UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE CRATEÚS

CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

FRANCISCO MARDÔNIO VIEIRA FILHO

MSIALLOCATOR: APLICAÇÃO DA META-HEURÍSTICA ILS PARA O PROBLEMA DE ALOCAÇÃO DE MÁQUINAS VIRTUAIS EM UM DATA CENTER

CRATEÚS

2018

FRANCISCO MARDÔNIO VIEIRA FILHO

MSIALLOCATOR: APLICAÇÃO DA META-HEURÍSTICA ILS PARA O PROBLEMA DE ALOCAÇÃO DE MÁQUINAS VIRTUAIS EM UM DATA CENTER

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação do Campus de Crateús da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas de Informação.

Orientador: Prof. Me. Filipe Fernandes S B de Matos

CRATEÚS

2018

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

V715m Vieira Filho, Francisco Mardônio.
MSIAllocator: Aplicação da Meta-Heurística ILS para o problema de alocação de máquinas virtuais em um data center / Francisco Mardônio Vieira Filho. – 2018.
60 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Crateús, Curso de Sistemas de Informação, Crateús, 2018.
Orientação: Prof. Me. Filipe Fernandes S B de Matos.

1. Computação em Nuvens. 2. Data center. 3. Política. 4. Alocação de MáquinasVirtuais. 5. Iterated Local Search. I. Título.

CDD 005

FRANCISCO MARDÔNIO VIEIRA FILHO

MSIALLOCATOR: APLICAÇÃO DA META-HEURÍSTICA ILS PARA O PROBLEMA DE ALOCAÇÃO DE MÁQUINAS VIRTUAIS EM UM DATA CENTER

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação do Campus de Crateús da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas de Informação.

Aprovada em:

BANCA EXAMINADORA

Prof. Me. Filipe Fernandes S B de Matos (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Me. Antônio Emerson Barros Tomaz
Universidade Federal do Ceará (UFC)

Prof. Me. Lisieux Marie Marinho dos Santos Andrade
Universidade Federal do Ceará (UFC)

À minha família, por seu incentivo e investimento em mim. Mãe, seu cuidado e dedicação me deram coragem para enfrentar quaisquer obstáculos. Pai, suas palavras me ensinam a permanecer firme seguindo meus objetivos.

AGRADECIMENTOS

Aos meus pais, irmãos, namorada e amigos, que nos momentos de minha ausência dedicados ao estudo, sempre me fizeram entender que o futuro é feito a partir da constante dedicação no presente.

Ao Prof. Me. Filipe Fernandes dos Santos Brasil de Matos por me orientar (como um pai) em meu primeiro trabalho científico. À sua paciência e tranquilidade, que me trouxeram alívio quando me encontrei aperreado.

Ao Prof. Me. Rennan Ferreira Dantas por sua disposição e entusiasmo ao discutir sobre assuntos referentes a este trabalho, dando sugestões e realizando correções.

A Prof. Me. Lisieux Marie Marinho dos Santos Andrade por ter me dado boas sugestões e ideias sobre meta-heurísticas e me ensinado as regras de como escrever um trabalho científico.

Ao Prof. Me. Luiz Alberto do Carmo Viana por ter estado sempre disposto a conversar sobre ideias e maneiras de como resolver o problema abordado nesse trabalho.

Ao Prof. Me. Antonio Emerson Barros Tomaz pelas sugestões e observações feitas sobre este trabalho, e mesmo distante fisicamente, ter participado da minha banca avaliadora, junto a professora Lisieux e professor Filipe.

A Prof. Me. Amanda Drielly Pires Venceslau por ter acreditado em mim e sido a primeira professora que cativou em mim o desejo de me dedicar mais aos estudos, a partir disso perdi o medo da graduação.

Ao Prof. Me. Allysson Allex de Paulo Araújo por ter me apresentado uma área da computação que se difere de todas as outras e que é tão importante para o desenvolvimento de sistemas.

Ao Prof. Me. José Wellington Franco da Silva por sempre ter sido amigável e tornado a universidade um lugar que vai além do aprendizado de conteúdos e disciplinas.

Ao Prof. Dr. Giannini Italino Alves Vieira por ter me mostrado que com esforço e disciplina que podemos conquistar nossos sonhos.

Ao Prof. Me. Roberto Cabral Rabelo Filho por ter me motivado a escolher fazer cadeiras difíceis como optativas.

Ao Prof. Bruno de Castro Honorato Silva por ter me ensinado com tanta paciência e dedicação os fundamentos da programação, de tal forma que sem esta base eu não teria me desenvolvido no curso.

Ao Prof. Me. Livio Antonio Melo Freire por ter me transmitido conhecimentos valiosos sobre a área da computação, por ter me desafiado a aprender coisas difíceis e por ter me mostrado que mesmo com as raízes que eu possuo, posso chegar onde desejar por meio da educação e preparo.

Ao Prof. Me. André Meireles de Andrade por me ensinar muito sobre as tecnologias usadas no mercado de trabalho, por sempre dizer que devo confiar na minha capacidade e que não devo baixar a cabeça por causa das minhas raízes.

Aos meus colegas de estágio, as experiências e conversas contribuíram tanto com o meu desenvolvimento pessoal quanto com o desenvolvimento deste trabalho.

Agradeço principalmente a Deus, porque me deu a oportunidade de viver muitas experiências e aprender muito sobre um mundo da qual eu não conhecia nada. Durante toda a graduação eu tive certeza de que tudo ia dar certo ao final, mesmo com desafios e prazos apertados, e sempre deu certo. Agradeço por me capacitar e abençoar em todos os momentos, durante esses desafiadores 8 semestres.

”Porque sou eu que conheço os planos que tenho para vocês", diz o Senhor, "planos de fazê-los prosperar e não de lhes causar dano, planos de dar-lhes esperança e um futuro.”

(Jeremias 29:11)

RESUMO

Computação em Nuvens (*Cloud Computing*) é um novo modelo computacional onde os clientes terceirizam recursos computacionais, como capacidade de processamento e armazenamento de dados, os consumindo como serviço através da Internet. Nesse paradigma, os recursos são servidos por meio de um provedor que possui uma ampla infraestrutura física de máquinas servidoras, essa infraestrutura é conhecida por *data center*. Os clientes consomem tais recursos através de máquinas virtuais que são distribuídas entre as inúmeras máquinas servidoras que compõem o *data center*. A crescente utilização desse paradigma ocasionou uma rápida expansão na infraestrutura dos provedores, o que os faz consumir muita energia elétrica. Por conta disso, várias políticas de alocação de máquinas virtuais têm sido propostas com o objetivo de minimizar os gastos energéticos de um *data center* através de uma alocação inteligente de máquinas virtuais em servidores físicos. Entende-se como alocação inteligente a ação de distribuir as máquinas virtuais entre os servidores físicos de tal forma que seja possível gerar economia de energia no *data center*, porém sem degradar o desempenho dos serviços prestados. Este trabalho apresenta uma nova política de alocação, denominada MSIAllocator, que trata o Problema de Alocação de Máquinas Virtuais de forma semelhante ao Problema de Múltiplas Mochilas, utilizando-se da Meta-Heurística *Iterated Local Search* (ILS), a qual gera uma solução aleatória inicial e busca refinar a solução corrente por meio de perturbações e buscas locais. Esta nova política, economizou mais energia elétrica que todas as outras políticas analisadas neste trabalho, consumindo em média 109 kWh por dia, o que representa uma economia de 28% se comparada a política menos econômica. Deixou também a quantidade de migrações e as violações de SLA em níveis aceitáveis, com 21486 e 5,99% por dia, respectivamente.

Palavras-chave: Computação em Nuvens. *Data center*. Política. Alocação de Máquinas Virtuais. *Iterated Local Search*.

ABSTRACT

Cloud Computing is a new computing model where customers outsource computing resources, such as the ability to process and store data, consuming them as a service over the Internet. In this paradigm, the resources are served through a provider that has a large physical infrastructure of server machines, this infrastructure is known as a data center. Customers consume such resources through virtual machines that are distributed among the numerous server machines that make up the data center. The increasing use of this paradigm has caused a rapid expansion in the infrastructure of the providers, which causes them to consume a lot of electric energy. Because of this, several virtual machine allocation policies have been proposed with the goal of minimizing the energy costs of a data center through the intelligent allocation of virtual machines to physical servers. It is understood as intelligent allocation the action of distributing the virtual machines between the physical servers in such a way that it is possible to generate energy savings in the data center, but without degrading the performance of the services provided. This work presents a new allocation policy, called MSIAllocator, which addresses the Virtual Machine Allocation Problem in a similar way to the Multiple Backpack Problem, using the Iterated Local Search Meta-Heuristic (ILS), which generates a random solution the current solution through perturbations and local searches. This new policy saved more electricity than all the other policies analyzed in this study, consuming an average of 109 kWh per day, representing a saving of 28% if compared to the less economic policy. It also left the number of migrations and SLA violations at acceptable levels, with 21486 and 5.99% per day, respectively.

Keywords: Cloud Computing. Data Center. Policy. Allocation of Virtual Machines. Local Search Iterated.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Nos últimos anos, muitas empresas estão substituindo seu modelo tradicional de computação por serviços de nuvem. No modelo tradicional, cada empresa é responsável por construir e manter sua infraestrutura de TI, custeando gastos com a aquisição, a manutenção e o funcionamento (principalmente energia elétrica) dos equipamentos de TI. Os serviços de nuvem são providos através de um modelo computacional chamado de Computação em Nuvens. Diferente do modelo anterior, neste novo paradigma, as empresas terceirizam as ações de construir e manter sua infraestrutura de TI para um Provedor de Serviços em Nuvem e passam a consumir estes recursos de TI em forma de serviço. No modelo em Nuvens as empresas são atraídas pela redução dos custos e pelo aumento da confiabilidade na infraestrutura de TI (MATOS, 2015).

O serviço prestado por um Provedor de Serviços a um cliente (empresa contratante) é formalizado através de um contrato, conhecido como Acordo de Nível de Serviço, ou, simplesmente, SLA (*Service Level Agreement*). Nesse acordo, são definidos os direitos e os deveres dos envolvidos, bem como as penalidades (na maioria das vezes, financeiras) em casos de transgressões as regras por ele definidas. O SLA inclui detalhadamente informações do contrato, como limites de qualidade, prazos, valores, suporte técnico e etc.

Contudo, no inicio da Computação nas Nuvens, os Provedores de Serviço dedicavam um servidor completo para a execução da aplicação (AMARANTE, 2013). Esse modelo é satisfatório quando a aplicação consome todo o poder computacional do servidor. Contudo, caso isso não aconteça, o servidor se torna sub-utilizado, o que causa desperdício de recursos computacionais e custos desnecessários com refrigeração e energia para o *data center* (AMARANTE, 2013), além do alto impacto ambiental negativo que esse consumo de energia pode ocasionar (SALLES *et al.*, 2016). Neste contexto surgiu a necessidade de um avanço tecnológico para lidar com esse desperdício.

Percebeu-se então que para tirar melhor proveito da capacidade computacional de um servidor, era necessário executar aplicações de forma paralela. A partir dessa ideia de multiprocessamento, surgiu o conceito de Virtualização. A Virtualização consiste na abstração dos recursos físicos do hardware e os oferece como recursos virtuais às aplicações de alto nível (CARISSIMI, 2009). Com a Virtualização, cada máquina física pode ser dividida em

várias máquinas virtuais, que por sua vez possuem suas próprias quantidades de recursos (CPU, memória e rede), desde que não sejam maior que os limites do hardware do seu servidor.

Por meio da Virtualização, é possível consolidar servidores, ou seja, alocar uma grande quantidade de máquinas virtuais em um número pequeno de máquinas físicas e, dessa maneira, fazer com que os servidores trabalharem com grande parte da sua capacidade computacional. Através da consolidação de servidores é possível economizar bastante energia, pois os servidores que não possuem máquinas virtuais naquele momento podem ser desligados ou postos em modo de hibernação (modos com nenhum ou baixo consumo de energia). No entanto, quando muitas máquinas virtuais estão em uma mesma máquina física, elas disputam por recursos computacionais, como tempo de CPU ou espaço para alocação de memória. Essa concorrência pode fazer com que o desempenho das aplicações hospedadas seja degradado, ocasionando violações no SLA, uma vez que ele garante ao cliente limites sobre a qualidade do serviço. A partir disso surge o dilema entre consolidar e não consolidar servidores, ou, ao menos, como consolidá-los de tal modo que não degrade o desempenho dos servidores.

1.2 JUSTIFICATIVA

O uso da Computação em Nuvens está em constante crescimento (ALTOMARE; CESARIO, 2017). O aumento da demanda implica em um elevado número de *data centers* e uma maior utilização de recursos físicos dos servidores e de equipamentos de infraestrutura de TI (em especial de rede). No entanto, o aumento na utilização destes recursos leva a um maior gasto energético, pois além do consumo de energia dedicado para o funcionamento dos equipamentos, as máquinas físicas emitem bastante calor e, com isso, faz-se necessária uma eficiente refrigeração. O aumento no consumo de energia causa ainda um elevado impacto ambiental (ALHARBI *et al.*, 2016).

Pesquisas mostram que a taxa de crescimento no consumo energético dos *data centers* é de 18% ao ano e que, até 2020, eles serão responsáveis por 2% do consumo mundial de energia (SHARMA; REDDY, 2015). Segundo Kaplan *et al.* (2008), em 2008 todos os *data centers* do planeta, juntos, foram responsáveis pela emissão de tanto dióxido de carbono (CO_2) quanto a Argentina e a Holanda, acrescentou ainda que continuando dessa maneira, em 2020 a produção de CO_2 irá quadruplicar. Um *data center* consome em média o equivalente a 25000 residências (BUYYA *et al.*, 2010).

Dado esse contexto, se faz necessária uma política de alocação de máquinas virtuais

inteligente que busque economizar energia no *data center* e proporcionar uma redução no impacto ambiental causado pelo consumo de energia do *data center*, porém evite a concorrência em demasia por recursos computacionais que leva as violações de SLA e perdas de desempenho computacional.

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Propor e implementar uma nova política de alocação de máquinas virtuais em um *data center*, visando reduzir o consumo de energia e manter seu desempenho em níveis aceitáveis.

1.3.2 Objetivos Específicos

(a) Propor uma nova política de alocação de máquinas virtuais visando reduzir o consumo de energia do *data center*.
(b) Implementar a proposta no simulador CloudSim.
(c) Determinar os cenários de testes.
(d) Determinar as métricas utilizadas para analisar o desempenho.
(e) Analisar os resultados obtidos comparando-os com políticas de alocação existentes.
(f) Identificar os impactos positivos e negativos da nova proposta.

2 FUNDAMENTAÇÃO TEÓRICA

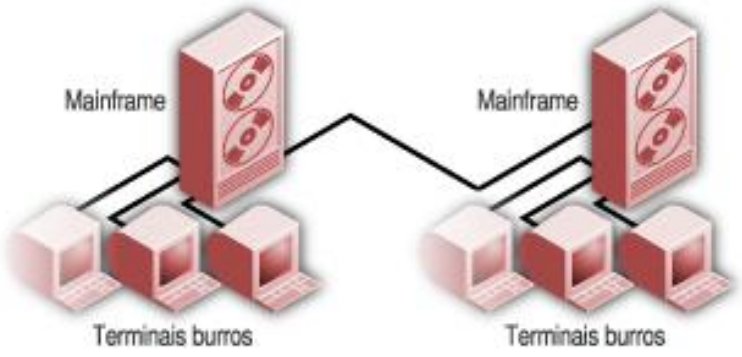
Este capítulo apresenta toda a fundamentação teórica necessária para entender a política de alocação apresentada neste trabalho. Dentre os assuntos abordados estão a computação em nuvens, o problema de múltiplas mochilas e a meta-heurística *Iterated Local Search*.

2.1 COMPUTAÇÃO EM NUVENS

A Computação em Nuvens consiste em uma infraestrutura distribuída de computadores que disponibilizam um conjunto de serviços, tais como rede, servidores, armazenamento e aplicações com o objetivo de prover funcionalidades a seus usuários de maneira rápida e eficiente (BORGES *et al.*, 2011).

Apesar do termo ser novo, esse conceito de computação é antigo, por volta de 1960 e 1970 os *mainframes* eram muito utilizados porque facilitavam e agilizavam as tarefas diárias das instituições. Um *mainframe* é um computador com alto poder de processamento e armazenamento de dados que supre as demandas de um conjunto de terminais burros. Um terminal burro é um computador que possui apenas operações de entrada e saída, onde todo o processamento é feito por um *mainframe* (MATOS, 2015). A Figura 1 representa terminais burros conectados a *mainframes*, note que os terminais burros possuem apenas monitor e teclado, responsáveis por operações de saída e entrada, respectivamente.

Figura 1 – Terminais burros conectados a mainframes



Fonte: (REDES. . . , 2001)

Esse paradigma de computação se originou com a ampliação do conceito adotado pelos *mainframes*, onde os usuários podem utilizar um serviço sem necessariamente ter o recursos físico para isso. Os usuários são atraídos para esse modelo por suas características vantajosas, como virtualização de recursos, serviços sob demanda, independência de localização,

elasticidade e escalabilidade (BORGES *et al.*, 2011).

A virtualização de recursos é possível por meio da utilização de máquinas virtuais, onde estas proporcionam a separação entre os serviços e os recursos físicos, pois pode-se utilizá-las sem considerar em qual *hardware* ela está sendo executada, ou onde esse hardware está localizado. Na Computação em Nuvem os recursos computacionais são consumidos através da internet, ou seja, os usuários podem utilizá-los de qualquer lugar do mundo e em qualquer dispositivo, como celulares ou computadores, independente da plataforma que o cliente estiver utilizando.

De maneira análoga aos serviços de utilidade pública, como água e energia, este serviço é ofertado sob demanda, ou seja, conforme sua necessidade os clientes podem demandar mais ou menos recursos computacionais, como tempo de processamento ou capacidade de armazenamento. A cada mudança os recursos da Nuvem são realocados dinamicamente. Nesse tipo de computação, os clientes pagam somente por aquilo que utilizarem, conceito conhecido como *pay-as-use*. A Computação em Nuvem possui escalabilidade porque tem potencial para atender a qualquer quantidade de recursos solicitadas pelos clientes, ela é também elástica, pois aumenta ou diminui a capacidade do sistema de acordo com a demanda.

O modelo conceitual encontrado com maior frequência na literatura é classificado entre três camadas, onde cada uma delas define o tipo de recurso a ser disponibilizado, são elas: Infraestrutura como Serviço (IAAS), Plataforma como Serviço (PAAS) e *Software* como Serviço (SAAS) (BUYYA *et al.*, 2010).

(a) **Infraestrutura como Serviço:** Nesta camada, a infraestrutura de hardware é oferecida como serviço, ou seja, os recursos computacionais são virtualizados para atender a demanda dos clientes. Os usuários utilizam esse serviço por meio de máquinas virtuais, onde especificam quanto de processamento, memória e rede querem consumir.

(b) **Plataforma como Serviço:** Camada responsável por ofertar plataformas como serviços, por exemplo banco de dados, serviços de armazenamento, ambientes de desenvolvimento de software e editores de texto.

(c) ***Software* como Serviço:** Camada composta por aplicações que são processadas no ambiente da nuvem e acessados por clientes através da internet. O *Google Docs* e o *Facebook* podem ser citados como exemplo de serviços deste nível.

A literatura classifica também este modelo em relação ao grau de compartilhamento, onde pode ser dividido em três categorias principais, são estas: Nuvens públicas, privadas ou

híbridas.

(a) **Nuvem Privada:** Uma nuvem privada, ou nuvem corporativa, é utilizada por uma única organização e possui um conjunto de restrições de acesso definidas por uma política. A instituição pode configurá-la como desejar, inclusive estabelecendo regras de *firewall*. A desvantagem é que todos as responsabilidade e custos com o gerenciamento da nuvem é unicamente da organização.

(b) **Nuvem Pública:** Neste tipo de nuvem, o público alvo não possui restrições, ou seja, todas as pessoas que possuírem o endereço do serviço podem utilizá-lo. São serviços que buscam suprir as necessidades dos usuários em geral, por isso generalizam as configurações, logo esta nuvem se torna inviável para serviços que exigem alta segurança.

(c) **Nuvem Híbrida:** A nuvem híbrida possui características da nuvem pública e privada. Esta nuvem pode atender serviços que exigem segurança e também serviços comuns a vários usuários. A complexidade em administrar todas essas características é a desvantagem desse tipo de nuvem.

A política de alocação de máquinas virtuais desenvolvida nesse trabalho é voltada para nuvens privadas, onde o serviço dessas nuvens se concentra na camada de infraestrutura como serviço. Uma vez explanado sobre Computação em Nuvens, é necessário com ter entendimento sobre o que são os problemas combinatórios para compreender o escopo deste trabalho. A próxima sessão define o que é um problema combinatório e faz uma ligação entre o problema de múltiplas mochilas e o problema de alocação de máquinas virtuais.

2.2 OTIMIZAÇÃO

A Otimização é um princípio que embasa a análise de vários problemas complexos de tomada de decisão. Tal análise envolve a seleção de valores para um conjunto de variáveis inter-relacionadas e tem o objetivo de medir a performance e verificar a qualidade da decisão. Este objetivo pode ser, dependendo do problema em questão, maximizado ou minimizado e sujeito às restrições que limitam a escolha dos valores possíveis às variáveis de decisão. É adequado aplicar uma otimização nos problemas combinatórios que possuem o aspecto de interesse que pode ser isolado e caracterizado adequadamente por um objetivo, como por exemplo: lucro ou custo em um cenário que busca determinar a rota de entrega de mercadorias (CABRAL, 1996).

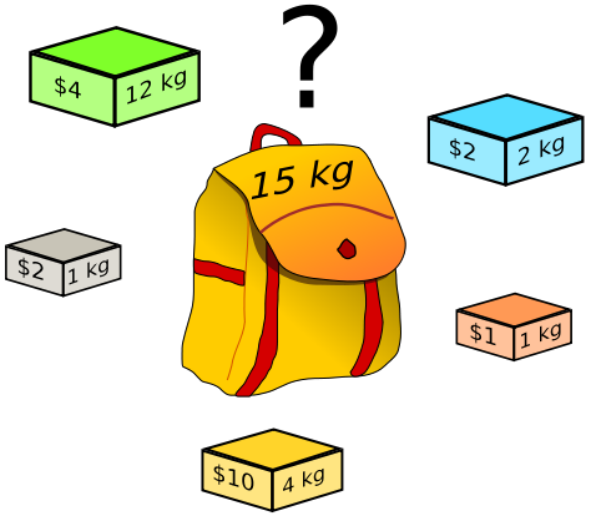
Após uma definição geral, a próxima subsessão trará um exemplo de um problema combinatório, conhecido como problema da mochila, e em seguida irá estendê-lo para o problema

de múltiplas mochilas.

2.2.1 PROBLEMA DE MÚLTIPLAS MOCHILAS

O problema da mochila surge com a necessidade de colocar um conjunto de itens em uma mochila, onde cada item tem um peso e um valor pré-definido. A mochila possui uma capacidade máxima de peso que pode suportar. Dessa maneira, o problema consiste em escolher os itens que serão postos dentro da mochila, com o objetivo de maximizar a soma dos valores dos itens carregados, porém sem extrapolar a capacidade máxima da mochila. A Figura 2 ilustra esse problema.

Figura 2 – Mochila com capacidade limitada e itens variados



Fonte: (AMARANTE, 2013)

Esse problema é categorizado como um problema combinatório, onde deve ser analisado um grande conjunto de possibilidades para só então conhecer a melhor solução possível.

Dado um conjunto de n itens, onde cada item i possui um peso w_i e um valor v_i agregado. Cada item possui um lucro associado, que é usado para definir a prioridade de inserção desse item na mochila. A mochila suporta no máximo um peso C. Para determinar se um item está ou não na mochila, é utilizado uma variável *booleana* x_i. Quando x_i é igual a 1, estão o item está na mochila, caso contrário o item não está na mochila. Como dito anteriormente, o objetivo desse problema é maximizar o lucro obtido através dos itens alocados na mochila.

Matematicamente, o problema é definido da seguinte maneira (AMARANTE, 2013):

$$Maximizar \quad z = \sum_{i=1}^{n} v_i x_i, \tag{2.1}$$

sujeito a

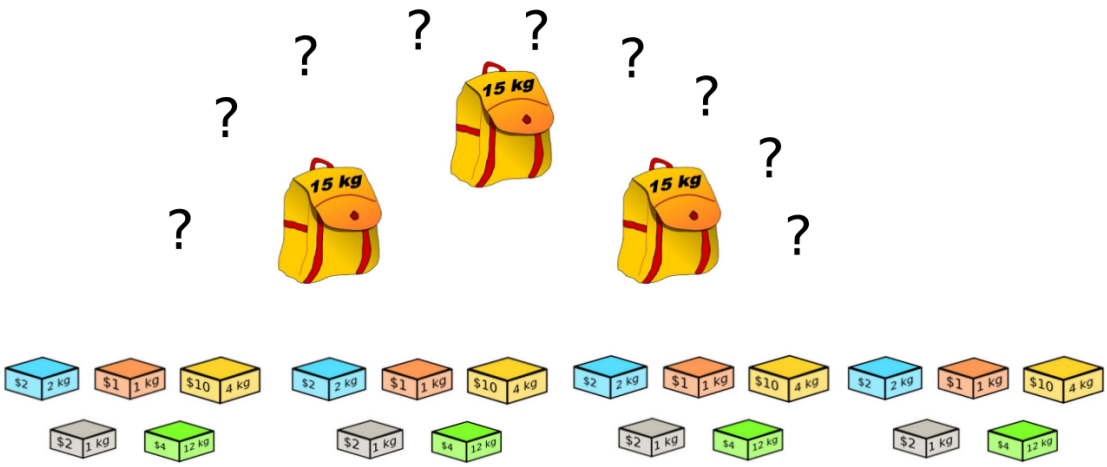
$$\sum_{i=1}^{n} w_i x_i \leq C \tag{2.2}$$

$$x_i = 1 \quad ou \quad x_i = 0, \quad i \in N = \{1,2,3,...,n\} \tag{2.3}$$

A Equação 2.1 é a função objetivo, onde busca-se maximizar o lucro obtido com a inserção de um item na mochila. A Equação 2.2 é uma restrição que garante que a soma dos pesos dos itens em uma mochila não ultrapasse a capacidade máxima C da mochila. Já a Equação 2.3 restringe quais itens devem estar na solução e quais não devem estar.

Uma variação do problema da mochila descrito anteriormente, é o problema de múltiplas mochilas. O contexto é o mesmo do problema original, mas este problema possui m mochilas, onde cada mochila $(j = 1,2,3,...,m)$ possui sua capacidade máxima C_j. A variável *booleana* que define em qual mochila está um determinado item é representada por x_{ij}, da qual informa se o item i está dentro da mochila j ou não, semelhante ao problema original. A Figura 3 representa o problema de múltiplas mochilas, observe que cada mochila possui uma capacidade máxima de carga, e que cada item possui um peso e um valor associado.

Figura 3 – Representação do problema de múltiplas mochilas



Fonte: (AMARANTE, 2013)

Matematicamente, pode-se modelar o problema da seguinte forma (MARTELLO; TOTH, 1990):

$$Maximizar \quad z = \sum_{j=1}^{m} \sum_{i=1}^{n} v_i x_{ij}, \tag{2.4}$$

sujeito a

$$\sum_{i=1}^{n} w_i x_{ij} \leq c_j, \quad {}_{j \in M} = \{1,2,3,...,m\} \tag{2.5}$$

$$\sum_{j=1}^{m} x_{ij} \leq 1, \quad {}_{i \in N} = \{1,2,3,...,n\} \tag{2.6}$$

$$w_i \leq max_{j \in N}\{c_j\}, \quad {}_{i \in M} \tag{2.7}$$

$$c_j \geq min_{i \in M}\{w_i\}, \quad {}_{j \in N} \tag{2.8}$$

$$v_i \quad e \quad c_j \quad \in \mathbb{Z}^{*}+, {}_{i \in N} = \{1,2,3,...,n\}, {}_{j \in M} = \{1,2,3,...,m\} \tag{2.9}$$

$$x_i = 1 \quad ou \quad x_i = 0, \quad {}_{j \in N}, \quad {}_{i \in M} \tag{2.10}$$

Análogo ao problema da mochila, a Equação 2.4 tem o objetivo de maximizar o valor obtido com os itens dispostos nas mochilas. A Equação 2.5 garante que a soma dos pesos dos itens comportados por uma mochila não ultrapasse o limite suportado por ela. A Equação 2.6 garante que o mesmo item não pode está em duas mochilas simultaneamente, ou seja, cada item só pode ser posicionado somente em uma única mochila. A Equação 2.7 garante que para cada item, existe uma mochila que consiga suportá-lo caso esteja vazia. A Equação 2.8 garante que a capacidade máxima de qualquer mochila é maior que o peso do menor item, ou seja, não existe mochila que não consiga comportar algum item. A Equação 2.9 faz com que o problema

trabalhe apenas com valores positivos. A Equação 2.10 define quais itens devem estar na solução e quais não devem estar.

Compreendido o problema combinatório de múltiplas mochilas, a próxima subsessão irá mapear suas restrições e objetivos com o problema em foco deste trabalho, o problema de alocação de máquinas virtuais.

2.2.2 PROBLEMA DE ALOCAÇÃO DE MÁQUINAS VIRTUAIS

É possível relacionar o problema de alocação de máquinas virtuais ao problema de múltiplas mochilas. Ao pensar na problemática dessa maneira, se torna possível resolver o problema com as soluções existentes para o problema de múltiplas mochilas. É possível relacionar as entradas desses problemas da seguinte forma:

(a) Os servidores equivalem as mochilas e a capacidade de processamento de um servidor equivale a capacidade da mochila.

(b) As máquinas virtuais representam os itens e a demanda de CPU de uma máquna virtual pode ser caracterizada como o peso do item.

(c) O consumo de energia causado por uma máquina virtual equivale a utilidade de um item.

Perceba que o objetivo do problema de múltiplas mochilas é maximizar o ganho, já o objetivo da política proposta é minimizar o consumo energético. Portanto, a função de avaliação deve ser modificada para que o problema seja de minimização, não mais de maximização.

Formalmente, a proposta pode ser definida da seguinte maneira: seja n o conjunto formado pelas máquinas virtuais que devem ser alocadas, e m o conjunto de servidores disponíveis em um *data center*. Cada máquina virtual $j \quad (j = 1,2,3,...,n)$ possui um peso w_j, formado pela quantidade de processamento p_j requisitada, e o consumo energético d_j ocasionado por ela. Cada servidor possui uma capacidade máxima de recursos de CPU chamada de c_i. A variável booleana x_{ij} recebe o valor 1 caso a máquina virtual j estiver alocada em um servidor i, caso contrário, essa variável booleana recebe o valor 0. O objetivo do problema de alocação de máquinas virtuais é minimizar o gasto energético do *data center*. Segue a modelagem matemática do problema de alocação de máquinas virtuais (AMARANTE, 2013):

$$Minimizar \quad z = \sum_{i=1}^{m} d_i, \tag{2.11}$$

sujeito a

$$\sum_{j=1}^{n} w_j x_{ij} \leq c_i, \quad i \in M = \{1,2,3,...,m\}, \tag{2.12}$$

$$\sum_{i=1}^{m} x_{ij} \leq 1, \quad j \in N = \{1,2,3,...,n\}, \tag{2.13}$$

$$w_j \leq max_{i \in M}\{c_i\}, \quad para \quad j \in N, \tag{2.14}$$

$$c_i \geq min_{j \in N}\{w_j\}, \quad para \quad i \in M, \tag{2.15}$$

$$w_j \quad e \quad c_i \quad sao \quad inteiros \quad positivos \tag{2.16}$$

$$x_{ij} = 0 \quad ou \quad 1, \quad i \in M, j \in N \tag{2.17}$$

A Equação 2.11 refere-se a função objetivo que busca minimizar o gasto energético do *data center*. A Equação 2.12 garante que a soma das demandas de CPU das máquinas virtuais alocadas em um servidor, não ultrapassam a capacidade máxima de processamento deste servidor. A Equação 2.13 restringe a alocação de cada máquina virtual em um único servidor. A Equação 2.14 garante que o *data center* possui pelo menos um servidor com capacidade para atender qualquer máquina virtual, ou seja, não existe uma máquina virtual tão grande que não possa ser alocada em algum servidor. A Equação 2.15 determina que não pode existir um servidor com capacidade de processamento inferior a menor demanda exigida entre as máquinas virtuais. A Equação 2.16 limita a valores positivos as demandas de CPU e capacidades dos servidores, ou seja, não pode haver servidor com capacidade de processamento negativa, nem máquina virtual que necessite de uma demanda negativa de CPU. A Equação 2.17 restringe a variável booleana x_{ij} entre os valores discretos 0 ou 1.

Apresentado alguns exemplos de problemas combinatórios (problema da mochila, problema de múltiplas mochilas e problema de alocação de máquinas virtuais), a próxima sessão explica como os problemas dessa categoria podem ser resolvidos e apresenta um método para resolvê-los, o *Iterated Local Search*.

2.3 MÉTODOS PARA RESOLUÇÃO DE PROBLEMAS COMBINATÓRIOS

Problemas de Otimização Combinatória podem ser resolvidos de dois modos, por um método exato ou um método aproximativo. Os métodos exatos garantem que a solução final é a melhor solução possível, chamada também de solução ótima global. Nos métodos aproximativos, não necessariamente a solução final é a solução ótima global, mas é uma ótima local que se aproxima da ótima global.

Para um problema com grandes instâncias, o método exato pode demorar um tempo computacional não aceitável para encontrar a solução ótima (OLIVEIRA, 2012), por analisar todas as possíveis soluções para o problema. Se uma solução aproximada for aceitável como solução para o problema, os métodos aproximativos podem ser utilizados. O tempo de execução de um algoritmo aproximativo é variável, ele depende da grandeza do problema e dos parâmetros envolvidos (CABRAL, 1996). Ou seja, quanto maior o problema a ser resolvido, maior o seu conjunto de soluções válidas. E quanto mais refinados forem os parâmetros do algorítimo, mais correta será a busca pela solução ótima.

Existem dois tipos de métodos aproximativos, as Heurísticas e as Meta-Heurísticas. O primeiro procura resolver um problema rapidamente, sem no entanto, garantir que esta seja a melhor solução possível para o problema, mas encontra uma solução aceitável que atende aos critérios definidos. São usadas para resolver um problema específico. Assim como o primeiro método, as Meta-Heurísticas buscam resolver o problema em um tempo computacional curto sem garantir encontrar a melhor solução do conjunto de soluções. No entanto, esse segundo método se difere do primeiro porque tem um caráter genérico, pode ser usados para resolver vários tipos de problemas, sendo necessário modificar apenas seus parâmetros.

2.3.1 ITERATED LOCAL SEARCH

A Meta-Heurística *Iterated Local Search* (ILS) explora o espaço de soluções através de perturbações em ótimos locais encontrados no decorrer das iterações (GLOVER; KOCHENBERGER, 2006). Seu funcionamento pode ser dividido em partes principais, são elas: Geração da Solução Inicial, Busca Local, Critério de Aceitação, Perturbação e Condição de Parada. A Figura 4 demonstra o funcionamento do algoritmo ILS.

Figura 4 – Procedimento heurístico *Iterated Local Search*

IteratedLocalSearch()
1. $s_0 \leftarrow GeraSolucaoInicial()$
2. $s^* \leftarrow BuscaLocal(s_0)$
3. **enquanto** (o critério de parada não estiver satisfeito) **faça**
4. $\quad s' \leftarrow Perturbacao(s^*)$
5. $\quad s^{*'} \leftarrow BuscaLocal(s')$
6. $\quad s^* \leftarrow CriterioDeAceitacao(s^*, s^{*'})$
7. **fim-enquanto**
8. **retorne** s^*;

Fonte: (GLOVER; KOCHENBERGER, 2006)

A Linha 1 desse algoritmo, gera uma solução inicial para o problema, ela pode ser qualquer solução válida, ou seja, qualquer solução que atenda as restrições do problema abordado. Ela pode ser gerada de maneira aleatória ou de maneira gulosa. Uma solução aleatória para um problema combinatório é gerar uma solução de qualquer modo desde que seja uma solução válida. Já uma solução gulosa busca seguir determinados padrões durante a alocação, como por exemplo, alocar os itens mais pesados primeiro ou utilizar mochilas mais pesadas antes das mais leves. É mais rápido gerar uma solução aleatória do que uma solução gulosa, no entanto, a qualidade de uma solução gulosa tende a ser melhor que de uma solução aleatória (POZO, 2008).

A Linha 2 realiza uma busca local na atual solução, ela gera um conjunto de vizinhanças da solução vigente, e escolhe o melhor dentre esses vizinhos. Uma vizinhança é um conjunto de soluções derivadas da solução atual. Para gerar um vizinho, deve-se realizar pequenas modificações no atual estado da solução.

A Linha 4 do algoritmo é a fase de perturbação de uma solução, que busca basicamente gerar uma solução (ou um vizinho) a partir de outra solução, por meio de algumas mudanças na solução vigente. No problema da mochila, por exemplo, essas mudanças podem ser realocações de itens de uma mochila para outra, respeitando as restrições. Note que a perturbação não se preocupa se a nova solução gerada é melhor ou pior que a solução corrente.

A Linha 6 é responsável por comparar a solução corrente com a solução gerada em cada iteração, ou seja, dadas duas soluções para o problema, o critério de aceitação define qual a melhor entre as duas soluções. A meta-heurística ILS busca em cada iteração, uma solução melhor que a solução vigente.

A Linha 3 é o controlador da quantidade de iterações, a condição de parada para essas iteração pode ser definida com base no tempo de execução do algoritmo, com base na qualidade da solução produzida ou mesmo na quantidade máxima de iterações. A meta-heurística ILS será executada enquanto a condição de parada da Linha 3 não for satisfeita.

Após entender o básico sobre o problema de alocação de máquinas virtuais e abordar uma maneira de resolver esse problema combinatório, é essencial compreender como outros autores resolveram esse mesmo problema. O próximo capítulo traz outras estratégias utilizadas para a resolução deste problema.

3 TRABALHOS RELACIONADOS

Esta seção discorre sobre alguns trabalhos que abordaram a mesma problemática que o presente trabalho, assim como descreve rapidamente suas soluções propostas.

3.1 *Optimizing Virtual Machine Consolidation in Virtualized Datacenters Using Resource Sensitivity*

Nasim *et al.* (2016) propõe uma estratégia para realizar a consolidação analisando os recursos utilizados por cada máquina virtual: processamento, memória primária e disco rígido. Ele busca alocar máquinas virtuais que necessitam dos mesmos recursos em servidores diferentes, com o objetivo de diminuir a concorrência pelos recursos em análise. Ao consolidar os servidores, é possível desligar ou colocar em modo de hibernação aqueles que não hospedam máquinas virtuais no momento da alocação.

Para tornar isso possível, o autor utilizou o conceito de sensibilidade definido por Taheri *et al.* (2017) (trabalho submetido para publicação em 2016, porém publicado em 2017). Uma máquina virtual é sensível a um recurso, se seu desempenho diminui muito caso este recurso lhe seja disponibilizado de maneira insuficiente. Ou seja, a sensibilidade de uma máquina virtual sobre um recurso é proporcional ao nível que ela necessita desse recurso. Em seu trabalho, o autor buscou alocar em um servidor máquinas virtuais sensíveis a recursos diferentes.

O autor investigou a importância de usar informações sobre a sensibilidade que cada máquina virtual possui em relação aos recursos que ela demanda. Trabalhou o problema com o foco multidimensional, portanto não considerou apenas o processamento para mensurar o desempenho das máquinas.

O conceito de overbooking se dá por permitir que sejam alocadas mais máquinas virtuais em um servidor que sua capacidade máxima permite, isso faz com que as máquinas virtuais concorram por recursos. Existem diferentes níveis de overbooking, caso seja igual a 1, significa que as demandas por recursos que os servidores receberão não será maior que sua capacidade de atendê-las, caso seja 1.5, então os servidores receberão 50% a mais de demanda que podem atender simultaneamente, caso seja 2, os servidores receberão 100% a mais de demanda que sua capacidade, e assim sucessivamente. O autor testou o algoritmo proposto usando diferentes níveis de overbooking, e notou uma melhoria de desempenho em comparação ao algoritmo que não usa informações de sensibilidade. Verificou-se também que quanto maior

o overbooking, maior era a economia de energia, contudo o desempenho era reduzido.

O algoritmo desenvolvido maximiza o desempenho das máquinas virtuais até o gasto de energia alcançar um limite superior determinado. O algoritmo pode ser adaptado para um sistema de prioridades, basta multiplicar o valor da sensibilidade pela prioridade que a máquina virtual deve ter. Nasim *et al.* (2016) afirma também que essa é a melhor estratégia teórica possível, em contrapartida, não se trata de uma heurística rápida, por esse motivo ele realizou os experimentos com apenas 45 máquinas virtuais e 20 servidores idênticos.

3.2 *A Robust Tabu Search Heuristic for VM Consolidation under Demand Uncertainty in Virtualized Datacenters*

Nasim e Kassler (2017) defende que para fazer uma consolidação eficiente deve-se conhecer a demanda de recursos de cada máquina virtual, o que segundo ele é difícil obter por causa da imprecisão na estimativa de demanda e da mudança dinâmica da carga de trabalho dos serviços em execução nas máquinas virtuais.

Para Nasim e Kassler (2017), uma estratégia robusta de alocação precisa considerar as variações de demanda de recursos de uma máquina virtual, porque mesmo consolidada, um servidor precisa ter capacidade para suportar tal variação. A estratégia adotada foi uma heurística baseada na Busca Tabu (TS). Adicionou a esta estratégia conceitos de Otimização Robusta (RO), que por sua vez trabalha com problemas de otimização onde os dados não são precisamente conhecidos, mas podem ser limitados. Para tornar a TS robusta, o autor modificou apenas a função de avaliação, ele adicionou um ruído R na solução atual, onde R reflete as mudanças esperadas nos dados de entrada. A cada iteração, a função de avaliação padrão, analisa apenas a relevância de um único resultado, já esta função de avaliação robusta examina um conjunto de soluções aproximadas ao invés de uma única solução, onde estas são compostas em um único valor. As máquinas virtuais de um servidor são ordenadas em ordem decrescente baseado no desvio máximo de demanda de recurso. A função analisa quanto as máquinas virtuais podem variar para continuar respeitando a restrição de capacidade máxima imposta por sua política. O resultado será aceito se o valor da função de avaliação for satisfatório.

Foi adotada a abordagem Γ *robustness* que controla o grau de conservadorismo, ou seja, determina quanto o *data center* será sensível as variações das máquinas virtuais, em outras palavras, determina quanto de capacidade um servidor terá para lidar com as incertezas de demandas de recursos. Γ representa o nível de proteção, e quanto maior seu valor, melhor ele

lida com as variações de demandas e menos violações de SLA acontecem. O autor considerou que o consumo de energia dos servidores é determinado apenas pela utilização da CPU.

3.3 *Which is the best algorithm for virtual machine placement optimization?*

Na literatura já foram propostos diversos algoritmos para o problema de alocação de máquinas virtuais, mas não há uma análise justa entre esses algoritmos. Em Mann e Szabó (2017), a comparação do desempenho das estratégias apresentadas é feita apenas com algoritmos não competitivos, geralmente com algoritmos de alocação aleatória ou com estratégias que não refletem as características principais do problema. Dado esse problema, o autor realizou a comparação de sete algoritmos de alocação distintos usando o mesmo ambiente e metodologia. Os trabalhos em análise focam apenas na utilização da CPU.

O ambiente de teste utilizado para realizar o experimento foi o CloudSim, para simular a rede e sua topologia, conversores para rastreamentos de carga de trabalho com o objetivo de monitorar o comportamento de um servidor frente as mudanças de carga, e por fim, um gerador de carga de trabalho para criar tarefas que exigem recursos computacionais de um servidor. Os atributos de cada máquina virtual são: A hora de chegada, desde o ponto inicial da simulação, a capacidade de CPU, de RAM e de disco solicitadas.

O autor utilizou como métrica para avaliar o desempenho das técnicas o impacto que a mudança de carga de trabalho causa no consumo de energia, a mudança da carga de CPU das máquinas virtuais, o tamanho das máquinas virtuais, a heterogeneidade do *data center*, as características de energia dos servidores e a detecção de sobrecarga.

As políticas de alocação analisadas em sua pesquisa foram propostas por Beloglazov *et al.* (2012), Lago *et al.* (2011), Guazzone *et al.* (2012), Chowdhury *et al.* (2015), Shi *et al.* (2013) e Calcavecchia *et al.* (2012).

3.3.1 BELOGLAZOV

Algoritmo proposto por Beloglazov *et al.* (2012), que já vem implementado no CloudSim, sugere realocar máquinas virtuais de servidores que estejam sobrecarregados. As máquinas virtuais que serão realocadas são aquelas que possuem menor tempo de migração. Um servidor é definido como sobrecarregado quando as máquinas virtuais nele alocadas estão consumindo ou utilizando uma porcentagem maior que a especificada pela política de alocação.

As máquinas virtuais sofrerão migração até que os servidores que as hospedam não estejam mais sobrecarregados.

3.3.2 LAGO

Lago *et al.* (2011) sugere alocar máquinas virtuais em servidores que possuam a maior eficiência energética. A eficiência energética é obtida através da divisão entre capacidade máxima de CPU e o consumo máximo de energia. Os servidores que possuem capacidade para receber determinada máquina virtual é classificado como um candidato a recebê-la. Entre os candidatos, aquele com maior eficiência energética é escolhido para abrigar a máquina virtual.

3.3.3 GUAZZONE

Guazzone *et al.* (2012) utilizou uma heurística BFD (*Best Fit Decreasing*) para definir uma alocação das máquinas virtuais. Em sua política, os servidores ligados tem prioridade aos servidores desligados, estes servidores são classificados em ordem decrescente de capacidade de CPU livre. Caso haja servidores com a mesma capacidade de CPU livre, os com menor consumo de energia ociosa possuem prioridade.

3.3.4 CHOWDHURY

Chowdhury *et al.* (2015) realiza a alocação de uma máquina virtual em um servidor que gere o maior crescimento no consumo de energia. Ainda que não pareça lógico, os autores defendem que alocações ruins podem causar estados melhores no futuro.

3.3.5 SHI

Shi *et al.* (2013) propõe seis algoritmos baseados no FFD (*First Fit Decreasing*). O autor sugere soluções para alocações de máquinas virtuais online (a alocação acontece enquanto novas máquinas virtuais chegam) e offline (a alocação acontece com uma quantidade fixa e bem conhecida de máquinas virtuais). O autor considera os recursos de CPU e memória principal em sua tomada de decisão através de uma variável chamada magnitude (Rq(v)). Entre os seis algoritmos, destacam-se o *PercentageUtil* e o *AbsoluteCapacity*. Ambos os algoritmos buscam, por meio da migração, liberar os servidores com menor consumo de recursos. *PercentageUtil* que aloca as máquinas virtuais para o servidor que está sendo mais utilizado, e o *AbsoluteCapacity*,

que por sua vez prefere alocar em servidor com maior capacidade.

3.3.6 CALCAVECCHIA

Calcavecchia *et al.* (2012) propôs o algoritmo Posicionamento Especulativo Reverso, ele usa o histórico das demandas de uma máquina virtual, para estimar seu comportamento futuro. A partir disso ele julga se deve ocorrer uma migração e para qual servidor a máquina virtual deve ser realocada.

4 MSIALLOCATOR

Este capítulo apresenta uma nova política de alocação de máquinas virtuais, denominada MSIAllocator. Esse trabalho concentra-se no desenvolvimento de uma política de alocação de máquinas virtuais em um *data center* considerando o consumo de energia e o desempenho dos servidores. O objetivo é minimizar o consumo energético dos servidores que contem as máquinas virtuais que devem ser atendidas pelo *data center*. Essa política utiliza a capacidade máxima de processamento de um servidor e seu consumo energético para determinar quais máquinas virtuais devem ser destinadas a ele. Dado a complexidade do problema abordado neste trabalho, utilizar um método aproximativo poderá trazer bons resultados. Para gerar a solução, foi utilizado a meta-heurística *Iterated Local Search* (ILS) (abordada na subseção 2.3.1).

A política de alocação de máquinas virtuais MSIAllocator pode ser dividida em duas partes principais: alocação inicial e realocação das máquinas virtuais. A primeira consiste em construir a solução inicial de maneira aleatória. A solução inicial é qualquer alocação válida das máquinas virtuais nos servidores de um *data center*. Entende-se como alocação válida, qualquer alocação tal que cada máquina virtual esteja alocada em um único servidor, e o somatório das demandas das máquinas virtuais alocadas em um servidor não ultrapassa a capacidade máxima do servidor em questão. O motivo da opção pela alocação aleatória foi o seu custo-benefício. Inicialmente, sabe-se somente a demanda máxima de recursos de cada máquina virtual e não os seus níveis reais de utilização, ou seja, o quanto cada máquina virtual consome por recurso. Dessa maneira, julgou-se desnecessária a aplicação de alguma técnica robusta e demorada na geração da solução inicial e optou-se por uma técnica simples e rápida: uma abordagem aleatória.

Já a segunda parte, onde se sabe as demandas reais por recursos, são selecionados através do MSIAllocator os servidores que deverão receber as máquinas virtuais que estão em processo de migração. O objetivo da política apresentada é economizar energia elétrica, então o melhor servidor será aquele que, ao receber a máquina virtual, resultar em uma maior economia de energia para o *data center*.

Vale ressaltar que, o trabalho em questão, não tem a responsabilidade de definir quais máquinas virtuais de um servidor devem ser migradas para outros servidores, pois isso é tarefa da política de seleção. O presente trabalho concentra-se em desenvolver uma política de alocação de máquinas virtuais, que realiza tanto a alocação inicial, quanto a realocação das máquinas virtuais selecionadas pela política de migração.

O Algoritmo representado na Figura 5 determina como ocorre o processo de escolha

do melhor servidor. Na linha 1, os servidores são ordenados de forma decrescente com base em sua capacidade máxima de processamento e na porcentagem de utilização de CPU. Esta ordenação prioriza a utilização dos servidores com mais capacidade de processamento e favorece sua consolidação. Na linha 4, a função *GeraSolucaoInicial()* aloca a máquina virtual da iteração no primeiro servidor que pode recebê-la, respeitando a atual ordenação desses servidores. Na linha 5, a função *BuscaLocal()* procura melhorar a solução anterior. Nessa função, busca-se o primeiro servidor que, ao receber a máquina virtual, economize mais energia que o servidor escolhido pela linha 4. O laço de repetição da linha 6 tem o objetivo de refinar a solução em cada iteração, nele é realizado uma perturbação na solução vigente, após busca-se melhorá-la, e por fim, é decidido se a iteração conseguiu ou não melhorar a solução. Na linha 8, a função *Perturbacao()* tem o objetivo de transformar uma solução em outra, para isso ela aloca a máquina virtual da iteração em qualquer servidor válido de maneira aleatória. Na linha 9, a *BuscaLocal()* tem o mesmo funcionamento da linha 5, no entanto, almeja melhorar a solução obtida na linha 8. Na linha 10, a função *CriterioDeAceitacao()* determina qual o servidor mais indicado até o momento para receber a máquina virtual em questão, ou seja, qual deles resultará em um menor gasto energético no final da simulação. Esse processo é repetido 25 vezes, tal valor foi estimado com base em observações empíricas.

Figura 5 – Pseudo-código da política MSIAllocator

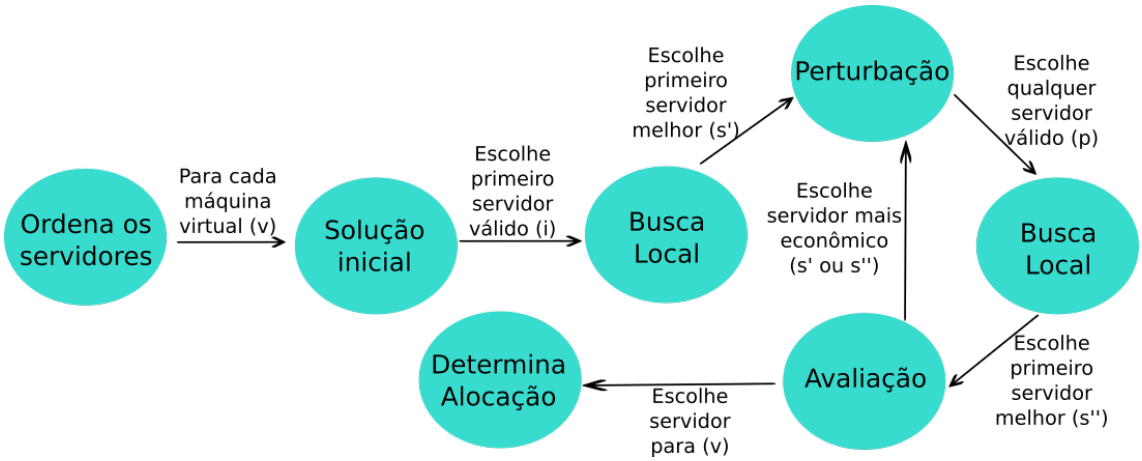
MSIAllocation(hosts, vms)
1. Sort(hosts)
2. **para cada** (vm em vms) **faça**
3. $i = 0$
4. $melhorHost \leftarrow GeraSolucaoInicial(hosts, vm)$
5. $melhorHost \leftarrow BuscaLocal(melhorHost, hosts, vm)$
6. **enquanto** (i < 25) **faça**
7. $i = i + 1$
8. $hostPerturbacao \leftarrow Perturbacao(melhorHost, hosts, vm)$
9. $hostLocal \leftarrow BuscaLocal(hostPerturbacao, hosts, vm,)$
10. $melhorHost \leftarrow CriterioDeAceitacao(melhorHost, hostLocal, hosts, vm)$
11. **fim-enquanto**
12. **se** ($melhorHost$!= NIL) **faça**
13. Alocar($melhorHost$, vm)

Fonte: próprio autor

O pseudo-código expresso na Figura 5 pode ser visualizado graficamente pelo diagrama da Figura 6. Cada círculo representa uma função e as setas descrevem a ação realizada

na etapa. A execução inicia com a ordenação dos servidores, como detalhado anteriormente. O restante dos passos serão executados para cada máquina virtual que precisa ser alocada. O primeiro passo é encontrar o primeiro servidor i da lista de servidores que possa receber a máquina virtual v em questão (etapa de gerar a solução inicial). Após isso, é feita uma busca linear na lista a procura do primeiro servidor s' que possa receber v, e que, ao recebê-la, gaste menos energia elétrica que i (primeira etapa de busca local). O próximo passo é escolher aleatoriamente algum servidor válido p que possa receber v (etapa de perturbar a solução). Após isso, é executada novamente uma busca linear a procura do primeiro servidor válido s'', e que, ao receber v, gaste menos energia elétrica que p (segunda etapa de busca local). Então, uma decisão deve ser tomada: É melhor que a máquina virtual v seja alocada em s' ou em s''? (etapa de avaliação da solução encontrada) O ciclo formado por *Perturbação(), BuscaLocal() e Avaliação()* será executado 25 vezes. Finalmente, a máquina virtual v será alocada no servidor que, ao hospedar v, resulte em um menor gasto energético ao *data center* (etapa de determinar a alocação de v).

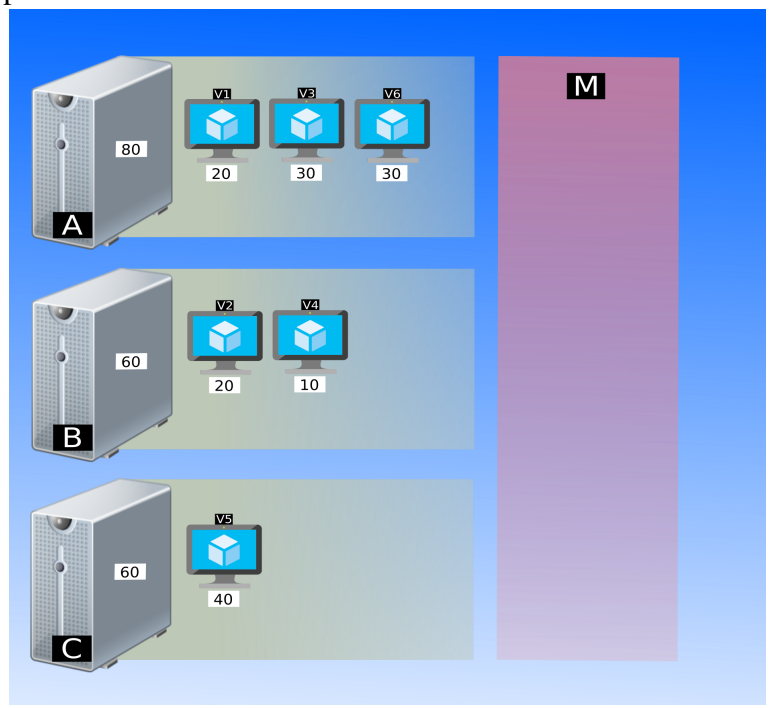
Figura 6 – Processo de realocação da política MSIAllocator



Fonte: Próprio autor

A Figura 7 exemplifica uma possível solução inicial aleatória. Considere que o os servidores A, B e C possuem capacidades máxima de processamento 80, 60 e 60 MIPS, respectivamente. Imagine que o servidor A possui as máquinas virtuais v_1, v_3 e v_6, com uma demanda de recursos de 20, 30 e 30 MIPS, respectivamente. O servidor B possui as máquinas virtuais v_2 e v_4, com demanda de 10 e 20 MIPS, respectivamente. E o servidor C possui a v_5, com demanda 40 MIPS. Na área em vermelho, chamada de M, ficará as máquinas virtuais que devem ser realocadas.

Figura 7 – Exemplo de servidores antes do refinamento



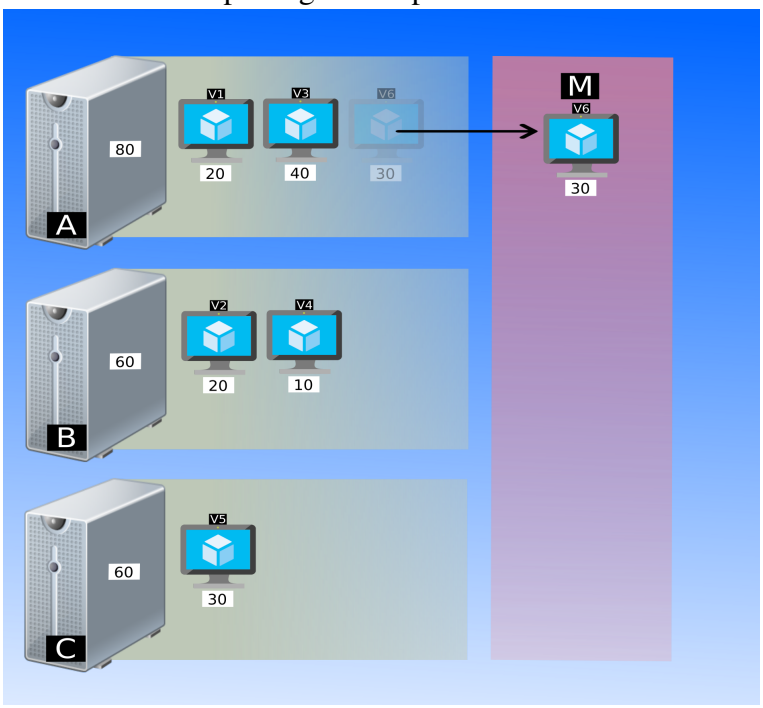
Fonte: Próprio autor

Se as máquinas virtuais de um servidor passarem a requisitar mais CPU do que quando foram alocadas nele, este servidor poderá se tornar sobrecarregado, ocasionando a perda de desempenho e a eventual violação de SLA. Já se essas máquinas virtuais passarem a requisitar menos CPU do que quando foram alocadas nele, este servidor poderá se tornar subutilizado, podendo ocasionar prejuízo financeiro ao *data center*, uma vez que ele consome muita energia estando ativo e trabalha pouco.

A variação das demandas de CPU pode tornar os servidores subutilizados ou sobrecarregados. Quando um servidor está sobrecarregado é necessário realocar algumas de suas máquinas virtuais para outros servidores para controlar a violação de SLA. Quando um servidor se torna subutilizado, então todas as suas máquinas virtuais devem ser transferidas para outros servidores com o intuito de tornar o servidor inativo. Os servidores ativos são priorizados em relação aos inativos para promover a consolidação das máquinas virtuais, visto que minimizar a quantidade de servidores ativos pode representar uma economia de energia.

A Figura 8 exemplifica uma possível variação, onde a máquina v_3 passou a requisitar 10 MIPS a mais. E a máquina virtual v_5 passou requisitar 10 MIPS a menos. Note também, que neste exemplo, a política de seleção decidiu que a máquina virtual v_6 precisa ser migrada para outro servidor.

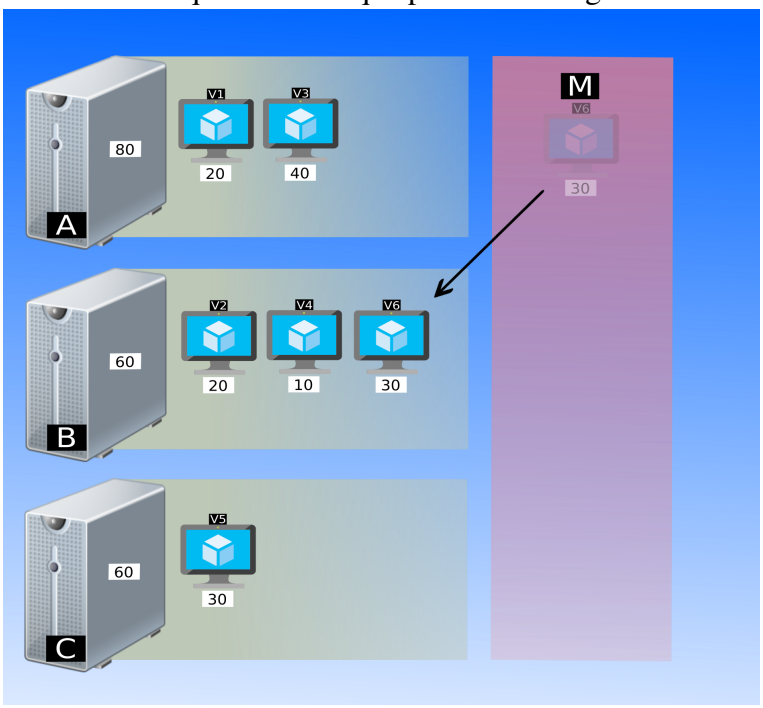
Figura 8 – Estado dos servidores após algum tempo



Fonte: Próprio autor

A meta-heurística aplicada nesse trabalho realiza a realocação das máquinas virtuais selecionadas para migração, uma por vez, onde o primeiro passo para começar a refinar a solução é alocar cada máquina virtual em um servidor de forma aleatória, ver linha 4 da Figura 5. Essa alocação é feita ordenando todos os servidores com base na capacidade máxima de processamento e no consumo de energia, em seguida a máquina virtual será alocada no primeiro servidor capaz de recebê-la. A Figura 9 sugere que o servidor selecionado no exemplo foi o B.

Figura 9 – Realocando uma máquina virtual que precisa ser migrada



Fonte: Próprio autor

Para realizar essa tarefa utilizando o ILS, é necessário definir como ocorrerá a perturbação de uma solução, a busca local, a função de aceitação e o critério de parada do algoritmo. As subseções a seguir descrevem cada uma dessas funções propostas por este trabalho.

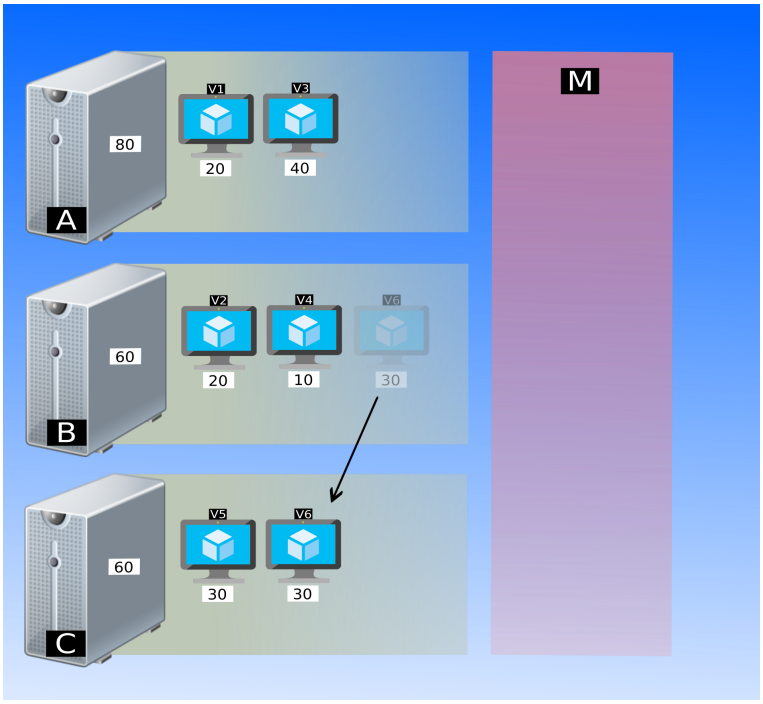
4.1 PERTURBAÇÃO

A fase de perturbação corresponde a modificações na solução atual, para que esta se transforme em outra solução. Perturbações grandes alteram significativamente as características da solução, isso é ruim porque a solução acaba perdendo suas propriedades e se transforma em uma solução aparentemente aleatória (GLOVER; KOCHENBERGER, 2006). Perturbações muito pequenas alteram muito pouco as características da solução, o que também não é interessante para esta abordagem, pois a solução gerada irá ser muito semelhante a atual, obtendo assim poucas melhorias.

No presente trabalho, a perturbação de uma solução se dá por realocar uma máquina virtual de um servidor para outro servidor ativo, desde que a alocação obedeça as restrições do problema e que este servidor não se torne sobrecarregado após a alocação. A escolha de qual servidor receberá essa máquina virtual é aleatória. Este passo é realizado na linha 8 da Figura 5.

Seguindo com o exemplo, uma perturbação pode ser mover a máquina virtual v_6 do servidor B para o servidor C, como demonstrado na Figura 10:

Figura 10 – Exemplo de perturbação



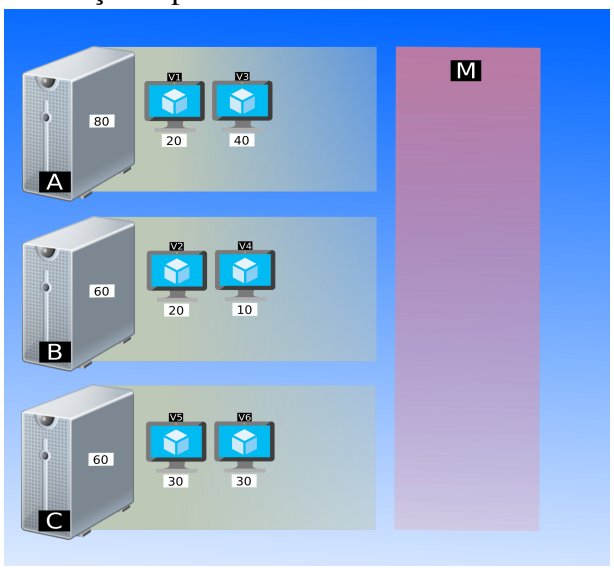
Fonte: Próprio autor

4.2 BUSCA LOCAL

A parte principal deste trabalho é a busca local, porque a partir de uma solução qualquer ela procura realizar o refinamento da solução, encontrando outra melhor. A busca local utilizada foi a que procura a primeira solução melhor que a atual. Os servidores ativos são ordenados de forma decrescente de capacidade de CPU livre, de modo a evitar que servidores fiquem sobrecarregados. Este passo é realizado na linha 9 da Figura 5.

A busca local procura consolidar as máquinas virtuais, realocando-as para servidores que consumem menos energia elétrica que o atual onde estão, desde que sejam capazes de atendê-las de modo que evite uma considerável degradação do desempenho e minimize o consumo energético. Com o decorrer das iterações do ILS, a solução deve ser reorganizada de forma a minimizar a quantidade de energia utilizada. Para demonstrar tal resultado, o exemplo dado anteriormente deve ser transformado em uma solução com a forma representada na Figura 11.

Figura 11 – Exemplo de alocação esperada



Fonte: Próprio autor

4.3 FUNÇÃO DE ACEITAÇÃO

Após a perturbação de uma solução, é necessário saber se a nova solução gerada pela busca local é melhor ou pior que a original. O papel da função de aceitação é definir entre duas soluções qual a melhor. Portanto ela recebe duas soluções como parâmetro e calcula quanto de energia elétrica ambas estão consumindo, em seguida retorna aquela com o menor gasto energético. A solução retornada será a melhor da iteração atual. Este passo é realizado na linha 10 da Figura 5.

Um servidor ativo, mesmo estando inutilizado, consome cerca de 70% da energia que gasta quando está sendo utilizado em pico (FAN *et al.*, 2007), os outros 30% são proporcionais às tarefas que estiverem sendo executadas. Para mensurar quanto de energia elétrica cada solução está consumindo naquele momento, calcula-se o somatório do consumo de energia de cada servidor presente na solução.

4.4 CRITÉRIO DE PARADA

Neste trabalho, o critério de parada do ILS adotado foi o número de iterações. Dessa maneira pode-se controlar a quantidade de soluções visitadas em relação ao tempo de processamento. É desejável possuir o controle da qualidade da solução em relação ao tempo de execução, pois com isso é possível regular esses valores de forma a priorizar a qualidade da solução ou a velocidade de resposta. Assim, foi definido 25 iterações com base em observações

empíricas, como pode ser observado na linha 6 da Figura 5.

Após definir o funcionamento da política de alocação MSIAllocator, é necessário meios de validar seu desempenho. O próximo capítulo detalha os cenários em que os testes foram executados, as métricas utilizadas para medir o desempenho da política e os resultados obtidos em cada métrica.

5 ANÁLISE DE DESEMPENHO

Após a exposição da política MSIAllocator apresentada neste trabalho, é necessário validar sua eficiência em relação as demais políticas já existentes. Para executar tal tarefa, foi escolhido o simulador *CloudSim* (CALHEIROS *et al.*, 2011). Esta ferramenta é apropriada para simular os procedimentos e relacionamentos de vários elementos que compõe uma arquitetura convencional de Computação em Nuvens como, por exemplo, *data centers*, servidores, máquinas virtuais, políticas de seleção e de migração etc.

A proposta MSIAllocator foi testada em dois cenários. Inicialmente, foram realizados testes no cenário proposto por Mann e Szabó (2017), onde a MSIAllocator obteve bons resultados quando comparada com as políticas analisadas pelo trabalho correlato. Visando aumentar a confiabilidade dos resultados, testou-se a MSIAllocator com as três melhores políticas do Cenário 1 no cenário proposto por Beloglazov e Buyya (2012). Neste cenário a MSIAllocator também obteve bons resultados quanto a métrica de energia.

Nas sessões a seguir serão detalhados os dois cenários de testes e quatro métricas, onde o primeiro cenário é o mesmo utilizado no trabalho de Mann e Szabó (2017) e o segundo é o mesmo utilizado por Beloglazov e Buyya (2012). Ao final deste capítulo será feita uma análise dos resultados obtidos nas simulações.

5.1 Cenário 1 - Proposto por Mann e Szabó (2017)

Para o resultado da validação simulada ser considerada próxima da realidade, buscou-se utilizar especificações próximas de um *data center* convencional, como as configurações dos servidores, as características das máquinas virtuais e tarefas com perfis de consumo real. Para atender esses requisitos, Mann e Szabó (2017) utilizou as mesmas configurações e métricas que Beloglazov e Buyya (2012), que serão descritas a seguir.

O *data center* foi configurado para ter 800 servidores, onde metade destes é do tipo HP ProLiant G4 e o restante é do tipo HP ProLiant G5. A diferença desses dois tipos de servidores é a capacidade de processamento, onde possuem 1860 MIPS por núcleo e 2660 MIPS por núcleo, respectivamente. A Tabela 1 apresenta todas as configurações de cada um desses tipos de servidores.

Tabela 1 – Configurações dos servidores utilizados

Recurso	HP ProLiant G4	HP ProLiant G5
Número de Núcleos	2	2
Processamento	1860 MIPS	2660 MIPS
Memória Primária	4 GB	4 GB
Memória Secundária	1 TB	1 TB
Largura de Banda	1 Gbit/s	1 Gbit/s
Qtde de servidores	400	400

Fonte: (BELOGLAZOV; BUYYA, 2012)

Além da definição desses servidores, é necessário determinar também qual o consumo energético que cada um possui. O consumo energético é computado com base na sua carga de trabalho em um determinado instante, o nível de consumo é mensurado através da porcentagem da carga de trabalho. A Tabela 2 detalha a relação da carga com o consumo. Os valores intermediários são calculados através da interpolação entre os valores piso e teto mais próximos a ele.

Tabela 2 – Consumo de energia por carga de trabalho

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

Fonte: (BELOGLAZOV; BUYYA, 2012)

Para uma simulação próxima da realidade, as máquinas virtuais também devem ter características reais. A empresa Amazon disponibilizou através do serviço Amazon EC2 (AMAZON, 2018) quatro instâncias de máquias virtuais, as quais foram utilizadas neste trabalho, são estas: Microinstância (*Micro Instance*), Instância Pequena (*Small Instance*), Instância Extra Grande (Extra Large Instance) e Instância Média Otimizada Para Computação (*High-CPU Medium Instance*). A Tabela 3 apresenta as configurações de cada uma desses tipos de máquinas virtuais adotados.

Tabela 3 – Configurações das máquinas virtuais

Recurso	High-CPU Medium	Extra Large	Small	Micro
Qtde. de núcleos	1	1	1	1
Processamento	2500 MIPS	2000 MIPS	1000 MIPS	500 MIPS
Memória Primária	870 MB	1.7 GB	1.7 GB	613 MB
Tamanho da máquina virtual	2.5 GB	2.5 GB	2.5 GB	2.5 GB
Largura de banda	10 Mbits/s	10 Mbits/s	10 Mbits/s	10 Mbits/s

Fonte: Adaptada de (BELOGLAZOV; BUYYA, 2012)

Uma entidade importante do simulador é o *Cloudlet*, cada instância de Cloudlet tem a função de gerenciar a execução de uma tarefa, como determinar a quantidade de recursos requisitados por ela em um determinado momento. Em todos os cenários cada Cloudlet é executado por uma única máquina virtual. As tarefas (carga de trabalho) possuem três fontes oriundas de ambientes reais: PlanetLab, Bitbrains e Google Cluster.

Os dados obtidos através do PlanetLab retratam a carga de trabalho de 1052 máquinas virtuais durante 1 dia. Já os obtidos através do Google Cluster retratam a carga de trabalho de 12000 servidores durante 29 dias. Os dados obtidos através do provedor BitBrains são de 1750 máquinas virtuais em um período de 4 meses. Nas três fontes citadas o período de amostragem se deu a cada 5 minutos. As demais características desses três ambientes são descritas na Tabela 4.

Tabela 4 – Resumo dos rastreamentos de carga de trabalho do mundo real utilizados

Origem	Virtualizado	CPU	Memória	Disco
PlanetLab	sim	sim	não	não
Google	não	sim	sim	sim
Bitbrains	sim	sim	sim	não

Fonte: Adaptada de (MANN; SZABÓ, 2017)

5.2 Cenário 2 - Proposto por Beloglazov e Buyya (2012)

O trabalho de Beloglazov e Buyya (2012) utilizou os dados oriundos do PlanetLab para compor o conjunto de dados de carga de trabalho. Esses dados foram obtidos através de um monitoramento de 10 dias, onde em cada dia, houve uma variação da quantidade de máquinas virtuais que necessitavam ser alocadas. A Tabela 5 faz o mapeamento dos dias em relação a quantidade de máquinas virtuais. O trabalho de Mann e Szabó (2017), como dito anteriormente, também utilizou dados oriundos do PlanetLab, entretanto, foi adotado apenas os dados referente ao dia 03/03/2011 (primeira entrada da Tabela 5). O porquê da utilização de apenas este dia específico não foi esclarecido pelo autor. Salvo esta exceção, os cenários propostos são idênticos: mesmas configurações de máquinas virtuais e de servidores, igual quantidade de servidores, padrão de consumo de energia equivalente das máquinas físicas.

Este cenário analisa, além da MSIAllocator, as três políticas que mais economizaram energia no cenário adotado por Mann e Szabó (2017) utilizando os dados do PlanetLab (Seção 5.5.1). As políticas selecionadas foram as duas propostas por Shi *et al.* (2013) e a proposta por Guazzone *et al.* (2012).

Tabela 5 – Quantidade de máquinas virtuais por dia de simulação

Dia	Data	Número de máquinas virtuais
1	03/03/2011	1052
2	06/03/2011	898
3	09/03/2011	1061
4	22/03/2011	1516
5	25/03/2011	1078
6	03/04/2011	1463
7	09/04/2011	1358
8	11/04/2011	1233
9	12/04/2011	1054
10	20/04/2011	1033

Fonte: Adaptada de (BELOGLAZOV; BUYYA, 2012)

5.3 Métricas

Este trabalho utilizará as mesmas métricas utilizadas por Mann e Szabó (2017) e mais uma referente a quantidade de migrações realizadas durante a simulação. A métrica com maior peso nas análises será a quantidade de energia consumida ao final da simulação, dado que o objetivo principal deste trabalho é minimizar o gasto energético.

As demais métricas são referentes ao desempenho do *data center*, são estas: *Performace Degradation due to Migration (PDM)*, que mede a violação de SLA causada pelas migrações, a *SLA violation Time per Active Host (SLATAH)*, que mede a violação causada pela utilização total do servidor, por fim, a quantidade de migrações efetuadas durante a simulação. Nas próximas sessões será detalhada cada métrica.

5.3.1 Quantidade de migrações e Performace Degradation due to Migration

Durante a execução das tarefas pelas máquinas virtuais, elas podem exigir mais ou menos capacidade de processamento de um servidor. Como dito anteriormente, esse processo pode tornar um servidor subutilizado ou sobrecarregado, necessitando assim realizar migrações para manter a utilização dos servidores em determinados níveis.

O processo de migrar uma máquina virtual utiliza parte dos recursos alocados a ela para executar a migração, isso pode gerar uma violação de SLA porque a máquina virtual não está tendo acesso a todo o recurso que deveria ser disponibilizado. O *CloudSim* representa essa violação através da métrica PDM. Essa métrica calcula, em porcentagem média, a quantidade média de MIPS que estava indisponível às máquinas virtuais durante a migração. A Fórmula 5.1

demostra matematicamente como essa métrica é calculada.

$$PDM = \frac{1}{M} \sum_{i=1}^{M} \frac{C_{dj}}{C_{rj}} \tag{5.1}$$

Onde:

M = Número de máquinas virtuais

C_{dj} = CPU não utilizada devido a migração

C_{rj} = CPU solicitada durante a migração

5.3.2 SLA violation Time per Active Host

Sempre que um servidor utilizar sua capacidade total de processamento, nenhuma das suas máquinas virtuais pode solicitar mais recursos de *hardware*, porque basta que alguma entre elas solicite uma pequena quantidade de MIPS a mais para ocasionar uma violação de SLA.

O *CloudSim* representa essa violação através da métrica SLATAH. Essa métrica calcula, em porcentagem média, a quantidade de tempo em que os servidores utilizaram sua capacidade máxima de recursos. A Fórmula 5.2 demostra matematicamente como essa métrica é calculada.

$$SLATAH = \frac{1}{N} \sum_{i=1}^{N} \frac{T_{si}}{T_{ai}} \tag{5.2}$$

Onde:

N = Número de máquinas virtuais

T_{si} = Tempo que o servidor i está utilizando toda CPU

T_{ai} = Tempo que o servidor i está ativo

5.4 Configuração da máquina que efetuou a simulação

A simulação foi executada em uma máquina com processador Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz com arquitetura 64-bit. Possui uma memória RAM DDR3 de 8 GB com velocidade de clock 1600 MHz. A memória secundária possui 1 TB de capacidade. A placa de vídeo é integrada, especificamente, é uma Intel HD graphics 520. O sistema operacional utilizado é o Ubuntu versão 16.04. A IDE utilizada foi o Eclipse Neon.3 Release (4.6.3).

5.5 Resultados dos testes do cenário 1

Após efetuar a simulação considerando diferentes cargas de trabalho com as políticas utilizadas por Mann e Szabó (2017) e a política proposta neste trabalho, esta sessão irá apresentar os resultados através de uma comparação baseada nas métricas citadas anteriormente.

Por questões de simplificação, os nomes das políticas de alocação foram abreviados na construção dos gráficos e na análise dos resultados. A Tabela 6 faz o mapeamento dos nomes reais das políticas de alocação de máquinas virtuais para as siglas definidas.

Tabela 6 – Mapeamento entre siglas e nome das políticas

Sigla	Nome	Autor
MSI	MSIAllocator	próprio autor
LAG	LagoAllocator	(LAGO *et al.*, 2011)
PER	PercentageUtil	(SHI *et al.*, 2013)
ABS	AbsoluteCapacity	(SHI *et al.*, 2013)
CAL	Especulativo Reverso	(CALCAVECCHIA *et al.*, 2012)
CHO		(CHOWDHURY *et al.*, 2015)
GUA	Approach-M	(GUAZZONE *et al.*, 2012)

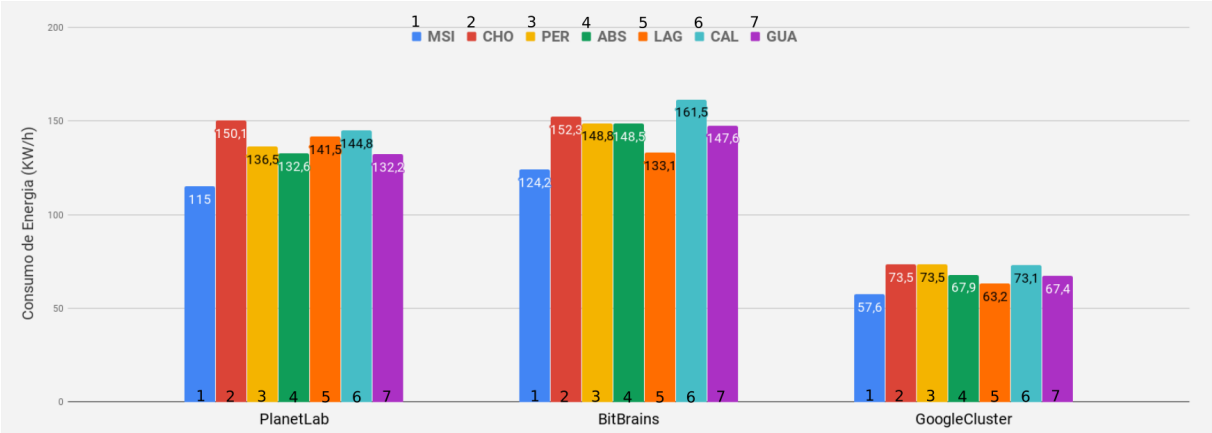
Fonte: próprio autor

5.5.1 Consumo de energia

No quesito economia de energia, percebe-se, através do gráfico apresentado na Figura 12, que a política MSIAllocator apresentou melhores resultados que as propostas analisadas por Mann e Szabó (2017). Nos três conjuntos de carga de trabalho, estima-se que a política MSIAllocator apresentou uma economia de 17.2 kWh, 8.9 kWh e 5.6 kWh em relação às políticas segundas colocadas que foram, respectivamente, GUA (PlanetLab) e LAG (BitBrains e GoogleCluster). Apresentou também uma economia de 35.1 kWh, 37.3 kWh e 15.9 kWh se comparada com as políticas que mais consumiram energia que foram, respectivamente, CHO

(PlanetLab e GoogleCluster) e CAL (BitBrains). Tais resultados comprovam a eficiência da política MSIAllocator em relação a economia de energia.

Figura 12 – Consumo energético em cada instância de dados



Fonte: próprio autor

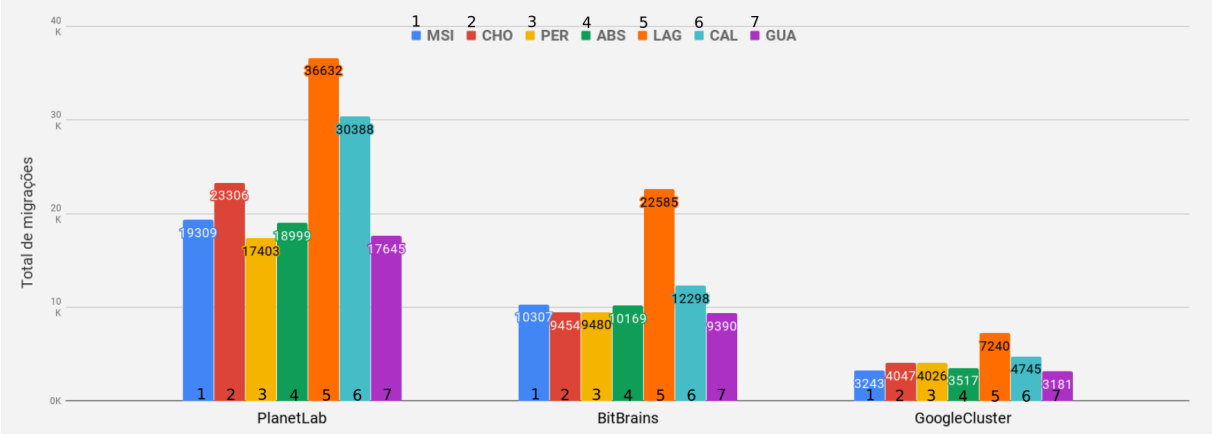
5.5.2 Quantidade de migrações

Quando um servidor está sobrecarregado, a política de seleção escolhe algumas de suas máquinas virtuais para serem migradas para outros servidores. Quando um servidor está subutilizado, todas as máquinas virtuais serão migradas para outros servidores. Uma boa política de alocação, deve realizar suas alocações de tal forma que evite que os servidores fiquem sobrecarregados ou subutilizados, para que minimize a quantidade de migrações realizadas.

Em relação ao número total de migrações realizadas durante a simulação, a política de alocação sugerida neste trabalho apresentou resultados semelhantes ao das políticas que realizaram menos migrações. Na Figura 13, pode-se notar, que nos três conjuntos de carga de trabalho a política MSIAllocator realizou cerca de 1900, 900 e 60 migrações a mais que as melhores políticas nessa métrica que foram, respectivamente, PER (PlanetLab) e GUA (BitBrains e GoogleCluster).

Apesar de não obter os melhores resultados, nota-se, através desses dados, que a política apresentada obteve desempenho satisfatório. A quantidade de migrações reflete a estabilidade que as alocações efetuadas geram nos servidores. Um número muito grande de migrações significa que a política está fazendo alocações que terão que ser refeitas em um curto período de tempo. Já uma quantidade pequena de migrações, significa que a política está tomando decisões seguras nas alocações das máquinas virtuais.

Figura 13 – Quantidade de migrações em cada instância de dados



Fonte: próprio autor

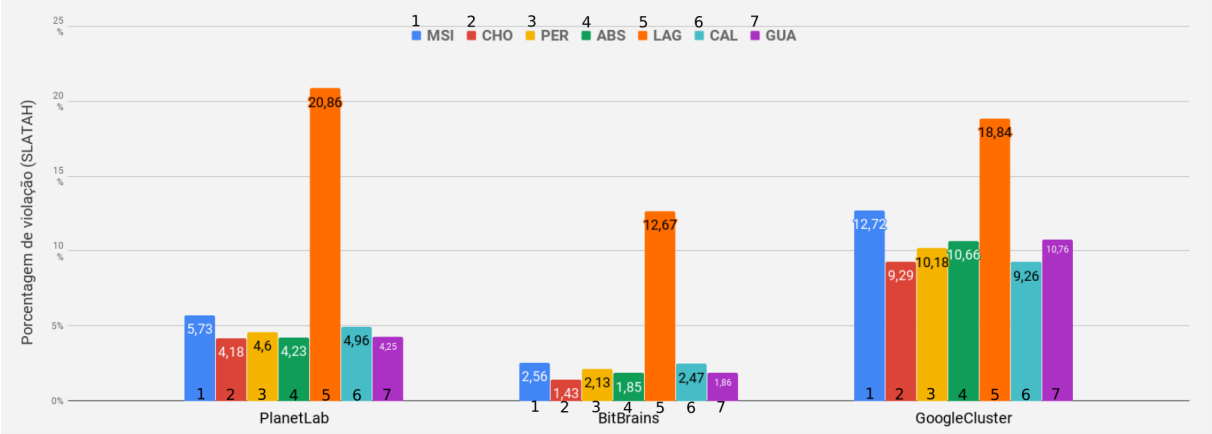
5.5.3 Métrica SLATAH

Esta métrica avaliadora foi a única, dentre as quatro, que se mostrou completamente desfavorável a política MSIAllocator, como pode ser observado na Figura 14. Dentre as sete propostas analisadas, a política de alocação sugerida neste trabalho foi a que apresentou o segundo pior desempenho, ganhando apenas da política LAG, com uma vantagem de 13,55%, quando utilizado os dados do PlanetLab. A MSIAllocator violou em média 1,81% a mais que a política que menos violou essa métrica, que foi a CHO, também quando utilizado os dados do PlanetLab.

Apesar de ter sido pior que a maioria das políticas analisadas, é um resultado compreensível, pois existe um *tradeoff* entre o consumo energético e o desempenho dos servidores. Quanto mais consolidados estiverem os servidores, menos energia o *data center* consumirá, no entanto, as máquinas virtuais irão concorrer mais por recursos físicos, reduzindo assim o desempenho dos servidores e ocasionando uma maior violação de SLA.

Esse comportamento pode ser justificado pela maneira com que a MSIAllocator determina quais os melhores servidores para receber as máquinas virtuais. Para tal, a função *CriterioDeAceitacao()*, que tem como objetivo escolher a melhor solução entre as duas a ela apresentadas, opta por aquela que apresenta um menor incremento de energia no datacenter, ou seja, aquela que mais consolida os servidores e tende a aumentar o número de violações de SLA.

Figura 14 – Violações SLATAH em cada instância de dados



Fonte: próprio autor

5.5.4 Métrica PDM

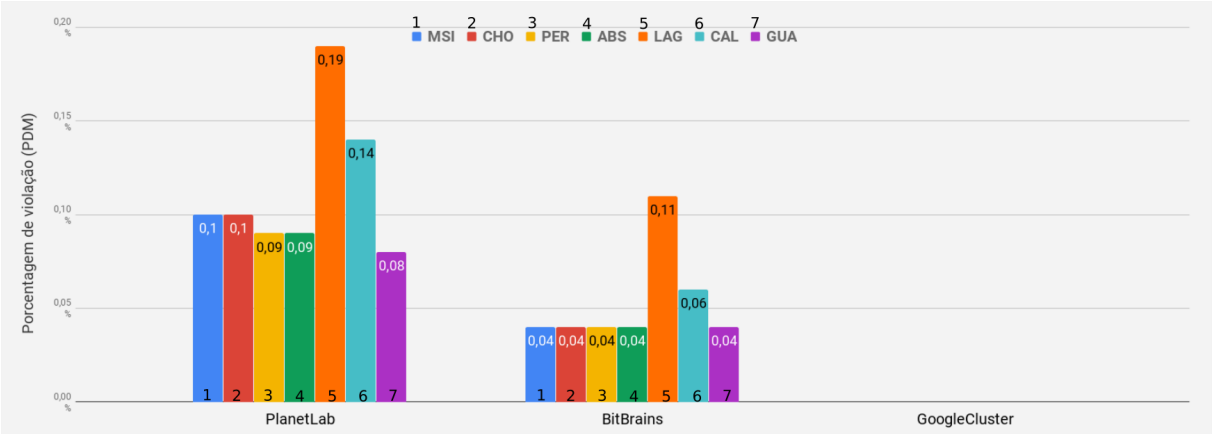
Na medição da violação de SLA causada pelas migrações, a política proposta neste trabalho apresentou resultado satisfatório, sendo semelhante ao da política que violou menos a métrica PDM, que foi a GUA (PlanetLab e BitBrains). Destaca-se que, a maior parte das políticas analisadas (dentre elas aquela proposta neste trabalho) obtiveram resultados praticamente iguais nos três cenários estudados. Comparando a porcentagem de violação realizada pela MSIAllocator com a política que violou menos essa métrica (ABS), tem-se uma diferença média de 0,02%, quando utilizado os dados do PlanetLab.

Esta violação está diretamente relacionada a quantidade de migrações efetuadas, ou seja, quanto mais migrações uma política realizar, maior a probabilidade de ocasionar uma violação de SLA. Tal relação pode ser comprovada comparando as Figuras 13 e 15. Note que na primeira figura, a política LAG realizou mais migrações que todas as outras, e a política CAL foi a segunda que realizou mais migrações, ao passo que as demais políticas ficaram com valores proporcionais. Na segunda imagem, o mesmo padrão se repete: a politica que mais violou essa métrica foi a lago e a segunda que mais violou foi a calavecchia, ao passo que as demais políticas ficaram com um valor semelhante.

Perceba também, que na primeira figura, a quantidade de migrações realizadas pelas políticas na simulação, utilizando os dados do PlanetLab, foi maior que na simulação utilizando os outros dois conjuntos de dados. Por isso as políticas violaram mais a métrica PDM utilizando o conjunto de dados do PlanetLab. Utilizando o conjunto de dados oriundos do GoogleCluster, a quantidade de migrações foi bem mais baixa que utilizando os outros dois conjuntos de dados.

Na Figura 15, nenhuma política violou a métrica PDM quando a simulação utilizou os dados de carga de trabalho do GoogleCluster, tal fenômeno ocorreu porque a quantidade de migrações efetuadas na simulação não foram suficientes para que a porcentagem de violação de SLA fosse significativa.

Figura 15 – Violações PDM em cada instância de dados



Fonte: próprio autor

5.6 Resultados dos testes do cenário 2

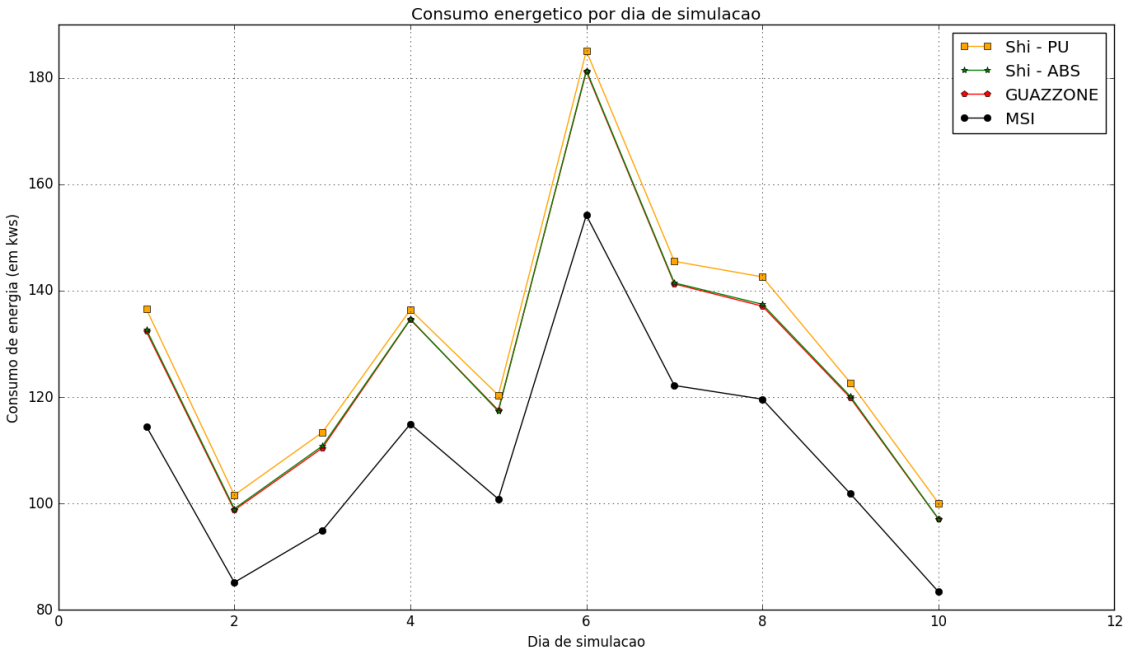
Esta sessão irá analisar os resultados das quatro políticas que apresentaram os menores consumos de energia com base na utilização de CPU (GUA, PER, ABS e MSI) dos servidores quando submetidas aos 10 dias do conjunto de dados propostos por Beloglazov e Buyya (2012) e coletados diretamente PlanetLab. Para análise serão consideradas as mesmas métricas do Cenário 1: O consumo energético, a quantidade de migrações, a violação da métrica SLATAH e PDM.

5.6.1 Consumo de energia

Analisando a métrica de consumo energético, percebe-se, através do gráfico apresentado na Figura 16, que, em todos os dias, a política MSIAllocator apresentou melhores resultados que as três melhores políticas comparadas por Mann e Szabó (2017). A economia média comparando-a com a segunda e a última política mais econômica (GUA e PERC) foi de, respectivamente, 17,85 kWh e 21,27 kWh. Os resultados obtidos demonstram que o consumo energético de cada política obedeceu um padrão durante os 10 dias de simulação, por exemplo, a política que consumiu menos energia no primeiro dia, também consumiu menos energia nos

outros dias. Tal resultado comprova a eficiência da política MSIAllocator em relação a economia de energia.

Figura 16 – Consumo energético em cada dia de simulação

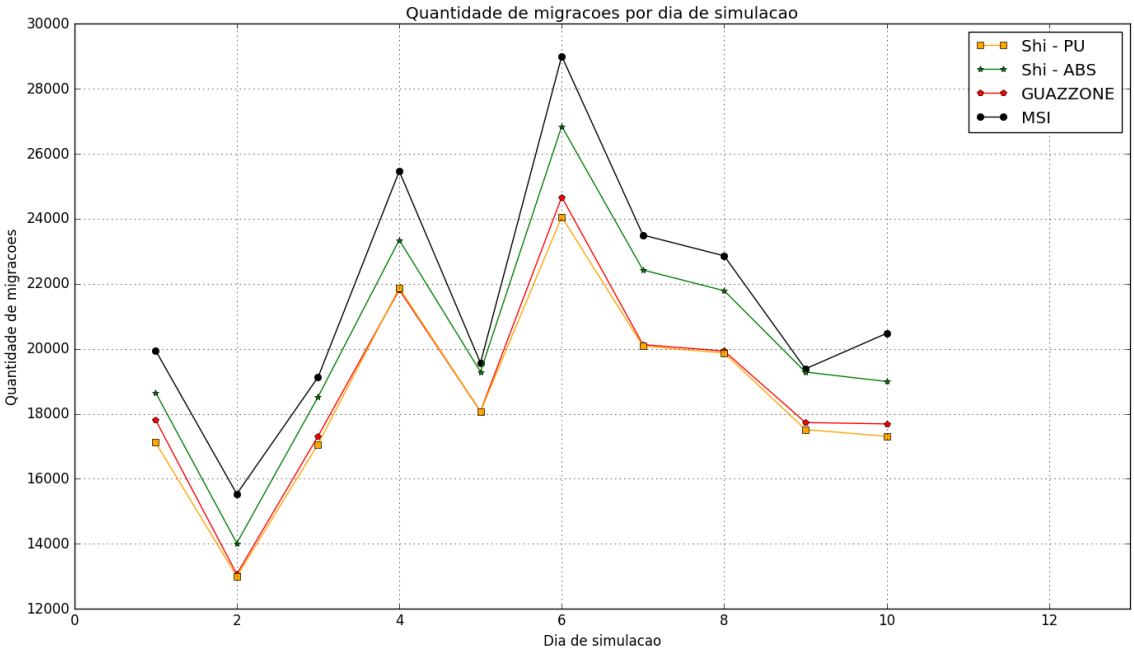


Fonte: próprio autor

5.6.2 Quantidade de migrações

Analisando a quantidade de migrações realizadas em cada dia de simulação, percebe-se através da Figura 17, que a política MSIAllocator foi pior as que três políticas supracitadas, apesar disso, seus resultados podem ser considerados intermediários. A diferença média das migrações realizadas pela MSIAllocator e a política que realizou menos migrações (PER) é de apenas 2891. Já em comparação com a penúltima política que realizou menos migrações (ABS), em média, a MSIAllocator efetuou apenas 1169 migrações a mais. Os resultados obtidos demonstram também que a quantidade de migrações que cada política efetuou obedeceu um padrão durante os 10 dias de simulação, por exemplo, a política que realizou menos migrações no primeiro dia, também realizou menos migrações nos outros dias.

Figura 17 – Migrações realizadas em cada dia de simulação

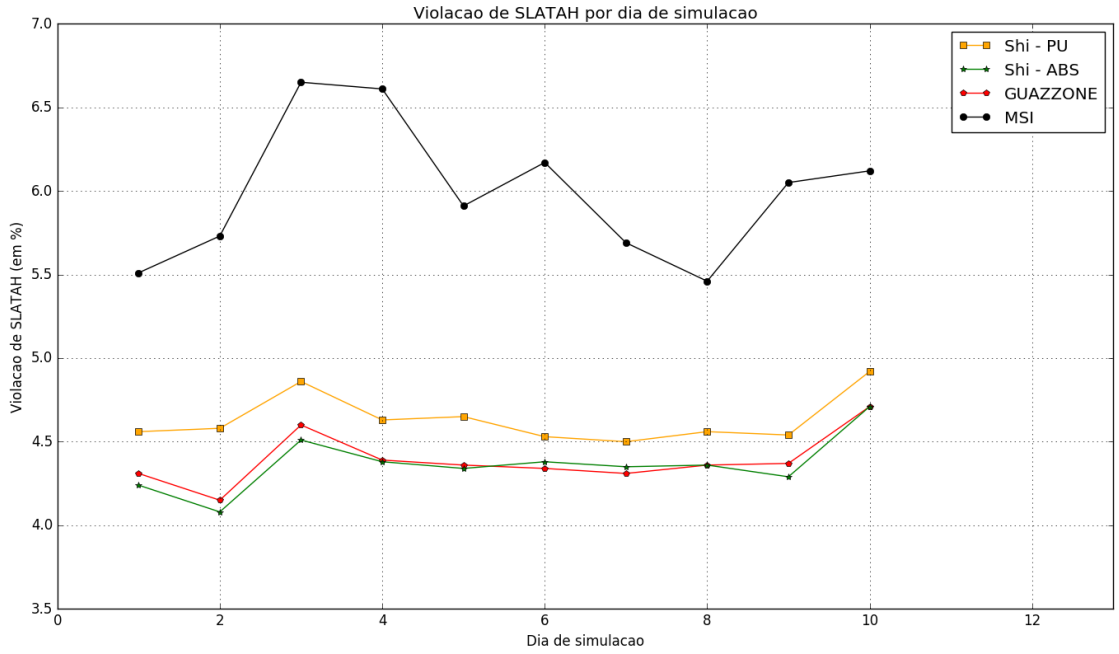


Fonte: próprio autor

5.6.3 Métrica SLATAH

Esta métrica avaliadora foi a única, dentre as quatro, que se mostrou completamente desfavorável a política MSIAllocator, como pode ser observado na Figura 18. A proposta deste trabalho busca consolidar as máquinas virtuais nos servidores para economizar energia, tal ação tende a aumentar a utilização dos servidores, facilitando a violação da SLATAH. A diferença média entre a porcentagem de violações SLATAH realizadas pela MSIAllocator e a política que apresentou a menor violação (ABS) foi de 1,63%. Os resultados obtidos demonstram que a porcentagem de violação da métrica SLATAH, causado por cada política, não obedeceu um padrão durante os 10 dias de simulação, por exemplo, a política que violou menos no primeiro dia, não causou necessariamente menos violação nos outros dias. Tal análise demonstra que considerar apenas um dia de simulação não é uma alternativa que obtém resultados concisos.

Figura 18 – Violações SLATAH em cada dia de simulação

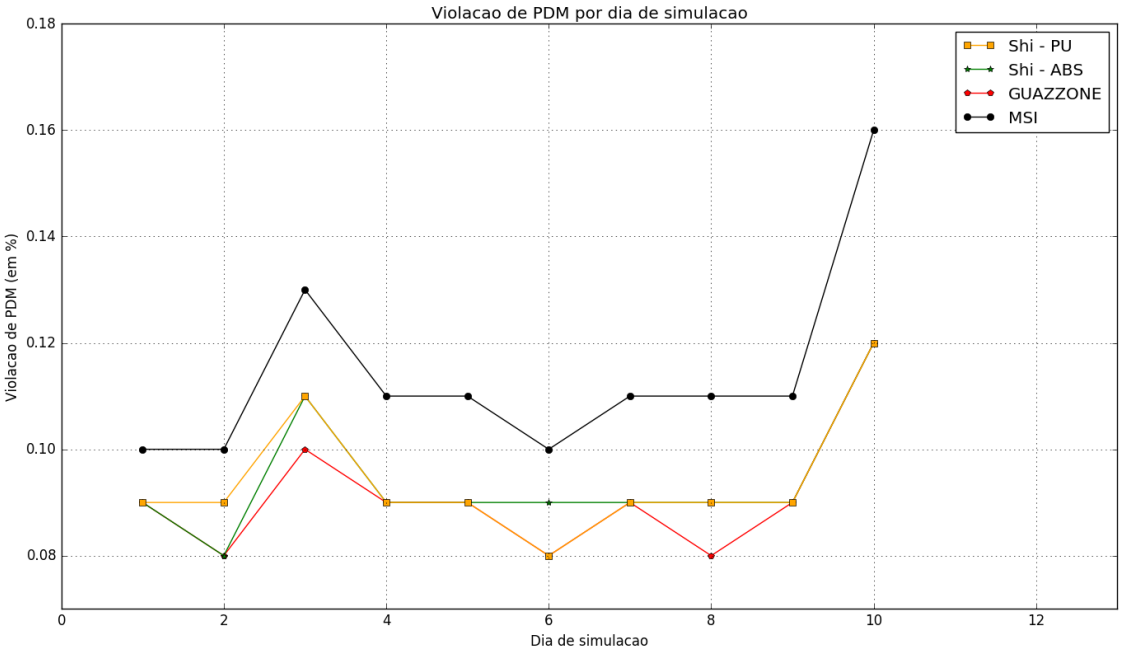


Fonte: próprio autor

5.6.4 Métrica PDM

Analisando a métrica PDM, percebe-se, através do gráfico apresentado na Figura 19 que a política MSIAllocator pior que as outras três. A diferença média da porcentagem de violação PDM realizadas pela MSIAllocator e a política que apresentou a menor violação (GUA) foi de 0,02%. Os resultados obtidos demonstram que a porcentagem de violação da métrica PDM, causado por cada política, não obedeceu um padrão durante os 10 dias de simulação, por exemplo, a política que violou menos no primeiro dia, não causou necessariamente menos violação nos outros dias. Mais uma vez, tal análise demonstra que considerar apenas um dia de simulação não é uma alternativa que obtém resultados concisos.

Figura 19 – Violações PDM em cada dia de simulação



Fonte: próprio autor

6 CONCLUSÃO

Este trabalho definiu uma política de alocação de máquinas virtuais, denominada de MSIAllocator, com o objetivo de economizar energia elétrica de um datacenter em um ambiente de Computação em Nuvens. Diferente das outras políticas analisadas neste trabalho, a MSIAllocator utilizou uma Meta-Heurística ILS para determinar o posicionamento das máquinas virtuais. A contribuição deste trabalho foi, além de desenvolver uma nova política de alocação, compará-la com outras políticas existentes no estado da arte utilizando diferentes cenários baseados em ambientes reais.

A utilização de diferentes ambientes de testes e a configuração baseada em ambientes reais, dão confiança e segurança aos resultados obtidos. A análise desses resultados demonstra que o presente trabalho alcançou seu objetivo principal ao minimizar o consumo de energia elétrica. A política proposta economizou mais energia que todas as outras políticas. No Cenário 2, a MSIAllocator economizou 17,85 kWh em comparação a segunda política que consumiu menos energia. Por consequência do *tradeoff* natural entre economia de energia e perca de desempenho ocasionado pela consolidação de máquinas virtuais, a política MSIAllocator apresentou o pior resultado dentre todas as políticas analisadas em relação a métrica SLATAH. Contudo, vale ressaltar, que a política proposta neste trabalho violou apenas 1.6% mais do que a política que obteve o melhor resultado (política proposta por Shi *et al.* (2013)), esse percentual de violação está próximo ao das demais políticas. Tal comportamento é previsível, uma vez que a economia de energia apresentada justifica a perda de desempenho. Já em relação a quantidade de migrações e na métrica PDM, no Cenário 2 a MSIAllocator obteve valores intermediários e satisfatórios de 21486 e 0,11%, respectivamente.

Como trabalhos futuros, podem ser realizadas melhorias na função de aceitação e utilizar uma ferramenta para determinar qual o melhor parâmetro possível para o número de iterações da Meta-Heurística ILS, como, por exemplo, o iRace (IRACE, 2016); Comparar o desempenho da MSIAllocator com outras políticas que também implementem Meta-Heurísticas; Analisar o impacto financeiro que as políticas de alocação trazem ao Provedor de Serviço, isto é, se o controle de gastos com a energia realmente representa uma economia quando confrontado com as possíveis multas aplicadas por violações de SLA; A fim de alcançar o objetivo deste trabalho, podem ser considerados também a utilização da memória RAM, memória de disco e largura de banda para determinar o posicionamento das máquinas virtuais, e não considerar apenas o consumo de CPU, ou seja, tornar a política multi-critério. Uma vez que a alocação é

baseada apenas na utilização de CPU, máquinas virtuais que demandam pouca CPU e muita memória, poderão concorrer muito por recursos físicos, violando demasiadamente a SLA; Analisar o impacto financeiro que as políticas de alocação trazem ao Provedor de Serviço, isto é, se o controle de gastos com a energia realmente representa uma economia quando confrontado com as possíveis multas aplicadas por violações de SLA.

REFERÊNCIAS

ALHARBI, F.; TAIN, Y. C.; TANG, M.; SARKER, T. K. Profile-based static virtual machine placement for energy-efficient data center. In: IEEE. **High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2016 IEEE 18th International Conference on**. [S.l.], 2016. p. 1045–1052.

ALTOMARE, A.; CESARIO, E. A data-driven approach based on auto-regressive models for energy-efficient clouds. In: IEEE. **Cluster, Cloud and Grid Computing (CCGRID), 2017 17th IEEE/ACM International Symposium on**. [S.l.], 2017. p. 1062–1069.

AMARANTE, S. R. M. **Utilizando o problema de múltiplas mochilas para modelar o problema de alocação de máquinas virtuais em computação nas nuvens**. 2013.

AMAZON. 2018. Disponível em: http://aws.amazon.com/pt/ec2/. Acesso em: 06 nov 2018.

BELOGLAZOV, A.; ABAWAJY, J.; BUYYA, R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. **Future generation computer systems**, Elsevier, v. 28, n. 5, p. 755–768, 2012.

BELOGLAZOV, A.; BUYYA, R. 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.**, v. 24, n. 13, set. 2012.

BORGES, H. P.; SOUZA, J. N. de; SCHULZE, B.; MURY, A. R. **Computação em Nuvem**. 2011.

BUYYA, R.; BELOGLAZOV, A.; ABAWAJY, J. Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges. **arXiv preprint arXiv:1006.0308**, 2010.

CABRAL, L. dos A. F. **Introdução à Pesquisa Operacional**. 1996.

CALCAVECCHIA, N. M.; BIRAN, O.; HADAD, E.; MOATTI, Y. Vm placement strategies for cloud scenarios. In: IEEE. **Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on**. [S.l.], 2012. p. 852–859.

CALHEIROS, R. N.; RANJAN, R.; BELOGLAZOV, A.; ROSE, C. A. D.; BUYYA, R. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. **Software: Practice and experience**, Wiley Online Library, v. 41, n. 1, p. 23–50, 2011.

CARISSIMI, A. Virtualização: Princípios básicos e aplicações. **Minicurso da 9ª Escola Regional de Alto Desempenho-ERAD**, p. 39–69, 2009.

CHOWDHURY, M. R.; MAHMUD, M. R.; RAHMAN, R. M. Study and performance analysis of various vm placement strategies. In: IEEE. **Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2015 16th IEEE/ACIS International Conference on**. [S.l.], 2015. p. 1–6.

FAN, X.; WEBER, W.-D.; BARROSO, L. A. Power provisioning for a warehouse-sized computer. In: ACM. **ACM SIGARCH Computer Architecture News**. [S.l.], 2007. v. 35, n. 2, p. 13–23.

GLOVER, F. W.; KOCHENBERGER, G. A. **Handbook of metaheuristics**. [S.l.]: Springer Science & Business Media, 2006. v. 57.

GUAZZONE, M.; ANGLANO, C.; CANONICO, M. Exploiting vm migration for the automated power and performance management of green cloud computing systems. In: SPRINGER. **International Workshop on Energy Efficient Data Centers**. [S.l.], 2012. p. 81–92.

IRACE. 2016. Disponível em: http://iridia.ulb.ac.be/irace/. Acesso em: 18 dez 2018.

KAPLAN, J. M.; FORREST, W.; KINDLER, N. **Revolutionizing data center energy efficiency**. [S.l.], 2008.

LAGO, D. G. d.; MADEIRA, E. R.; BITTENCOURT, L. F. Power-aware virtual machine scheduling on clouds using active cooling control and dvfs. In: ACM. **Proceedings of the 9th International Workshop on Middleware for Grids, Clouds and e-Science**. [S.l.], 2011. p. 2.

MANN, Z. A.; SZABÓ, M. Which is the best algorithm for virtual machine placement optimization? **Concurrency and Computation: Practice and Experience**, Wiley Online Library, v. 29, n. 10, p. e4083, 2017.

MARTELLO, S.; TOTH, P. **Knapsach problems: Algorithms and Computer Implementations**. [S.l.]: John Wiley Sons, Inc. New York, NY, USA ©1990, 1990. ISBN 0-471-92420-2.

MATOS, F. F. dos Santos Brasil de. **VBALANCE: Uma política de seleção de máquinas virtuais para balanceamento de cargas em cloud computing**. 2015.

NASIM, R.; KASSLER, A. J. A robust tabu search heuristic for vm consolidation under demand uncertainty in virtualized datacenters. In: IEEE. **Cluster, Cloud and Grid Computing (CCGRID), 2017 17th IEEE/ACM International Symposium on**. [S.l.], 2017. p. 170–180.

NASIM, R.; TAHERI, J.; KASSLER, A. J. Optimizing virtual machine consolidation in virtualized datacenters using resource sensitivity. In: IEEE. **Cloud Computing Technology and Science (CloudCom), 2016 IEEE International Conference on**. [S.l.], 2016. p. 168–175.

OLIVEIRA, L. de. **O problema do corredor de comprimento mínimo : algoritmos exatos, aproximativos e heurísticos**. 2012.

POZO, D. A. **Implementação da Metaheurística GRASP para o Problema do Caixeiro Viajante Simétrico**. 2008.

REDES de Computadores. 2001. Disponível em: http://redesa2001rj.tripod.com/dicas/redes_de_computadores.htm. Acesso em: 30 jun 2018.

SALLES, A. C.; ALVES, A. P. F.; DOLCI, D. B.; LUNARDI, G. L. Tecnologia da informação verde: um estudo sobre sua adoção nas organizações. **RAC-Revista de Administração Contemporânea**, Associação Nacional de Pós-Graduação e Pesquisa em Administração, v. 20, n. 1, 2016.

SHARMA, N. K.; REDDY, G. R. M. Novel energy efficient virtual machine allocation at data center using genetic algorithm. In: IEEE. **Signal Processing, Communication and Networking (ICSCN), 2015 3rd International Conference on**. [S.l.], 2015. p. 1–6.

SHI, L.; FURLONG, J.; WANG, R. Empirical evaluation of vector bin packing algorithms for energy efficient data centers. In: IEEE. **Computers and Communications (ISCC), 2013 IEEE Symposium on**. [S.l.], 2013. p. 000009–000015.

TAHERI, J.; ZOMAYA, A. Y.; KASSLER, A. vmbbprofiler: a black-box profiling approach to quantify sensitivity of virtual machines to shared cloud resources. **Computing**, Springer, v. 99, n. 12, p. 1149–1177, 2017.