algoritmos e Configurações

Para validar o algoritmo proposto, nomeado *Lago Allocator* com qualidade de serviço (*QoS*), os resultados das simulações foram comparados ao melhor algoritmo encontrado com objetivo de gerenciar energia no CloudSim – *Minimum Power Diff*. Também usamos o *Lago Allocator* original para comparar aos novos algoritmos, que contemplam *QoS*. Não comparamos os resultados a dois algoritmos clássicos da literatura – *Round Robin* e *Best Resource Selection* – pois esses já se demonstraram ser inferiores ao MPD para o objetivo do presente trabalho em [Lago et al. 2011]

O algoritmo *Minimum Power Diff*, implementado por Beloglazov et al., é usado como modelo principal de economia de energia no *CloudSim*. Cada máquina virtual recebida é alocada ao servidor que consumirá a menor quantidade de energia para

executar a máquina virtual. Todos os algoritmos postergarão o escalonamento quando não houver mais servidores capazes de receber máquinas virtuais.

Os algoritmos escolhidos foram implementados usando diferentes combinações dos seguintes recursos: *Power Aware* (habilidade de desligar servidores desocupados); DVFS (Dimensionamento Dinâmico de Tensão e Frequência), tratando os *P-States* como um modelo linear onde os servidores desocupados consomem 70% de sua potência e os que trabalham em potência máxima consomem 100%; *Threshold* (limiar de migração de cargas virtuais tentando manter as entidades de processamento dos servidores com no mínimo 80% de atividade); e *Fan Control* (com consumo linear de potência proporcional ao DVFS).

5.7. Resultados das Simulações

Trinta simulações foram realizadas para cada algoritmo, para cada uma das configurações mencionadas na Seção 5.3. As siglas usadas nesta seção são as mesmas apresentadas ao final da Seção 5.1. O resultado desejável é a redução nos tempos de execução das cargas com alta prioridade sem um aumento substancial do consumo de energia, assim como os tempos de execução das cargas de baixas prioridades também não devem ser inaceitavelmente acentuados.

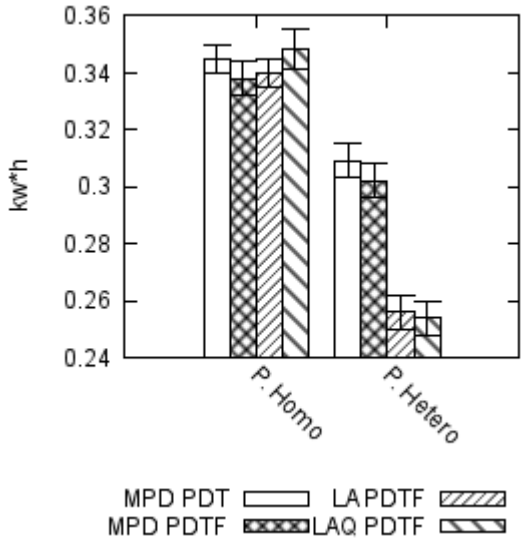

Figura 1: Consumo de Energia entre MPD PDT, MPD PDTF, LA PDTF, LAQ PDTF em Data Centers Pequenos

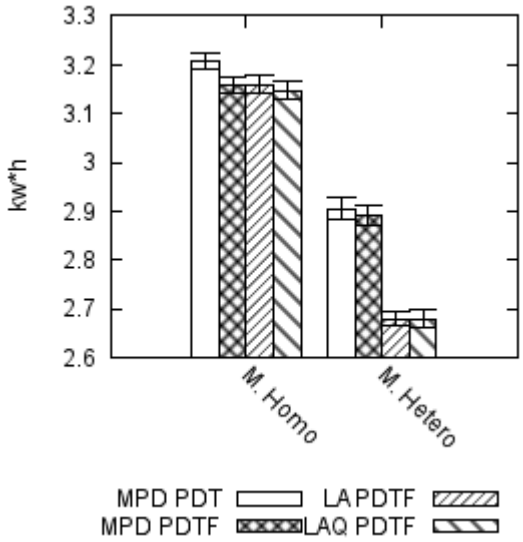

Figura 2: Consumo de Energia entre MPD PDT, MPD PDTF, LA PDTF, LAQ PDTF em Data Centers Médios

Apresentamos nas Figuras 1, 2 e 3 os resultados de consumo de energia dos algoritmos que apresentaram os melhores resultados nas simulações (com intervalo de confiança de 95%). Contemplamos o melhor algoritmo encontrado, o MPD com configurações PDT (MPD PDT); o MPD acrescido do que foi proposto em nosso trabalho anterior, o *Fan Control* (MPD PDTF); o *Lago Allocator* (LA PDTF); e o *Lago Allocator* com Qualidade de Serviço (LAQ PDTF) que é o algoritmo proposto neste trabalho.

De modo geral, os resultados das simulações não cientes de energia consumiram aproximadamente o dobro da energia do pior caso das simulações cientes de energia e, por isso, não foram demonstradas graficamente – são inquestionavelmente a pior solução

do ponto de vista do consumo de energia.

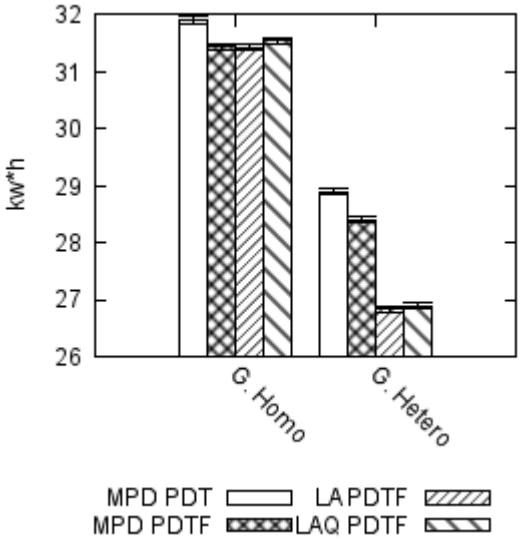

Figura 3: Consumo de Energia entre MPD PDT, MPD PDTF, LA PDTF, LAQ PDTF em Data Centers Grandes

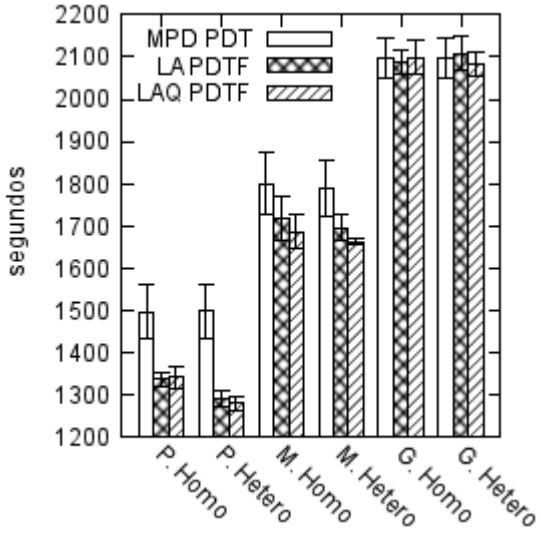

Figura 4: Makespan de MPD PDT, LA PDTF, LAQ PDTF

Observando os gráficos é possível perceber uma superioridade do LA e do LAQ sobre o MPD e praticamente um empate entre o LA e LAQ. Utilizando os resultados mostrados nas Figuras 1, 2 e 3 é possível tirar algumas conclusões acerca dos resultados obtidos, apresentados na Tabela 3.

Tabela 3: Melhores Algoritmos de Alocação na Comparação de Consumo de Energia entre MPD PDT, MPD PDTF, LA PDTF, LAQ PDTF

	P. Homo	***P. Hetero***	***M. Homo***	***M. Hetero***	***G. Homo***	***G. Hetero***
LA PDTF x MPD PDT	Empate	LA PDTF (de 15,5% a 25,9%)	LA PDTF (de 0,4% a 2,7%)	LA PDTF (de 7,0% a 9,9%)	LA PDTF (de 1,1% a 1,9%)	LA PDTF (de 7,3% a 8,1%)
LA PDTF x MPD PDTF	Empate	LA PDTF (de 12,5% a 23,1%)	Empate	LA PDTF (de 6,6% a 9,2%)	Empate	LA PDTF (de 5,4% a 6,3%)
LAQ PDTF x MPD PDT	Empate	LAQ PDTF (de 16,5% a 26,8%)	LAQ PDTF (de 0,8% a 3,0%)	LAQ PDTF (de 6,9% a 9,9%)	LAQ PDTF (de 0,8% a 1,5%)	LAQ PDTF (de 7,1% a 7,9%)
LAQ PDTF x MPD PDTF	Empate	LAQ PDTF (de 13,5% a 24%)	Empate	LAQ PDTF (de 6,5% a 9,3%)	Empate	LAQ PDTF (de 5,3% a 6,1%)
LA PDTF x LAQ PDTF	Empate	Empate	Empate	Empate	LA PDTF (de 0% a 0,8%)	Empate

Era de se esperar que o LAQ, devido à qualidade de serviço que presta, resultasse em um consumo de energia significativamente maior do que o LA. No entanto, para as simulações realizadas, isso não ocorreu. Podemos constatar na Tabela 3 que, de fato, o LAQ foi pior que o LA apenas no caso de *data centers* grandes homogêneos e, mesmo assim, por uma diferença muito pequena.

O LAQ, assim como o LA, foi capaz de superar o MPD com configurações PDTF em todos os *data centers* heterogêneos. Além disso tanto o LAQ quanto o LA se apresentaram melhores quando confrontados com o MPD original (PDT) em todos os *data centers*, exceto no caso pequeno e homogêneo.

Considerando os resultados está claro que o LAQ é tão competitivo quanto o LA em relação ao consumo de energia. Além disso, nas simulações realizadas, o LAQ nunca foi considerado inaceitavelmente mais lento do que qualquer outro algoritmo.

Como era de se esperar, as execuções foram mais rápidas em um ambiente não ciente de energia. Mesmo assim, LAQ PDTF permaneceu em média entre 6,87% (nas simulações realizadas em *data centers* pequenos e heterogêneos) e 37,09% (nas simulações realizadas em *data centers* grandes e homogêneos) mais lento do que o melhor algoritmo em cada uma dessas simulações, respeitando as regras estabelecidas de tempos de execução aceitáveis. Na Figura 4 é apresentado um comparativo dos tempos de execução entre o MPD, o LA original e o LA com qualidade de serviço, adotando intervalo de confiança de 95%.

Com base nos resultados é possível constatar que existe um empate técnico entre o LAQ PDTF e o LA PDTF em todas as simulações realizadas. Esse empate é um bom resultado, tendo em vista que a utilização de qualidade de serviço poderia afetar negativamente o *makespan* das cargas no algoritmo original.

Assim como no caso do LA PDTF, as migrações também impactam significativamente o tempo de execução de todas as tarefas no LAQ PDTF. Em média essas migrações significaram um aumento de tempo consumido de 11,57% para *data centers* pequenos e homogêneos; 6,53% para pequenos e heterogêneos; 36,08% para médios homogêneos; 31,93% para médios heterogêneos; 58,58% para grandes homogêneos; e 60,93% para grandes heterogêneos.

Tabela 4: Comparação de Consumo de Tempo entre LA PDTF e LAQ PDTF para *Cloudlets* (em segundos)

Cenário	*Priorid. Carga*	*LA PDTF (1)*	*LAQ PDTF (2)*	*Melhor*
Pequeno Homo	Baixa	677,97±5,98	892,10±7,62	(1), de 22,67% a 25,31%
	Normal	641,38±4,52	650,65±4,40	(1), de 0,05% a 2,78%
	Alta	649,53±6,27	451,23±4,07	(2), de 29,22% a 31,81%
Pequeno Hetero	Baixa	667,77±5,58	870,17±6,11	(1), de 22,07% a 24,43%
	Normal	632,22±4,68	636,85±3,87	Empate
	Alta	639,10±4,89	440,03±4,66	(2), de 29,88% a 32,39%
Médio Homo	Baixa	671,81±5,42	923,34±6,04	(1), de 26,17% a 28,30%
	Normal	664,35±4,35	641,70±3,08	(2), de 2,31% a 4,50%
	Alta	666,62±6,26	456,06±3,61	(2), de 30,39% a 32,76%
Médio Hetero	Baixa	664,66±3,82	924,66±6,10	(1), de 27,22% a 29,00%
	Normal	664,53±3,36	643,84±2,98	(2), de 2,17% a 4,05%
	Alta	667,59±6,03	451,90±2,15	(2), de 31,37% a 33,24%
Grande Homo	Baixa	682,60±2,10	959,60±3,03	(1), de 28,42% a 29,31%
	Normal	682,02±1,56	654,53±1,49	(2), de 3,59% a 4,47%

Cenário	*Priorid. Carga*	*LA PDTF (1)*	*LAQ PDTF (2)*	*Melhor*
	Alta	682,65±1,55	462,88±1,38	(2), de 31,84% a 32,55%
Grande Hetero	Baixa	686,11±1,98	965,43±2,73	(1), de 28,52% a 29,34%
	Normal	685,18±1,71	654,86±1,64	(2), de 3,95% a 4,90%
	Alta	685,94±1,74	464,05±1,43	(2), de 31,97% a 32,73%

Com relação à qualidade de serviço baseada em prioridades, com o objetivo de minimizar o tempo necessário para executar *cloudlets* de alta prioridade, temos os resultados apresentados na Tabela 4.

Como podemos observar, para os cenários simulados o alocador proposto ao contemplar qualidade de serviço baseada em prioridades foi melhor de 29,22% a 32,73% na execução das cargas com alta prioridade, ou seja, um resultado bastante significativo. Em contrapartida ele foi pior na execução das cargas de baixa prioridade de 22,07% a 29,34%. Para as cargas cuja prioridade foi "normal", o LAQ apresentou alguma melhoria para os casos de *data centers* médios e grandes.

6. Conclusões

Neste artigo trabalhamos no processo de escalonamento de máquinas virtuais e cargas de trabalho na computação em nuvem, adicionando o conceito de qualidade de serviço baseada em prioridades de cargas. Isto foi feito com aderência ao conceito de computação verde, tendo como objetivo minimizar os tempos de execução de cargas com alta prioridade sem impactar negativamente o consumo de energia de *data centers*.

Propusemos um algoritmo simples no processo de submissão de cargas de trabalho para máquinas virtuais, que atua conjuntamente com um algoritmo proposto em trabalho anterior, sendo capaz de realizar essa submissão em *data centers* não-federados, homogêneos e heterogêneos, com o tamanho das cargas de trabalho conhecidas ou não. Este algoritmo lida com qualidade de serviço, ranqueando cargas de trabalho de acordo com suas prioridades durante o processo de escalonamento, permitindo que cargas de alta prioridade sejam executadas mais rapidamente.

Resultados obtidos por simulação sugeriram que com o modelo de gerência de prioridade de cargas adotado neste trabalho, conseguimos reduzir o tempo de processamento de cargas com alta prioridade em aproximadamente 31%. As cargas normais sofreram pouco impacto nas simulações e as de baixa prioridade tiveram um incremento no seu tempo de execução de cerca de 26%.

Em suma pode-se afirmar que a adição de qualidade de serviço no *Lago Allocator* não impacta significativamente o consumo de energia, não modifica o *makespan* das cargas de trabalho, mas reduz de forma substancial o tempo de execução das cargas que possuem prioridades elevadas.

Propomos como trabalho futuro fazer a avaliação prática e de *workloads* específicos como forma de complementar os resultados obtidos neste artigo. Também é nosso objetivo adaptar o algoritmo apresentado para lidar com *data centers* federados. Pretendemos estudar heurísticas, como as aplicáveis ao problema do empacotamento, para aperfeiçoar o algoritmo para o caso de cargas virtuais cujos tamanhos possam ser medidos ou estimados. Além disso podemos também estudar técnicas para otimizar automaticamente o valor do limiar para realização da migração de máquinas virtuais,

tornando a nuvem mais eficiente no consumo de energia.

7. Agradecimentos

Os autores agradecem à FAPESP (2010/14592-9 e 2009/15008-1) e ao CNPq pelo apoio financeiro.

Referências

Beloglazov e Buyya 2010 : BELOGLAZOV, A.; BUYYA R.. Energy Efficient Resource Management in Virtualized Cloud Data Centers. 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. 2010

Binder e Suri 2009 : BINDER, W.; SURI, N.. Green Computing: Energy Consumption optimized service hosting. SOFSEM '09: Proceedings of the 35 Conference on Current Trends in Theory and Practice of Computer Science. 117-128. 2009

Bittencourt e Madeira 2011 : BITTENCOURT, L. F; MADEIRA, E. R. M.. HCOC: A Cost Optimization Algorithm for Workflow Scheduling in Hybrid Clouds. Journal of Internet Services and Applications, Springer London. 207-227. Volume 2. 2011

Buyya et al. 2009 : BUYYA, R.; YEO, C.; VENUGOPAL, S.; BROBERG, J.; BRANDIC, I.. Cloud computing and emerging IT platforms: Vision, hype and reality for delivering computing as the 5th utility. Future Generation Computer Systems 25. 599-616. 2009

CloudSim 2009 : BUYYA, R.; RANJAN, R.; CALHEIROS, R.N.. Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. International Conference on High Performance Computing & Simulation. 2009

Dasgupta et al. 2011 : DASGUPTA, G.; SHARMA, A.; VERMA, A.; NEOGI, A.; KOTHARI, R.. Workload Management for Power Efficiency in Virtualized Datacenters. Communications of the ACM. 131-141. Volume 54, Número 7. 2011

Lago et al. 2011 : LAGO, D. G., MADEIRA, E. R. M., BITTENCOURT, L. F.. Power-Aware Virtual Machine Scheduling on Clouds Using Active Cooling Control and DVFS. 9th International Workshop on Middleware for Grids, Clouds and e-Science - MGC 2011. 2011

Rabaey 1996 : RABAEY, J. M.. Digital Integrated Circuits. Prentice Hall. 1996

Srikantaiah et al. 2008 : SRIKANTAIAH, S.; KANSAL A.; ZHAO F.. Energy Aware Consolidation for Cloud Computing. Microsoft Research. 2008

Voorsluys et al. 2009 : VOORSLUYS, W.; BROBERG, J.; VENUGOPAL, S.; BUYYA, R.. Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation. Cloud Computing Lecture Notes in Computer Science. Volume 5931. 2009

Xu et al. 2009 : XU, M.; CUI, L.; WANG, H.; BI, Y.. A Multiple QoS Constrained Scheduling Strategy of Multiple Workflows for Cloud Computing. 2009 IEEE International Symposium on Parallel and Distributed Processing with Applications. 629-634. 2009