Heap Allocator: Uma Polıtica de Alocação de Máquinas Virtuais em Ambiente de Computação em Nuvens baseada em Heap com Prioridade

Matheus Magalhães de Carvalho[1], **Francisco Mardônio Vieira Filho**[1], **Filipe Fernandes S B de Matos**[2], **Rafael Lopes Gomes**[1], **Joaquim Celestino Júnior**[1]

[1]Universidade Estadual do Ceará (UECE)
Av. Dr Silas Munguba, 1700, Fortaleza-CE

[2]Universidade Federal do Ceará (UFC)
BR 226, KM 4 - Venâncios, Crateús-CE

{matheus.magalhaes,celestino}@larces.uece.br

{mardoniovf,rafaellgom}@gmail.com,filipe.fernandes@crateus.ufc.br

Abstract. *Cloud Computing represents a paradigm that provides computing resources on virtual machines that are grouped and allocated according to customer requests. The increasing seeking for this type of service caused an increasing demand for electric energy. Several types of research aim to meet the demand for resources in Cloud Computing, and at the same time, reduce the energy consumption in the data center. Within this context, this paper presents a virtual machine allocation policy called Heap Allocator, which mitigates the power consumption of the data center. A comparative analysis of the proposed solution with other existing mechanisms showed the proposed solution reduces the data center power consumption by approximately* 1.5%*.*

Resumo. *Computação em Nuvens representa um paradigma computacional onde recursos físicos são agrupados e alocados em máquinas virtuais de acordo com as solicitações dos clientes. A crescente busca por esse tipo de solução ocasionou uma maior demanda por energia elétrica. Diversas pesquisas visam atender a contínua demanda de recursos em Computação em Nuvens, e ao mesmo tempo, melhorar o consumo energético no data center. Dentro deste contexto, este artigo apresenta uma Política de Alocação de Máquinas Virtuais chamada* Heap Allocator*, que busca mitigar o consumo de energia de um data center. Uma análise comparativa da proposta com outras políticas existentes mostrou uma redução do consumo de energia do data center em aproximadamente* $1,5\%$*.*

1. Introdução

Atualmente, muitos sistemas computacionais e sistemas de informação estão substituindo seu modelo tradicional de computação por um novo conhecido como Computação em Nuvens. O modelo tradicional compreende a divisão das máquinas entre clientes e servidoras. Os servidores possuem grande capacidade de processamento e armazenamento e disponibilizam seus recursos e serviços às máquinas clientes através de uma rede como,

por exemplo, a Internet. O papel das máquinas clientes é interagir com os usuários e consumir tudo aquilo que é ofertado pelos servidores. Um *data center* representa um conjunto de servidores e de equipamentos de rede, que interliga esses servidores e os conectam aos usuários através da Internet [Rong et al. 2016]. Um problema decorrente do modelo tradicional de computação é o desperdício de recursos pela infraestrutura do *data center*. Neste modelo, cada servidor hospeda uma única aplicação ou serviço que, devido a variação constante de demanda, pode, muitas vezes, subutilizar os recursos computacionais da máquina em questão. Este sub-uso pode levar a uma ociosidade dos recursos e ocasionar gastos e desperdícios para o *data center*.

Para solucionar esse problema, o modelo de Computação em Nuvens emprega a técnica da Virtualização. Tal mecanismo consiste na virtualização dos recursos físicos do *hardware* e os oferece como recursos virtuais às aplicações de alto nível [Zhang et al. 2010]. Dessa forma, é possível implementar várias aplicações dentro de um mesmo servidor físico. Cada aplicação é inserida em uma Máquina Virtual (VM) para a execução. Uma VM consiste de um subconjunto de recursos virtuais do servidor físico de forma que não exista sobreposição entre elas.

A crescente procura por serviços em Nuvem motivou o aumento no número e no tamanho dos *data centers*. Um dos problemas relacionados a essa expansão é o alto consumo de energia do *data center*. Para se ter uma idéia, em 2013, segundo [Delforge 2014], estima-se que apenas os *data centers* dos Estados Unidos consumiram 91 bilhões kilowatts-hora de eletricidade e que tal valor seria suficiente para abastecer todas as residências da cidade de Nova York por dois anos seguidos. Outros estudos revelam que o índice de crescimento no consumo energético dos *data centers* é de 18% ao ano e que, até 2020, eles serão os responsáveis por 2% do consumo mundial de energia [Sharma and Reddy 2015]. Um estudo mais recente da Forbes [Forbes 2017] aponta que todos os *data centers* do planeta consumiram cerca de 416 terawatts de energia. Tal valor representa cerca de 40% a mais do consumo energético de todo o Reino Unido.

O presente trabalho propõe uma nova Política de Alocação de Máquinas Virtuais com o objetivo de reduzir o consumo de energia elétrica de um *data center* em ambiente de Computação em Nuvens. A política em questão utilizará uma estrutura de dados conhecida como MaxHeap para definir o servidor com a maior eficiência energética e escolhê-lo para receber uma VM em processo de alocação.

O restante deste trabalho está organizado da seguinte forma: A Seção 2 abordará os principais trabalhos relacionados com a técnica aqui proposta. A Seção 3 apresentará os principais conceitos associados a estrutura de dados Heap. A Seção 4 descreverá a proposta deste trabalho: a Política de Alocação HeapAllocator. A Seção 5 apresentará as configurações do cenário utilizado na simulação e as métricas usadas para a avaliação de desempenho das Políticas. A Seção 7 apresentará as conclusões obtidas e possíveis trabalhos futuros.

2. Trabalhos relacionados

Em [Beloglazov and Buyya 2012], o problema de alocação de máquinas virtuais, considerando o consumo de energia, é dividido em duas partes: 1) Escolha do servidor que receberá a VM; 2) Otimização da localização das VMs alocadas. No primeiro caso, é realizada a modelagem como um problema do empacotamento e é utilizado o algoritmo

Modified Best Fit Decreasing. Nesse algoritmo, a lista de máquinas virtuais a serem alocadas são ordenadas em ordem decrescente de acordo com a taxa de utilização de CPU. Em seguida, cada máquina virtual é alocada no servidor no qual haverá o menor aumento de consumo de energia após a alocação. No segundo caso são utilizadas políticas de migração de VMs onde são estabelecidos o limite inferior e o superior de utilização dos recursos de um servidor. O objetivo é manter todos os servidores ativos dentro da faixa estabelecida pela política. Apesar de apresentar bons resultados para os cenários testados, essa abordagem não diferencia os servidores em relação a sua capacidade de processamento e seu consumo de energia. Desse modo, em um cenário heterogêneo o algoritmo pode consolidar as máquinas virtuais em um servidor onde a eficiência energética é baixa, ocasionando um consumo de energia maior do que o esperado.

Em [Nasim et al. 2016], os autores apresentam uma técnica para realizar a alocação de máquinas virtuais considerando a energia elétrica consumida e os recursos utilizados por elas: processamento, memória principal e memória secundária. Sua estratégia aloca máquinas virtuais que necessitam intensamente de um mesmo tipo de recurso em servidores diferentes. O autor recorreu ao conceito de sensibilidade formulado por [Taheri et al. 2017]. A sensibilidade de uma máquina virtual por um recurso computacional é diretamente proporcional ao nível que ela necessita desse recurso, ou seja, o quanto seu desempenho é degradado caso esse recurso lhe seja disponibilizado de maneira insuficiente. Os autores afirmam que essa técnica não é uma heurística rápida, por isso realizaram os experimentos com pequenas instâncias para o problema de alocação de máquinas virtuais, especificamente com 45 máquinas virtuais e 20 máquinas físicas. Todas as máquinas físicas possuem a mesma configuração, o que representaria um data center homogêneo.

Em [Nasim and Kassler 2017], os autores acreditam que para realizar uma consolidação energeticamente eficiente deve-se conhecer as demandas de cada máquina virtual por recursos computacionais. Eles afirmam também que esses dados são difíceis de se obter, uma vez que a carga de trabalho dos serviços em execução nas máquinas virtuais sofrem variação com um curto intervalo de tempo. Uma estratégia robusta de alocação deve realizar a consolidação de forma que o servidores suportem tal variação sem violar demasiadamente o SLA. A heurística utilizada foi baseada na Busca Tabu (TS). O autor acrescentou em sua estratégia ideias de Otimização Robusta, que trabalha com problemas onde não se tem dados precisos, mas que podem ser limitados. Para isso a função de avaliação foi modificada adicionado um ruído R na solução atual, onde R representa as alterações esperadas nos dados de entrada. Ela observa quanto as máquinas virtuais podem variar para continuar atendendo a restrição de capacidade máxima definida por sua política. Observe que se a variação for maior que o limite definido, a política terá seu desempenho degradado.

Em [Okada et al. 2015], são propostos três algoritmos para a alocação de máquinas virtuais em um *data center*: um baseado no *First Fit Decreasing (FFD)* [Yue 1991] e outros dois derivados do *Power Aware Best Fit Decreasing* (PABFD) [Beloglazov 2013]. O FFD opera alocando VMs em servidores com base em uma lista ordenada (decrescente) de uso de CPU. Primeiramente, a lista de VMs é ordenada de forma descrescente com base no uso de CPU. Posteriormente, tal lista é percorrida, bem como a lista de servidores disponíveis. Cada VM é alocada no primeiro servidor com

recursos suficientes para armazená-la. A contribuição do trabalho de [Okada et al. 2015] é suspender todos os servidores que não receberam nenhuma máquina virtual durante o processo inicial de alocação.

O PABFD, por sua vez, trabalha com uma estimativa da energia antes da alocação efetiva da VM em um servidor. Assim, como no FFD, inicialmente, a lista de VMs é ordenada de forma decrescente de acordo com a utilização de CPU. As listas de VMs e de servidores também são percorridas. Entretanto, ao invés de alocar a VM no primeiro servidores com capacidade para armazená-la, é feito uma análise e é escolhido o servidor que sofrerá o menor aumento de energia após a alocação. Além da suspensão dos servidores que não possuem máquinas virtuais, a outra contribuição de [Okada et al. 2015] foi criar o GPABFD (*Global Power Aware Best Fit Decreasing*). O GPABFD é uma modificação na função que estima o aumento energértico da VM a ser alocado em um servidor. Ao contrário da proposta original, o GPABFD avalia o efeito energético em uma escala "global", ou seja, em relação a todo *data center*. Dessa maneira, a alocação é feita visando reduzir o consumo energético de todo o *data center* e não apenas do servidor local.

O primeiro problema de ambas as abordagens é que a eficiência energética dos servidores durante o processo de alocação não é analisada, o que pode ocasionar à uma resposta não ótima do problema, mesmo com a modificação sugerida pelo GPABFD que não utiliza a eficiência energética para escolher o melhor servidor para uma VM. Em segundo lugar, devido à natureza dinâmica de um *data center*, ordenar uma lista de VMs é algo difícil de ser realizado, pois tal lista estará sempre em constante modificação, devido à chegada ou eventuais saídas de VMs do *data center*. O *HeapAllocator* busca mitigar esses dois problemas.

3. Fundamentação Teórica

Uma *Heap* é uma estrutura de dados que permite a implementação de uma fila de prioridades [Lafore 2017]. Uma fila de prioridade é um tipo abstrato de dados que armazena uma coleção de elementos organizados de acordo com uma prioridade [Goodrich and Tamassia 2013]. Em uma fila de prioridade, os elementos são armazenados de maneira ordenada baseado nos seus níveis de prioridades.

A fila de prioridade pode ser classificada como de máximo ou de mínimo. A fila de máximo é aquela cujo a prioridade de cada pai é maior ou igual que qualquer um de seus filhos. Já no caso da fila de mínimo, ocorre o contrário, ou seja, a prioridade de cada pai é menor ou igual que qualquer de seus filhos. Dessa maneira, o primeiro elemento da fila será o maior e o menor dos valores da estrutura (para as filas de máximo e de mínimo, respectivamente).

Os dois principais métodos em uma fila de prioridade são: inserção e remoção de elementos. A operação de inserção consiste em adicionar um elemento na fila de prioridade, mantendo a ordenação dos níveis de prioridade. A operação de remoção remove e retorna o elemento com maior ou menor prioridade, de acordo com o tipo da estrutura implementada.

A estrutura *Heap* pode ser visualizada como uma árvore binária e representada através de um vetor. Cada elemento do vetor corresponde a um nó da árvore binária e o nó raiz é o maior ou o menor elemento de acordo com o nível de prioridade estabelecido.

Uma *Heap* implementa os métodos de inserção e remoção de uma fila de prioridades em tempo logarítmico [Cormen 2009]. No método de inserção da *Heap*, o novo elemento é sempre inserido no final da estrutura. Após isso, são realizadas comparações entre a prioridade do novo elemento com as prioridades dos elementos já inseridos na estrutura. Considerando uma *Heap* de prioridade máxima, então nesse caso, se o elemento possuir uma prioridade maior do que seu pai, eles serão trocados de posição. Esse comportamento é mantido até que a prioridade do nó pai seja maior do que a prioridade do novo elemento.

Já no procedimento de remoção, o topo da estrutura, que pode conter ou o maior ou o menor elemento de acordo com a prioridade previamente estabelecida, é sempre retirado. Após a remoção, a estrutura se reorganiza através de comparações. Ao remover o topo da *Heap*, o último elemento da estrutura é posto no lugar do elemento excluído. Assim, o novo topo será comparado com os seus nós filhos. Nesse caso, o elemento será permutado com o filho de maior prioridade. Esse comportamento é mantido até que os nós filhos do elemento tenham prioridades menores do que ele.

4. A Política Heap Allocator

A Política *HeapAllocator* utiliza uma estrutura *Heap* com prioridade máxima, cujo objetivo principal é reduzir o consumo de energia elétrica de um *data center*. Uma vantagem da *Heap* com prioridade é que suas operações de inserção e remoção possuem uma complexidade computacional na ordem $O(nlogn)$. Assim, espera-se que a *HeapAllocator* seja computacionalmente rápida. A *HeapAllocator* é uma Política de Alocação monocritério, sendo que neste trabalho, é considerada apenas a CPU (em MIPS) como critério para determinar se um servidor físico é capaz ou não de receber uma máquina virtual.

A estrutura *Heap* da *HeapAllocator* é responsável por ordenar informações de servidores que já possuem máquinas virtuais. A estrutura em *Heap* é ordenada de acordo com a métrica de eficiência energética (*M*) definida na Fórmula 1. Tal métrica é o resultado da divisão entre a eficiência energética do servidor em análise (E_f) e a taxa de CPU ociosa no mesmo servidor ($MIPS_{disp}$) naquele instante.

$$M = \frac{E_f}{MIPS_{disp}} \tag{1}$$

De acordo com [Lago et al. 2012], a eficiência energética (E_f) de um servidor é obtida através da divisão entre a quantidade máxima de MIPS por ele disponibilizada ($MIPS_{max}$) e a quantidade máxima de energia consumida por esse mesmo servidor (E_{cons}). A fórmula da eficiência energética pode ser visualizada na Fórmula 2.

$$E_f = \frac{MIPS_{max}}{E_{cons}} \tag{2}$$

Como o valor da métrica é diretamente proporcional à eficiência energética, quanto maior o seu valor, mais apropriada é a alocação de uma máquina virtual no servidor em questão. Dessa maneira, em uma *Heap* com prioridade máxima, o melhor candidato para receber uma nova máquina virtual, em termos de eficiência energética, estará sempre no topo da estrutura.

Os passos da política de alocação de máquinas virtuais *HeapAllocator* são descritos em Algoritmo 1.

Algoritmo 1: Política Heap Allocator

```
   Entrada: Lista de VMs, Lista de Hosts Vazios e Heap de Hosts
   Saída: VMs alocadas nos servidores
 1 início
 2     enquanto Lista de VMs não é vazia faça
 3         VM atual recebe a primeira VM da Lista de VMs;
 4         enquanto heap não está vazia faça
 5             Host atual recebe o host do topo da heap;
 6             se Host atual possui CPU disponível então
 7                 Alocar VM atual no Host atual selecionado;
 8                 Retirar VM atual da Lista de VMs;
 9                 Atualizar Host atual na heap com prioridade;
10             fim
11             senão
12                 Retirar topo da heap;
13             fim
14         fim
15         se heap está vazia então
16             Retirar Host atual da Lista de Host Vazios;
17             Alocar VM atual no Host atual selecionado;
18             Inserir Host atual na heap com prioridade;
19             Retirar VM atual da Lista de VMs;
20         fim
21     fim
22 fim
```

O algoritmo recebe três estruturas como entradas: 1) Uma lista com máquinas virtuais passíveis de alocação no servidor (*Lista de VMs*); 2) Uma lista, ordenada com base na eficiência energética (E_f), com todos os servidores vazios, ou seja, sem nenhuma máquina virtual alocada (*Lista de Hosts Vazios*); 3) a *Heap de Hosts* com todos os servidores ativos, ou seja, que possuem ao menos uma máquina virtual alocada.

O algoritmo ainda possui duas variáveis relacionadas ao processo de decisão sobre a alocação de uma máquina virtual em um servidor. A variável *Host atual* refere-se a um servidor candidato a receber uma das máquinas virtuais. Já a variável *VM atual* está relacionada a uma máquina virtual da *Lista de VMs* que está em processo de alocação.

O objetivo do algoritmo é primeiro tentar alocar a *VM atual* em servidores ativos com base na ordem definida pela *Heap de Hosts* (Linhas 4-14). Caso não encontre algum apto a recebê-la, o algoritmo deverá alocá-la no primeiro servidor da *Lista de Hosts Vazios* (Linhas 15-20), ou seja, no servidor inativo que apresenta a maior eficiência energética. Tal comportamento é motivado pela tentativa da Política em procurar, inicialmente, consolidar as máquinas virtuais nos servidores ativos que se encontram na *Heap*. Esta consolidação possibilita a economia de energia, pois todos os servidores que estão

vazios podem ser postos em um modo de economia de energia, por exemplo, em modo de hibernação.

O primeiro passo do algoritmo é verificar se a *Lista de VMs* não está vazia (Linha 2), ou seja, se existem máquinas virtuais a serem alocadas nos servidores. Caso positivo, retira-se a primeira máquina virtual da lista e a armazena em *VM atual* (Linha 3). Feito isso, é verificado se a *Heap* também não está vazia (Linha 4). Caso positivo, o servidor no topo da *Heap* é escolhido e armazenado na variável *Host atual* (Linha 5). Na Linha 6, é verificado se o servidor tem capacidade de receber a máquina virtual em análise. Caso positivo, a máquina virtual é alocada (Linha 7) e a *Heap* atualizada (Linha 9). Caso negativo, retira-se o topo da *Heap* e continua-se a busca por um servidor com capacidade suficiente para alocar a máquina virtual (Linha 12). Esta busca acontece sempre obedecendo a ordem definida pela *Heap*.

Caso a *heap* esteja vazia (Linha 15), o *HeapAllocator* tem de escolher um servidor que ainda não possui nenhuma máquina virtual. Assim, a variável *Host atual* é preenchida com o primeiro servidor com capacidade para receber a máquina virtual (Linha 16), a *VM atual* é alocada no *Host atual* (Linha 17) e o servidor é inserido na *Heap* (Linha 18). O processo continua até que todas as VMs sejam alocadas em um servidor (Saída do algoritmo).

5. Cenário de Simulação

Neste trabalho, foi utilizado o simulador CloudSim [Calheiros et al. 2011] pelo fato de já possuir algumas Políticas de Alocação de VMs implementadas em sua versão nativa, além de sua ampla aceitação pela comunidade acadêmica.

Com o objetivo de aproximar os ambientes simulados a um *data center* real, buscou-se projetar um cenário com servidores e máquinas virtuais com configurações utilizadas em um ambiente real e tarefas com perfis de consumo de recursos físicos equivalentes à realidade. Para este trabalho foi utilizado o cenário proposto em [Beloglazov and Buyya 2012], pois atende ao perfil desejado de cenário e possui um conjunto de Políticas de Alocação que serão usadas na validação dessa proposta.

O *data center* simulado é composto por 800 servidores, sendo metade deles HP ProLiant G4 e a outra metade HP ProLiant G5. Os dois tipos de servidores são constituídos por um processador de dois núcleos, 4 gigabytes de memória RAM, 1 Gbit/s de largura de banda e 1 terabyte de armazenamento secundário. Quanto a capacidade de processamento, o HP ProLiant41 G4 possui 1860 MIPS por núcleo e o HP ProLiant G5 tem 2660 MIPS por núcleo.

A Tabela 1 define o modelo de consumo de energia dos servidores supracitados. O consumo de energia de cada servidor é calculado em função do seu nível de carga de trabalho para atender as máquinas virtuais em um determinado tempo. Os percentuais intermediários são obtidos através da interpolação dos valores piso e teto mais próximos ao percentual em análise.

As configurações das máquinas virtuais também seguem modelos reais comercializados. A empresa Amazon disponibiliza quatro instâncias de máquinas virtuais: Microinstância (*Micro Instance*), Instância Pequena de propósito geral (*Small Instance*), Instância Extra Grande de propósito geral (*Extra Large Instance*) e Instância Média Oti-

mizada para computação (*High-CPU Medium Instance*). Tais instâncias serão retratadas na simulação com as configurações descritas na Tabela 2.

Um componente importante para a simulação de uma tarefa no CloudSim é a Cloudlet. Uma Cloudlet é responsável pelo gerenciamento dos recursos necessários para a execução de uma tarefa entregue à ela. Para o cenário proposto por [Beloglazov and Buyya 2012], foi utilizado o projeto CoMon [Park and Pai 2006] com dados sobre o consumo de CPU de máquinas virtuais em dez dias aleatórios entre Março e Abril de 2011. Cada histórico de consumo da CPU coletado deverá ser entregue a uma Cloudlet, a qual será atribuída a uma única máquina virtual para processamento.

O CloudSim foi configurado para simular cada um dos dez dias. O projeto CoMon colheu os dados de consumo de CPU das máquinas virtuais a cada 5 minutos. Portanto, na simulação as máquinas virtuais terão uma variação no consumo de CPU a cada 5 minutos em todos os 10 dias.

Tabela 1. Padrão do consumo de energia dos servidores

Servidor	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
HP G4 (W)	86	89.4	92.6	96	99.5	102	106	108	112	114	117
HP G5 (W)	93.7	97	101	105	110	116	121	125	129	133	135

Tabela 2. Resumo das configurações das VMs

Recurso	High-CPU Medium	Extra Large	Small	Micro
Número de núcleos	1	1	1	1
Processamento	2500 MIPS	2000 MIPS	1000 MIPS	500 MIPS
Memória Principal	870 MB	1.7 GB	1.7 GB	613 MB
Tamanho da VM	2.5 GB	2.5 GB	2.5 GB	2.5 GB
Lagura de Banda	10 Mbit/s	10 Mbit/s	10 Mbit/s	10 Mbit/s
Número de VMs	Nº de Cloudlets			

Visando comparar os resultados obtidos neste trabalho, foram utilizadas as políticas de alocação de máquinas virtuais definidas em [Buyya et al. 2010] e [Beloglazov and Buyya 2012]:

1. ***Interquatile Range* (IQR):** Determina o limiar máximo do estado do servidor com base na diferença entre o primeiro e o terceiro quartil do histórico de consumo de CPU do referido servidor [Jain 1991];
2. ***Local Regression* (LR):** Determina uma linha de tendência com base no histórico de consumo de CPU e procura estimar o valor da próxima observação. O servidor é considerado sobrecarregado se tal estimativa satisfazer uma inequação presente em [Beloglazov and Buyya 2012]

3. ***Local Robust Regression* (LRR):** Semelhante à política anterior, mas os termos da inequação são obtidos através de técnicas um pouco mais robustas. A LRR tem como objetivo mitigar o problema que os outliers trazem para a política LR [Beloglazov and Buyya 2012].
4. ***Median Absolute Deviation* (MAD):** Determina o limite máximo do estado do servidor com base na mediana dos desvios absolutos de cada dado obtido do histórico de consumo de CPU do servidor e da mediana deste conjunto [Jain 1991];
5. ***Static Threshold* (THR):** Define estaticamente o limite superior de utilização para servidores e mantém a utilização total da CPU abaixo desse limiar.

Para realizar a simulação, é necessário a presença de políticas de seleção de VM. Quando um servidor está sobrecarregado, uma política de seleção determina quais entre suas máquinas virtuais devem ser migradas para outro servidor, com o intuito de aliviar a carga de trabalho do servidor em questão. Após uma análise dos resultados apresentados em [Beloglazov and Buyya 2012], a política de Máxima Correlação (MC) foi escolhida para os testes. A política de seleção MC possui como base a ideia proposta por [Verma et al. 2009], onde a VM que possui maior correlação de utilização CPU em relação às outras VMs é selecionada para ser migrada. Tal correlação é obtida com a aplicação da técnica de Regressão Linear Múltipla (RLM) e sua qualidade mensurada através do Coeficiente de Regressão.

Para validar a eficiência da política de alocação *HeapAllocator*, foram adotadas quatro métricas baseadas no trabalho em [Beloglazov and Buyya 2012].

5.1. Energia total consumida

A referida métrica calcula a quantidade de energia consumida pelo *data center* ao final de cada dia simulado. O CloudSim calcula a energia consumida por cada servidor com base na sua utilização de CPU em um determinado instante de tempo, segundo definido na Tabela 1. Ao final de cada dia simulado, o próprio CloudSim soma a quantidade total de energia consumida por cada servidor e obtém o gasto total do *data center* (referente apenas aos servidores e não a toda a infraestrutura de TI).

5.2. Número total de migrações

Esta métrica registra a quantidade de migrações de máquinas virtuais realizadas pelo *data center* ao longo do dia simulado. Dessa forma, a cada migração realizada, o CloudSim incrementa uma variável referente ao número de migrações. Ao final do dia, o conteúdo da variável é apresentado.

5.3. *Performance degradation due to migration* (PDM)

Para cada máquina virtual em processo de migração, a quantidade de CPU atribuída a ela sofre uma degradação de 10% pelo próprio simulador CloudSim. Tal penalidade tem como finalidade simular o tratamento de eventos relacionados ao procedimento de migração da máquina virtual. De acordo com [Beloglazov and Buyya 2012], essa degradação é representada como uma violação de SLA, pois uma parte da quantidade de CPU da máquina virtual não está sendo utilizada por ela. A métrica PDM calcula o percetual médio de MIPS indisponível às máquinas virtuais durante a migração e é definida pela Fórmula 3, onde M é o número de VMs, C_{d_j} é a quantidade de processamento

perdido devido a migração pela VM *j*, e C_{r_j}, a quantidade de processamento exigido durante a migração da VM *j*.

$$PDM = \frac{1}{M} \sum_{j=1}^{M} \frac{C_{d_j}}{C_{r_j}} \tag{3}$$

5.4. *SLA violation time per active host* (SLATAH)

Nos cenários de Computação em Nuvens, as máquinas virtuais podem utilizar somente a quantidade de *hardware* atribuída a elas para executar suas tarefas. No trabalho de [Beloglazov and Buyya 2012], todos os casos onde um servidor sobrecarregado está com a utilização de CPU em 100% são considerados como violação de SLA, uma vez que servidores nesse estado não conseguem oferecer mais unidades de processamento às máquinas virtuais neles hospedadas. A métrica SLATAH calcula o percetual médio do tempo em que os servidores estavam nessa situação (consumindo toda a CPU disponível) enquanto ativos. A métrica SLATAH é definida na Fórmula 4, onde N é o número de servidores, T_{s_i} é a quantidade de tempo onde o servidor *i* consumiu 100% do seu CPU, e T_{a_i}, a quantidade de tempo onde o servidor *i* estava ativo.

$$SLATAH = \frac{1}{N} \sum_{i=1}^{N} \frac{T_{s_i}}{T_{a_i}} \tag{4}$$

6. Resultados

Os resultados foram obtidos através da análise do dados referentes as métricas de consumo total de energia, número de migrações, SLATAH e PDM. Estas métricas foram descritas nas seções anteriores. O resultado de cada política, em cada métrica, foi obtido através da mediana dos valores encontrados nos dez dias de simulação.

Quanto ao consumo de energia, a política *HeapAllocator* (H) alcançou os melhores resultados quando comparados aos resultados das propostas sugeridas por [Buyya et al. 2010] and [Beloglazov and Buyya 2012]. Como pode ser visualizado na Figura 1, a política HeapAllocator (H), proposta nesse trabalho, foi a primeira colocada e apresentou uma economia de aproximadamente 21% em relação a última colocada (a política THR) e 1% em relação as segundas colocadas (as políticas LR e LRR). Esse comportamento se deve a estrutura de Heap proposta pela *HeapAllocator* que ordena os servidores com base na sua eficência energética. Assim, a *HeapAllocator* escolhe sempre os servidores mais eficientes em termos energéticos e isso reflete nos resultados dessa métrica. A tentativa de alocar a maior quantidade de máquinas virtuais por servidor (consolidação) também contribui para este resultado positivo.

Em relação à quantidade de migrações, mais uma vez, a política *HeapAllocator* obteve bons resultados. Como pode ser visto na Figura 2, a *HeapAllocator* foi mais uma vez a primeira colocada e apresentou uma redução de 4% em relação a última colocada (a política THR) e obteve redução de 2% comparada ao da segunda colocada (a política IQR). Tal comportamento se deve à consolidação satisfatória de máquinas virtuais proposta pela *HeapAllocator* que minimiza a necessidade de realizar migrações de máquinas virtuais.

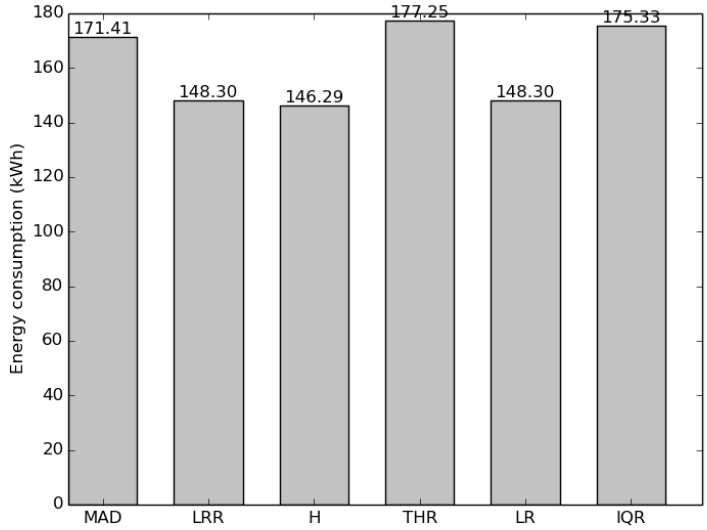

Figura 1. Resultados obtidos com a métrica consumo de energia

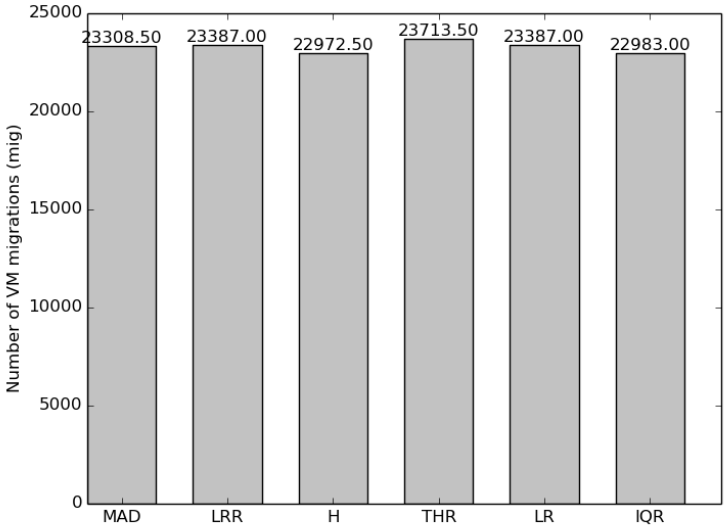

Figura 2. Resultados obtidos com a métrica número de migrações

Os resultados gerados pela métrica PDM foram os mesmos em relação a todas as políticas quando utilizadas em conjunto com a política de seleção MC (Figura 3). A política de seleção MC exige uma quantidade de tempo maior para verificar e retornar se um servidor está sobrecarregado ou não em relação às demais políticas de seleção, o que ocasiona um atraso maior para liberação de recursos que estão sendo utilizados pelas VMs que precisam realizar migração.

Em relação a métrica SLATAH, a política *HeapAllocator* apresentou resultados não satisfatórios em comparação com as demais propostas (Figura 4). A política *HeapAllocator* apresentou um dos piores resultados em relação às outras políticas de alocação. A

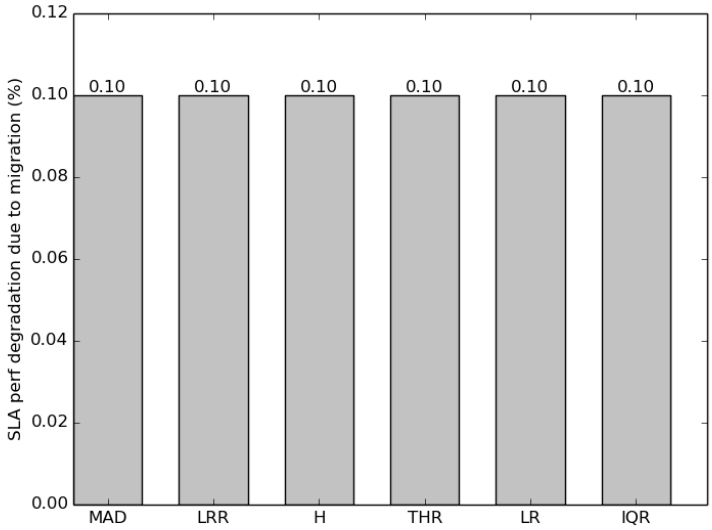

Figura 3. Resultados obtidos com a métrica PDM

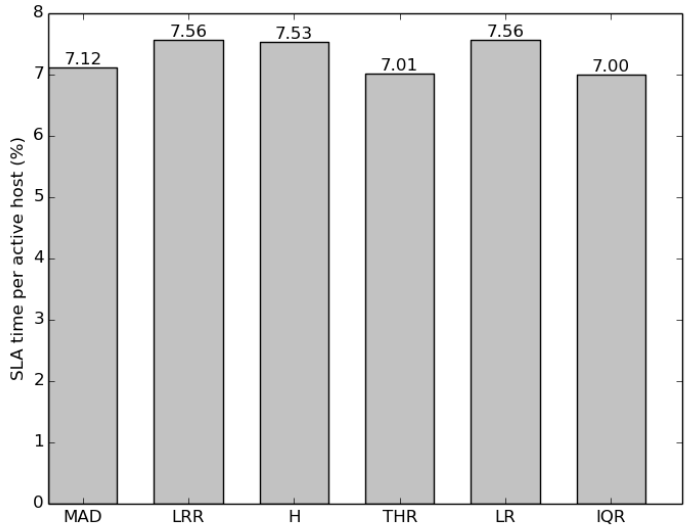

Figura 4. Resultados obtidos com a métrica SLATAH

HeapAllocator apresentou um diferença de aproximadamente 0.5% em relação a primeira colocada (política IQR). Isso é ocasionado pelo longo período necessário para verificação de sobrecarga do servidor, cujo método é importado das políticas de alocação LR e LRR, como explicado na seção anterior. Esse método de verificação de sobrecarga é mais custoso e representa um atraso em políticas relacionadas a regressão linear, deixando o servidor sobrecarregado por mais tempo mais rapidamente até a tomada de decisão, enquanto que a política IQR analisa se um servidor está sobrecarregado mais rapidamente, visto que o método possui complexidade $O(nlogn)$ [Ali and Smith 2006].

7. Conclusão

O principal objetivo deste trabalho foi definir uma política de alocação de máquinas virtuais em ambiente de Computação em Nuvens, denominado *HeapAllocator*, que usa a estrutura de dados *heap* com prioridade para definir o melhor servidor apto a receber uma máquina virtual em processo de alocação ou realocação. O processo de escolha do servidor é realizado visando reduzir o consumo de energia total do *data center*. Esta análise energética representa a contribuição desse trabalho.

A partir dos resultados obtidos, foi constatado que a política *HeapAllocator* alcançou desempenho satisfatório em comparação as outras políticas existentes em relação ao consumo de energia. A *HeapAllocator* conseguiu reduzir o consumo de energia durante a alocação de máquinas virtuais, graças ao mecanismo de tentar, inicialmente, consolidar as máquinas virtuais em poucos servidores e também por buscar usar apenas servidores energeticamente mais eficientes. Os resultados alcançados nas métricas relacionadas ao desempenho (PDM e SLATAH) apresentaram um pequeno acréscimo que indicam um maior número de violações de SLA. A política também reduziu o número total de migrações de máquinas virtuais.

Como trabalhos futuros, é sugerido um estudo para calibrar a fórmula usada na política *HeapAllocator* com o objetivo de reduzir ainda mais o consumo de energia. Além disso, é sugerido um estudo sobre a aplicação da *HeapAllocator* em conjunto com outras políticas de seleção, mas voltadas para a economia de energia também com o objetivo de reduzir ainda mais o consumo de energia. Por fim, é sugerido realizar testes com a política em ambientes reais ou emulados com o intuito de aumentar a credibilidade dos resultados obtidos via simulação.

Referências

Ali, S. and Smith, K. A. (2006). On learning algorithm selection for classification. *Applied Soft Computing*, 6(2):119–138.

Beloglazov, A. (2013). *Energy-efficient management of virtual machines in data centers for cloud computing*. PhD thesis, University of Melbourne, Australia.

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

Buyya, R., Beloglazov, A., and Abawajy, J. H. (2010). Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges. *Computing Research Repository*.

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., and Buyya, R. (2011). Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exper.*, 41(1).

Cormen, T. H. (2009). *Introduction to algorithms*. MIT press.

Delforge, P. (2014). New study: America's data centers consuming – and wasting – growing amounts of energy. Disponível em: https://www.nrdc.org/experts/pierre-delforge/new-study-americas-data-centers-consuming-and-wasting-growing-amounts-energy. Acesso em: 26-09-2018.

Forbes (2017). Why energy is a big and rapidly growing problem for data centers. Disponível em: https://www.forbes.com/sites/forbestechcouncil/2017/12/15/why-energy-is-a-big-and-rapidly-growing-problem-for-data-centers/3531b0835a30. Acesso em: 21-03-2019.

Goodrich, M. and Tamassia, R. (2013). *Estruturas de Dados e Algoritmos em Java - 4.ed.:*. Bookman Editora.

Jain, R. (1991). *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling*. Wiley.

Lafore, R. (2017). *Data structures and algorithms in Java*. Sams publishing.

Lago, D. G., Madeira, E. R., and Bittencourt, L. F. (2012). Escalonamento com prioridade na alocaçao ciente de energia de máquinas virtuais em nuvens. *XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 508–521.

Nasim, R. and Kassler, A. J. (2017). A robust tabu search heuristic for vm consolidation under demand uncertainty in virtualized datacenters. In *Cluster, Cloud and Grid Computing (CCGRID), 2017 17th IEEE/ACM International Symposium on*, pages 170–180. IEEE.

Nasim, R., Taheri, J., and Kassler, A. J. (2016). Optimizing virtual machine consolidation in virtualized datacenters using resource sensitivity. In *Cloud Computing Technology and Science (CloudCom), 2016 IEEE International Conference on*, pages 168–175. IEEE.

Okada, T. K., Vigliotti, A. P. M. D. L. F., Batista, D. M., and vel Lejbman, A. G. (2015). Consolidation of vms to improve energy efficiency in cloud computing environments. pages 150–158.

Park, K. and Pai, V. S. (2006). Comon: A mostly-scalable monitoring system for planetlab. *SIGOPS Oper. Syst. Rev.*, 40(1).

Rong, H., Zhang, H., Xiao, S., Li, C., and Hu, C. (2016). Optimizing energy consumption for data centers. *Renewable and Sustainable Energy Reviews*, 58:674–691.

Sharma, N. K. and Reddy, G. R. M. (2015). Novel energy efficient virtual machine allocation at data center using genetic algorithm. In *Signal Processing, Communication and Networking (ICSCN), 2015 3rd International Conference on*, pages 1–6. IEEE.

Taheri, J., Zomaya, A. Y., and Kassler, A. (2017). vmbbprofiler: a black-box profiling approach to quantify sensitivity of virtual machines to shared cloud resources. *Computing*, 99(12):1149–1177.

Verma, A., Dasgupta, G., Nayak, T. K., De, P., and Kothari, R. (2009). Server workload analysis for power minimization using consolidation. In *Proceedings of the 2009 Conference on USENIX Annual Technical Conference*, San Diego, California.

Yue, M. (1991). A simple proof of the inequality ffd (l) 11/9 opt (l) + 1, l for the ffd bin-packing algorithm. *Acta Mathematicae Applicatae Sinica*, 7(4):321–331.

Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18.