MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE MESTRADO EM SISTEMAS E COMPUTAÇÃO

FELIPE JULIANI FERNANDES

UM MÉTODO DE ESCALONAMENTO BASEADO NO COMPORTAMENTO DE APLICAÇÕES HPC PARA NUVENS COMPUTACIONAIS BALANCEANDO DESEMPENHO E EFICIÊNCIA ENERGÉTICA

Rio de Janeiro
2015

INSTITUTO MILITAR DE ENGENHARIA

FELIPE JULIANI FERNANDES

UM MÉTODO DE ESCALONAMENTO BASEADO NO COMPORTAMENTO DE APLICAÇÕES HPC PARA NUVENS COMPUTACIONAIS BALANCEANDO DESEMPENHO E EFICIÊNCIA ENERGÉTICA

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Sistemas e Computação.

Orientador: Profª. Raquel Coelho Gomes Pinto - D.Sc.
Co-orientador: Prof. Bruno Richard Schulze - D.Sc.

Rio de Janeiro
2015

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80-Praia Vermelha
Rio de Janeiro-RJ CEP 22290-270

005.1 Fernandes, Felipe Juliani

F363m Um método de escalonamento baseado no comportamento de aplicações HPC para nuvens computacionais balanceando desempenho e eficiência energética/ Felipe Juliani Fernandes, orientado por Raquel Coelho Gomes Pinto, Bruno Richard Schulze – Rio de Janeiro: Instituto Militar de Engenharia, 2015.

102 p.: il., tab.

Dissertação (mestrado) – Instituto Militar de Engenharia – Rio de Janeiro, 2015.

1. Curso de Sistemas e Computação – teses e dissertações. 2. Aplicações HPC. 3. Escalonamento de VMs. 4. Nuvens computacionais. I. Pinto, Raquel Coelho Gomes. II. Schulze, Bruno Richard. III. Título. IV. Instituto Militar de Engenharia.

INSTITUTO MILITAR DE ENGENHARIA

FELIPE JULIANI FERNANDES

UM MÉTODO DE ESCALONAMENTO BASEADO NO COMPORTAMENTO DE APLICAÇÕES HPC PARA NUVENS COMPUTACIONAIS BALANCEANDO DESEMPENHO E EFICIÊNCIA ENERGÉTICA

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Sistemas e Computação.

Orientador: Profª. Raquel Coelho Gomes Pinto - D.Sc.

Co-orientador: Prof. Bruno Richard Schulze - D.Sc.

Aprovada em 27 de Agosto de 2015 pela seguinte Banca Examinadora:

Profª. Raquel Coelho Gomes Pinto - D.Sc. do IME - Presidente

Prof. Bruno Richard Schulze - D.Sc. do LNCC

Prof. Anderson Fernandes Pereira dos Santos - D.Sc. do IME

Prof. Fabio Lopes Licht - D.Sc. da UCP

Rio de Janeiro
2015

Aos meus pais, professores e amigos.

AGRADECIMENTOS

Agradeço aos meus pais, Italia e Carlos por todo o amor, carinho e suporte que tive, não só durante essa fase de estudos, mas por todas as fases da minha vida em que pude tê-los ao meu lado.

À minha família, meu irmão Danilo, meus tios e primos (até os que moram longe e eu sempre fui visitar em busca de tranquilidade), que, todos às suas maneiras contribuíram para me manter focado no objetivo da conclusão desse ciclo.

À Vivi, que durante todo esse período suportou meus momentos de impaciência e falta de atenção, mas nunca deixou de me doar seu amor, carinho e companhia em todos os momentos.

Ao amigo David da Universidade Federal Rural de Pernambuco (UFRPE), que mesmo distante, sem seu apoio, ideias, comprometimento e também momentos de descontração não conseguiria executar este trabalho.

Aos meus amigos que me distraíram, por vezes, em reuniões, viagens e festas me fazendo tomar folego para continuar em frente em momentos complicados.

Agradeço a professora Raquel do Instituto Militar de Engenharia (IME) pelas orientações, trocas de conhecimento e sugestões, e principalmente pelo apoio e compreensão das minhas impossibilidade de estar presente em reuniões, por vezes, durante o desenvolvimento desse trabalho.

Não poderia deixar de citar também os professores do IME, Duarte, Anderson e Salles que acompanharam o trabalho durante a maior parte do tempo, sempre sugerindo e fazendo boas críticas para a melhoria do mesmo.

Ao professor Bruno do Laboratório Nacional de Computação Científica (LNCC) por, além de me orientar neste trabalho, ser um grande amigo e ter me inserido no mundo acadêmico por meio do grupo Computação Científica Distribuída (ComCiDis) no LNCC, onde adquiri não só conhecimentos, mas também boas amizades, lembranças e experiências que serviram de base para o ingresso e andamento do curso de mestrado no IME.

Ao professor Mury do grupo ComCiDis, que em momentos antes do início desse mestrado, ainda no LNCC, me proporcionou inúmeras ideias, possibilidades e projetos. Apesar de algumas discordâncias, sempre foi um entusiasta, acreditando em mim e me incentivando.

À todo o grupo ComCiDis, pelos bons momentos em trabalhos, congressos e descontrações no laboratório.

Por fim, aos alunos, colegas de sala e amigos que durante todo o empo foram de extrema importância quanto a superação dos momentos árduos enfrentados. Sendo assim também, à todos os professores e funcionários da Seção de Engenharia da Computação (SE/8) do IME.

Muito obrigado!

Felipe Juliani Fernandes

"A imaginação é mais importante que o conhecimento."
Albert Einstein

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

CO2	-	*Dióxido de Carbono*
CAPEX	-	*Capital Expenditure*
CPU	-	*Central Processing Unit*
DOE	-	*United States Department od Energy*
DVFS	-	*Dynamic Voltage and Frequency Scaling*
GB	-	*Giga Byte*
HPC	-	*High Performance Computing*
HPG	-	*Highest Potential Growth*
HPL	-	*High Performance Linpack*
IaaS	-	*Infrastructure-as-a-Service*
IAEE	-	*Inteligent Agent for Energy Estimation*
IO	-	*In/Out*
IPMI	-	*Intelligent Platform Management Interface*
KVM	-	*Kernel-based Virtual Machine*
kWh	-	*Kilo-watt hour*
LANL	-	*Los Alamos National Laboratory*
vCPU	-	*Virtual Central Processing Unit*
MB	-	*Mega Byte*
MIPS	-	*Million Instructions Per Second*
MM	-	*Minimization of Migrations*
NIST	-	*National Institute of Standards and Technology*
NPA	-	*Non Power-Aware*
OCCI	-	*Open Cloud Computing Interface*
PaaS	-	*Platform-as-a-Service*
PUE	-	*Power Usage Effectiveness*
QoS	-	*Quality of Service*
RAM	-	*Random Access Memory*
SaaS	-	*Software-as-a-Service*
SLA	-	*Service Level Agreement*

SO	-	*Sistema Operacional*
ST	-	*Single Threshold*
TDP	-	*Thermal Design Power*
TI	-	*Tecnologia da Informação*
UPS	-	*Uninterruptible Power Supply*
VM	-	*Virtual Machine*
W	-	*Watt*

RESUMO

Neste trabalho é apresentado o processo de desenvolvimento e validação de um método de escalonamento de VMs em nuvens computacionais focadas em HPC. O escalonador desenvolvido leva em consideração o tipo de carga de trabalho que as VMs irão executar (CPU-*bound* ou IO-*bound*) para decidir quando e em qual servidor as mesmas serão alocadas. O escalonador é capaz de reduzir o consumo energético do ambiente e a ocorrência de violações de SLA, sem reduzir o desempenho das VMs, ao alocar de maneira simultânea VMs que executam tarefas CPU-*bound* e outras que executam tarefas IO-*bound* de forma a consolidar recursos, aproveitando cada componente de *hardware* de cada servidor para seus devidos fins. Antes da validação de desempenho do escalonador, mediante uso do simulador CloudSim, foram conduzidos testes de desempenho com *benchmarks* sintéticos, representativos de cada tipo de aplicação. Assim, foi possível implementar um método que utiliza uma política de alocação de VMs baseada nas particularidades de desempenho identificadas durante os experimentos. Os resultados obtidos indicam que analisar as especificidades das infraestruturas e aplicações contribui para melhorar o aproveitamento dos recursos, aumentar os níveis de oferta de serviço e reduzir os problemas causados pelo uso dos recursos de forma concorrente.

ABSTRACT

In this work is shown the development and validation processes about a VM scheduler method in a HPC cloud environment. This scheduler considers the workload type (CPU or IO-bound) that the VMs will run to decide when and in which a host will be allocated. The scheduler is able to reduce the energy consumption of the environment and SLA violations without imply in performance losses in the VMs, allocating simultaneously VMs running CPU-bound tasks and IO-bound tasks, aiming a maximum resource consolidation, in level of hardware, as processor cores and IO devices components for its specific purposes. Before the scheduler performance validation through CloudSim simulation, two synthetic benchmarks that represents two application types were run in some VMs. From the observation of these experiments was possible to implement a method that uses a VM allocation policy based on application features during experiments. Therewith the obtained results shown that analysis of the infrastructure specifications and its characteristics might contributes to better resource usage, increase level services offers and reduce the problems caused by competition in resource usage.

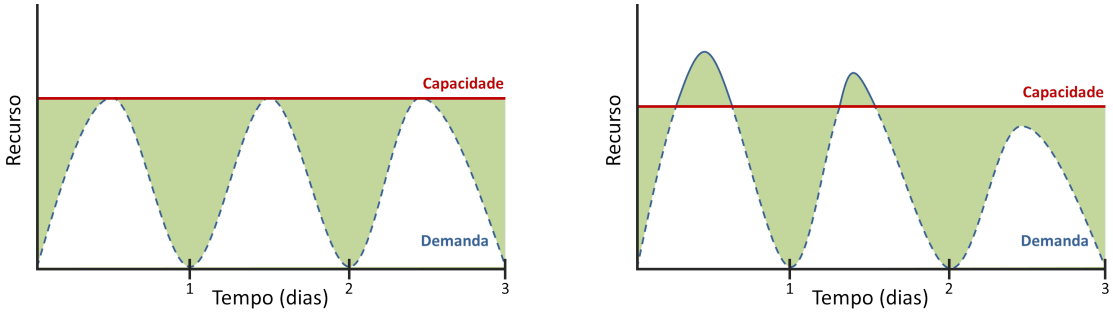
1 INTRODUÇÃO

Com a crescente demanda de utilização de recursos computacionais, seja pelas diversas necessidades dos usuários finais ou pelas necessidades dos fornecedores em garantir e expandir os serviços oferecidos, é possível constatar a importância do paradigma das Nuvens Computacionais e de suas tecnologias envolvidas no dia-a-dia das pessoas e das organizações industriais, comerciais e acadêmicas. Este modelo permite que seus consumidores mantenham basicamente uma infraestrutura composta por dispositivos de acesso e uma rede para a Internet, delegando aos provedores a responsabilidade de manter a infraestrutura computacional e os aplicativos e serviços em execução. Além da redução de gastos com equipamentos, corpo técnico e energia pelo lado dos consumidores, ainda pode-se ganhar com a possibilidade de uso de recursos computacionais "ilimitados", suporte especializado e garantias oferecidas pelos fornecedores em manter os serviços em produção.

Uma das grandes vantagens das nuvens computacionais é a possibilidade de oferecer infraestrutura para Computação de Alto Desempenho (HPC) (ROUSE, 2007a) à pessoas e organizações que necessitam desse tipo de recurso, porém que não possuam espaço físico, recursos financeiros ou capacitação técnica e operacional para implementar e manter infraestrutura própria destinada à HPC. Isso abre possibilidade para que até mesmo pequenas *startups* tenham acesso a um grande poder computacional sem necessitar manter um *datacenter* próprio que, além das demandas já citadas, apresenta elevado consumo energético, aumentando o ônus para a empresa e prejudicando o meio ambiente. Ao utilizar a infraestrutura hospedada em nuvem, essa responsabilidade é transferida a provedores especializados.

Nesse sentido, a questão energética precisa de atenção, pois *datacenters* são grandes vilões quando se pensa em sustentabilidade na área da Tecnologia da Informação (TI). É necessária grande carga de energia para mantê-los sempre operando de forma eficiente. Um típico *datacenter* com 1000 *racks* necessita de 10 Megawatts de energia para operar, segundo (GARG, 2011). O custo energético é um atributo significativo para essas organizações provedoras destes serviços, portanto tratá-lo é fundamental para reduzir gastos. Um exemplo simples dessa problemática pode ser encontrado em *datacenters* tradicionais,

que por exemplo mantém *websites*. Pela natureza desses ambientes, os mesmos acabam tendo picos esporádicos de acesso. Nesse caso, isso significa que o uso dos recursos computacionais é baixo e o desperdício de energia é existente devido à ociosidade em alguns períodos. Em outros momentos, o atendimento à demanda também pode ser afetado visto que a quantidade de recursos computacionais é fixa nesses ambientes clássicos. Na Figura 1.1 é ilustrada essa problemática.



(a) Excesso de provisionamento

(b) Sub-provisionamento

FIG. 1.1: Picos de carga em um *datacenter*

O eixo x representa o passar do tempo (quantidade de dias), o eixo y a quantidade de recursos demandada. A linha vermelha representa a capacidade do *datacenter*, ou seja, o que o mesmo possui de recursos para atender a demanda, está é representada pela linha tracejada sinuosa.

A demanda por recursos varia de acordo com o tempo e, por vezes, pode não utilizar a capacidade total dos recursos, causando prejuízo devido a ociosidade, como representado na Figura 1.1a. Já em outros momentos, pode até ultrapassar a capacidade máxima do *datacenter*, causando ao demandante problemas por conta da falta de recursos necessários, como representado na Figura 1.1b.

Os sistemas computacionais tradicionalmente eram desenvolvidos com foco em desempenho e a eficiência energética não era onsiderada. Porém, com o incremento das tecnologias relacionadas as redes móveis promovendo acesso a dados e internet de alta velocidade, os dispositivos móveis entraram em evidência e o desenvolvimento de sistemas, ferramentas e aplicações passou a ter que considerar essa questão energética em paralelo à questão do desempenho. Essa questão tornou-se prioritária, devido à necessidade de aumento na autonomia das baterias desses dispositivos, entretanto, por natureza, existe um conflito entre essas duas questões: quanto mais desempenho, mais demanda de energia é necessária.

Quanto aos *datacenters* essas questões também tem grande importância, pois do ponto de vista dos provedores, a redução do consumo energético é essencial para a redução de gastos com o mesmo. Todavia, a mobilidade delegou ainda mais consumo à recursos remotos, que justamente vêm desses ambientes, ou seja, a demanda por recursos em *datacenter* cresceu junto com a computação móvel. Logo um dos grandes desafios atuais da área computacional é fornecer serviços e ferramentas que consigam balancear estas duas propriedades, desempenho e eficiência energética, não onerando nem usuário e nem provedor.

A computação distribuída, seja em grade, *cluster* ou nuvem, tem a característica de grande concentração de equipamentos nos *datacenters*, isso evidencia os custos com energia, tanto em termos econômicos como ambientais. Com isso, a aplicação das tecnologias e conceitos desenvolvidos para a computação móvel, atualmente está evidenciada também em todos os equipamentos de TI e para a infraestrutura em geral.

Com o modelo das nuvens computacionais é possível obter maior eficiência energética, pois o mesmo utiliza técnicas como a virtualização e a consolidação de carga de trabalho. Com o melhor controle dos recursos computacionais em relação à demanda, a nuvem se mostra um modelo capaz de lidar com as questões energéticas e também equalizar a energia de acordo com as reais necessidades, evitando o desperdício. Destaca-se também a possibilidade de implantação de redundância de servidores de maneira mais fácil (RAMBHADJAN, 2010).

1.1 MOTIVAÇÃO

Atualmente discute-se sobre as questões de sustentabilidade e consciência ambiental no meio corporativo. As grandes organizações passaram a ter maior comprometimento com a responsabilidade social e com a sustentabilidade, devido à legislação mais restritiva e à necessidade de manter uma imagem positiva perante a sociedade (WESTPHALL, 2013). O esforço referente à redução dos impactos ecológicos durante todo o processo de produção, utilização e descarte de produtos e serviços na área da TI é reconhecido como atitude sustentável. Quando a TI tem comprometimento com essas ações, então ela passa a ser conhecida como TI Verde. Nessa direção, algumas agências federais e organizações comerciais americanas começaram a se beneficiar com o modelo da computação em nuvem. O Departamento de Energia Norte Americano (*United State Department of Energy* - DoE), por exemplo, e suas agências afiliadas investiram esforços na construção de uma

plataforma híbrida desenvolvida pelo Los Alamos National Laboratory (LANL) em colaboração com a indústria, chamada YOURCloud, cujo papel é ser um agente de nuvem (*cloud broker*) (DOE, 2012).

Um *cloud broker* é uma entidade em uma nuvem computacional que possui funções de gerenciamento de uso, desempenho e entrega dos serviços, bem como negociação de relacionamento entre os provedores e consumidores, facilitando a interação do usuário com o ambiente (MELL, 2011). A plataforma YOURcloud age como um único plano de controle em ambos, nuvens privadas e provedores de serviços em nuvens comerciais, dando aos usuários um ponto central para gerenciar servidores e serviços. O LANL teve como objetivo desenvolver o ambiente de nuvem pra consolidar *datacenters*, acelerar o provisionamento de recursos e assumir uma postura de TI Verde. Com essa postura foi possível desativar 105 servidores físicos e desligar três *datacenters* (DOE, 2012) (MOORE, 2013).

A computação em nuvem passou a ser largamente adotada por conta de suas inúmeras vantagens econômicas e possibilidades de utilização de recursos remotos em abundância. Por meio das suas características, esse modelo possibilitou melhorar a eficiência energética quanto ao consumo dos serviços e à gerência dos *datacenters* e, desta forma, isto está diretamente ligado aos conceitos de TI Verde. As nuvens computacionais que são mantidas com foco no monitoramento da eficiência energética são conhecidas como Nuvens Verdes (*Green Cloud Computing*) e atualmente esta abordagem está relacionado ao *datacenter* e ao gerenciamento de energia do mesmo. Em contrapartida um relatório do Greenpeace (GREENPEACE, 2013) aponta que as nuvens computacionais podem agravar o problema das emissões de carbono e o aquecimento global. O motivo é um aumento dramático na demanda por recursos computacionais nos próximos anos. O *datacenter* mais eficiente construído com as melhores taxas de utilização só mitigaria, em vez de eliminar, as emissões nocivas de carbono (GARG, 2011).

Apesar deste modelo possibilitar economias no custo com energia, os seus serviços aumentam a carga de informação nos *storages* e servidores de arquivos devido à sua característica de redundância ao se manter réplicas para atender especificações de segurança, desempenho e disponibilidade. Outro aumento observado é referente aos servidores de bancos de dados. Com sistemas que passam a ter que controlar mais informações de múltiplos usuários, o volume de dados nos bancos de dados cresce eminentemente. Por fim, mais um fator a ser destacado é o tráfego de rede constante devido à, por exemplo,

manipulação de grandes arquivos e à demanda de conexões remotas.

É preciso compreender ainda que existem pontos a serem avaliados além dos servidores que compõem parte da infraestrutura do *datacenter*. A comunicação é fortemente exigida e seu custo energético é passível de ser estudado para que os conceitos "verdes" sejam postos em prática de maneira integral, sob o aspecto desse paradigma computacional. Para isso o conceito de Rede Verde (*Green Networking*) surge como forma de esforço na ajuda da integração de *datacenters* energéticamente mais eficientes.

A motivação para o desenvolvimento deste trabalho se deu após observada a crescente demanda por grande capacidade computacional e a possibilidade concreta de sua oferta por meio da Computação em Nuvem, popularizando e facilitando o consumo dos recursos computacionais que, em contrapartida, aumenta a complexidade de gerenciamento e também o consumo de energia nos *datacenters* (BALIGA, 2011). Muitos trabalhos atuais abordam o tema traçando panoramas e sugerindo estratégias e soluções para o tratamento do Uso Efetivo de Energia (PUE) nos mesmos. E, de fato, cada vez mais utilizando recursos computacionais mais poderosos, esforços no sentido de minimizar o consumo de energia e os impactos ambientais neste setor são necessários.

1.2 DESCRIÇÃO DO PROBLEMA

Sabe-se que *datacenters* são grandes vilões quando trata-se de consumo energético. Entretanto, sua importância no mundo atual é impactante, pois cada vez mais serviços informatizados e responsabilidades diversas da área de TI são delegados e terceirizados, e é comum a utilização de ambientes de nuvens para alocar e executar projetos e tarefas vinculadas a esses serviços e responsabilidades. Tal fato ocorre pois o poder computacional e a especialização de organizações que detêm infraestrutura para atender essas necessidades agrega diminuição de custos e aumento de garantias aos projetos. Por outro lado, quando a demanda é por HPC ou por grandes massas de dados, esses fatores amplificam ainda mais o problema energético desses ambientes.

Diante disso, com o desenvolvimento desse trabalho busca-se a obtenção de eficiência energética em um *datacenter* que implemente uma nuvem computacional privada voltada para HPC, por meio de um método de escalonamento. Muitos estudos atualmente buscam melhorar a eficiência energética de *datacenters* tratando as influências que estão a margem dos sistemas computacionais que executam nos servidores como desligamento de servidores não utilizados, diminuição de utilização e potência de sistemas de refrigeração e também

dispositivos de *Uninterruptible Power Supply* (UPS) inteligentes. Todavia, neste estudo, a obtenção de melhores índices de energia está sendo tratado em um componente de *software* inerente a plataforma de nuvem, no caso um escalonador de máquina virtual (VM). Isso significa que nenhum dos outros esforços de obtenção de melhoria de eficiência energética precisa ser desconsiderado. Com este trabalho é possível que essas outras técnicas e ideias sejam agregadas.

Um dos grandes desafios de se desenvolver um escalonador que visa eficiência energética é que o mesmo não pode desconsiderar o desempenho das VMs, portanto esse escalonador e sua politica de alocação de VMs deve balancear a eficiência energética e o desempenho no ambiente de nuvem. Para que isso aconteça, o método tem como fundamento considerar as características de consumo energético das aplicações HPC nos servidores do *datacenter* para alocar os recursos e, assim então, executar as tarefas. Essa consideração envolve aplicações com duas características computacionalmente diferentes: intensas de CPU (CPU-*bound*) e intensas de IO (IO-*bound*).

Em sistemas operacionais, processos podem ser classificados como intensivos de CPU ou intensivos de IO, de acordo com sua utilização de processador e de dispositivos de IO (OLIVEIRA, 2004). O ideal em um sistema operacional é que o mesmo tenha suporte para um conjunto de processos CPU-*bound* e processos IO-*bound* de maneira balanceada. O que pode tornar a execução de aplicações tanto CPU-*bound* como IO-*bound* lentas é a má distribuição desses tipos de processo em um computador. Se muitos processos, de variadas aplicações forem do tipo CPU-*bound*, o processador será o gargalo do sistema e precisará processar todos eles, logo uma fila de processos precisará ser administrada pelo sistema operacional e, na medida que forem sendo terminados, os processos em espera começam a ser executados. Se todos forem do tipo IO-*bound*, o processador ficará parado enquanto todos os processos tentam acessar os periféricos, logo, o barramento desses periféricos será o gargalo e o mesmo que ocorrera com os processos CPU-*bound* ocorrerá, porém nos dispositivos de IO.

Neste trabalho, os pontos-chave abordado pelo método de escalonamento são as aplicações que podem ser classificadas como CPU-*bound* ou IO-*bound*, de acordo com a utilização do processador e dos dispositivos de IO. Uma aplicação CPU-*bound* utiliza muito tempo de CPU, ou seja, o seu tempo de execução é definido principalmente pelos *clocks* do processador. Já as aplicações IO-*bound* passam a maior parte do tempo no estado de espera no processador, pois realizam muitas operações de leitura e escrita. Nesse caso,

o tempo de execução é definido principalmente por estas operações. Na Figura 1.2 essas diferenças estão sendo ilustradas.

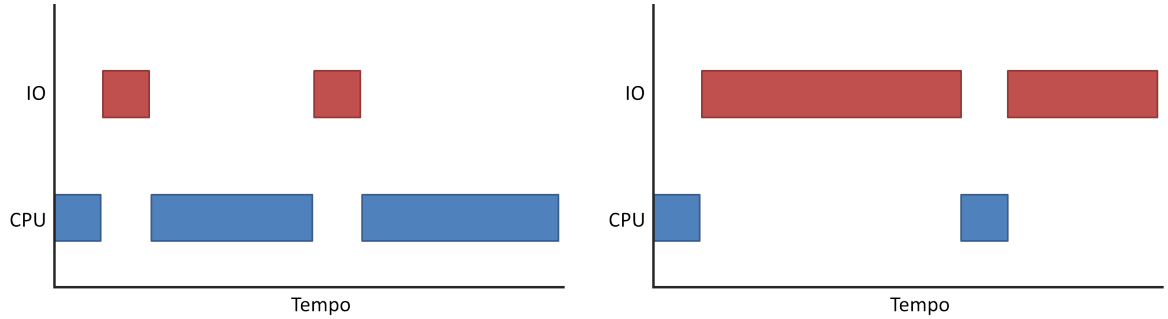


FIG. 1.2: Diferença entre aplicações CPU-*bound* e IO-*bound*

Como medida de obtenção dessa eficiência e até mesmo redução de custo, o método proposto baseou-se nas avaliações de características de aplicações HPC. Essas características são basicamente as necessidades computacionais relacionadas ao consumo de tempo de processador e de dispositivo de IO, ou seja, diferenças das aplicações intensivas de CPU e de IO. A ideia também é possibilitar a incorporação deste escalonador a qualquer projeto de *middleware* de nuvem computacional que vise atender a demanda de VMs com as diferenças computacionais expostas. Com isso espera-se adicionar a um ambiente de nuvem, um componente capaz de trazer eficiência energética por meio do escalonamento das instâncias das VMs que executem aplicações desses tipos.

Em aberto, existem questões que ainda precisam ser melhor avaliadas e que também são relacionadas ao consumo de energia. Essas questões têm relação com o posicionamento das instâncias e seus perfis de execução. Como exemplo, aplicações paralelas com intensiva troca de mensagem, se alocadas em recursos distantes, ainda que no mesmo *datacenter*, podem ocasionar maior consumo de energia na medida que as mensagens passam por um maior número de enlaces, como já comentado. Prever esse tipo de situação pode ocasionar em um melhor aproveitamento da energia que é despendida com a referida tarefa.

1.3 OBJETIVO

Atualmente o gerenciamento de energia é um dos pontos mais importante no contexto da Computação em Nuvem. Reduzir o uso de energia é um desafio complexo por conta das possibilidades que uma nuvem computacional oferece. A solicitação de aplicações, dados e recursos é intensa e de grande volume fazendo com que os servidores que atendem essas

requisições tenham a necessidade de processá-las rapidamente e em um curto período de tempo. A economia de energia pode ser alcançada por meio de várias estratégias como a consolidação de VMs de acordo com a utilização dos recursos, a equalização ou mesmo hibernação das CPUs quando os servidores estiverem desocupados, a realocação das cargas de trabalho ou VMs, dentre outras (BUYYA, 2010).

Buscando contribuir com um melhor consumo de energia em *datacenters* que forneçam HPC por meio de nuvens computacionais, de forma que esse consumo seja eficiente em relação a demanda, nesta proposta é exposta uma abordagem para melhor utilizar os recursos computacionais em relação ao consumo de energia de aplicações HPC, levando em conta os tipos de aplicações que serão instanciadas, por meio de um método de escalonamento de VMs.

Ultimamente esforços em pesquisas vêm se voltando para o aprimoramento da *Green Cloud Computing* por meio da utilização de técnicas, tais como equalização de frequência de CPUs, desligamento de servidores ociosos e controle de aparelhos refrigeradores (URGAONKAR, 2010) (BUYYA, 2010) (CHEN, 2010), bem como alocação e realocação inteligente de VMs (KARVE, 2006) (STEINDER, 2008). Além disso, outros trabalhos propõem modelos e estratégias que contribuem com eficiência de energia por meio da utilização de energias renováveis (HAQUE, 2013), além de escalonadores que focam no consumo de energia para executar tarefas e alocar recursos (GOIRI, 2011). Entretanto a instanciação de aplicações HPC e a alocação de recursos para uma computação de alto desempenho precisam ser melhor estudadas em função da busca dessa eficiência energética.

A instanciação de aplicações paralelas intensivas de CPU podem ter melhor eficiência energética se alocadas em recursos específicos, onde aconteça uma redução no tempo das trocas de mensagens, ou seja, se estiverem alocadas em núcleos de processamento próximos uns dos outros, aproveitando assim uma maior velocidade na troca de suas mensagens, devido a estarem se comunicando diretamente por um barramento veloz. Por outro lado, se o mesmo tipo de aplicação for erraticamente distribuído por servidores no *datacenter*, isso pode aumentar seu tempo de execução devido ao maior tempo despendido na troca de mensagens. Logo, aplicações que se comunicam devem ser alocadas em servidores próximos uns aos outros. Outros fatores como intenso acesso a memória e disco, bem como migrações de VMs (*live migration*) também podem prejudicar a questão energética se forem desconsideradas, ou seja, se não se tomar cuidado com a sua utilização em

relação aos tipos de aplicações. Distinguir aplicações CPU-*bound* de aplicações IO-*bound* é importante para determinar a melhor forma de alocar os recursos computacionais. Por conta disso, é preciso identificar suas diferenças em relação ao consumo energético para que se possa propor um método de escalonamento que tenha como objetivo melhorar a eficiência energética do ambiente. Estas hipóteses precisam ser verificadas para que se possa reduzir gastos com energia de tarefas secundárias, ou seja, que não sejam o foco da execução das aplicações.

Com este trabalho, então, pretendeu-se apresentar um método para a contribuição do PUE por meio de uma melhor utilização dos recursos computacionais em relação ao tipo de aplicação HPC executado sobre servidores que representem uma infraestrutura de nuvem computacional privada. A ideia do método é servir de referência para uma implementação de algoritmos que possam ser agregados a escalonadores de nuvem estimulando também o desenvolvimento nos padrões da *Open Cloud Computing Interface* (OCCI) (METSCH, 2011). Desenvolveu-se um método de escalonamento baseado no estudo da alocação, organização e execução de VMs com aplicações HPC que executam tarefas com características computacionais diferentes (como exemplificado acima, IO-*bound* e CPU-*bound*), considerando seus comportamentos e custos energéticos, e assim, estabelecendo uma estratégia de alocação para as mesmas.

Assim, nesse trabalho está sendo apresentado um algoritmo de escalonamento de VMs em um ambiente de nuvem computacional focado no provimento de recursos para HPC, otimizando a relação desempenho versus consumo energético, melhorando a execução de aplicações com determinados padrões computacionais. Seu método de alocação também visa minimizar as violações de *Service Level Agreement* (SLA). O algoritmo foi desenvolvido após a coleta de dados de consumo energético de servidores em função do tipo de aplicação executado nos mesmos, por VMs.

2 CONCEITOS BÁSICOS

Nesta seção, estão relacionados alguns conceitos aos quais este trabalho se contextualiza. São eles:

2.1 COMPUTAÇÃO DISTRIBUÍDA E DE ALTO DESEMPENHO

Para trabalhar sobre o modelo da Computação em Nuvem é importante compreender sua origem, para isso precisamos voltar aos conceitos da Computação Distribuída, área na qual insere-se a Computação em Nuvem.

Um sistema distribuído é uma coleção de computadores independentes que se apresenta ao usuário como um sistema único e consistente (TANENBAUM, 2003). Essa definição expõe a computação distribuída com o objetivo de disponibilizar poder computacional por meio de diversos computadores interligados por uma rede de computadores, ou mais de um processador trabalhando em conjunto no mesmo computador, para processar colaborativamente tarefas, de forma coerente e transparente, ou seja, como se apenas um único e centralizado computador estivesse executando-a.

No início, o modelo de computação distribuída objetivava a distribuição do processamento de uma aplicação por diversas máquinas físicas, resolvendo alguns problemas da computação centralizada, como exemplo, o sistema ficar inacessível devido uma máquina estar ocupada (WERNER, 2011). Não havia também a preocupação em redução de custos, redução de máquinas e equipamentos. Entretanto, atualmente existem outros fatores importantes a margem desse modelo como o custo energético e o desempenho de ambientes que o implementam.

Com o aumento da capacidade de processamento dos sistemas computacionais, até o inicio dos anos 2000, a principal maneira encontrada pelos fabricantes de microprocessadores para aumentar o desempenho de seus produtos era elevar a suas frequência de operação (TANENBAUM, 2006). Entretanto, existem limitações físicas que impedem o aumento de desempenho baseado na elevação da frequência de operação. Essas limitações são o limite máximo da velocidade da luz e a impossibilidade de condensar mais transistores. A quantidade de transistores no mesmo espaço físico é limitada pelas dimensões dos materiais que o constituem, não podendo ser reduzidos além do limite nanoscópico. Além

disso, quanto mais elementos são colocados em um mesmo espaço, mais calor é liberado pelas correntes elétricas que circulam em seus componentes, o que implica em dificuldades para a dissipação do calor (ALMEIDA, 2008).

A HPC está relacionada à área da Computação Distribuída por meio da prática de agregar o poder da computação de forma a proporcionar um desempenho muito maior do que se poderia conseguir com computadores típicos, como estações de trabalho, à fim de resolver complexos problemas da ciência, engenharia, ou de negócios (ROUSE, 2007a).

Esse termo abrange não só computadores, mas também a tecnologia de rede, os algoritmos e os ambientes necessários para o uso destes sistemas, que podem variar de *clusters* de computadores à supercomputadores. Estas plataformas provedoras de HPC podem possuir sistemas com memória compartilhada, distribuída, combinação de vários processadores e a utilização da computação paralela, visto que por meio da computação paralela o poder de processamento de uma única CPU pode ser multiplicado para resolver uma tarefa. Isso quer dizer que a computação paralela permite que problemas muito complexos, que exigem muitos cálculos, ou pesquisas em gigantes bases de dados, possam ser resolvidos em um tempo razoável. Isto se aplica, por exemplo, a casos como a previsão global do tempo e o sequenciamento do genoma humano (FISCHBORN, 2006).

A utilização de *clusters* de computadores para a HPC, é cada vez mais beneficiada pela evolução das tecnologias de processadores e pelas suas interconexões, ao mesmo tempo em que a utilização de chips multi-cores como *nodes* desses *clusters* criam um ambiente paralelo multi-nível. Com tantos núcleos de processamento operando de maneira concorrente e considerando os esforços em direção a um desenvolvimento tecnológico sustentável, faz-se essencial o desenvolvimento de sistemas paralelos que, além de prover alto desempenho, consigam fazê-lo com um eficaz consumo de energia elétrica, contribuindo assim para um melhor aproveitamento dos recursos naturais do planeta (MOR, 2010).

2.2 VIRTUALIZAÇÃO

Na comunidade acadêmica é notado um consenso sobre a virtualização ser a tecnologia chave da Computação em Nuvem em, por exemplo, (VAQUERO, 2008), (ARMBRUST, 2009), (BUYYA, 2008) e (ENDO, 2010). A virtualização é uma camada de abstração que permite que uma ou mais instancias de VMs sejam hospedadas simultaneamente em um ou mais hospedeiros físicos (SIMONS, 2010) (BESERRA, 2014).

Para o emprego da virtualização em um ambiente de HPC, alguns pontos fundamentais

precisam ser observados:

- Ausência de sobrecarga: a virtualização não deve provocar impactos significativos no desempenho do sistema, sendo este um "ponto critico" para a HPC (YE, 2010);
- Maximização dos recursos: a virtualização deve maximizar o uso dos recursos físicos disponíveis [Nussbaum et al., 2009], impedindo que recursos sejam subutilizados (JOHNSON, 2011) (BESERRA, 2012);
- Gerenciamento do ambiente: a virtualização deve melhorar a administração do ambiente, permitindo a criação e destruição rápida de VMs e distribuição flexível de recursos de *hardware*;
- Aumento de confiabilidade: a virtualização deve aumentar a confiabilidade do ambiente, provendo mecanismos de redundância e recuperação de falhas (YE, 2010) (SIMONS, 2010);
- Segurança: a virtualização deve aumentar, ou ao menos não diminuir, a segurança do ambiente mediante isolamento de aplicações em VMs (YE, 2010);

Esses requisitos podem ser considerados como um bom ponto de partida para a análise do uso das ferramentas de virtualização em ambientes de HPC.

O uso de virtualização em ambientes de HPC, apesar de promissor, possui limitações. Para atender melhor alguns dos pontos relacionados, algumas ferramentas de virtualização acabam por reduzir o suporte à outros. Além disso, mais desafios surgem na maturação dessa tecnologia, como a minimização dos efeitos da concorrência por recursos de um mesmo servidor físico no uso de seus recursos por diversas VMs (BESERRA, 2014), uma melhor adaptação a arquiteturas computacionais *multicore* e *many-core* (CHENG, 2013) (WALTERS, 2014), a necessidade de uma melhor gestão de recursos energéticos, focando na economia e o envelhecimento de *software* virtualizador.

Existem três abordagens principais para prover a virtualização dos recursos físicos dos servidores. São elas: virtualização total; virtualização assistida por *hardware*; e para-virtualização. A virtualização total permite a execução de SOs convidados em um SO hospedeiro sem necessidade de modificar o *kernel* de nenhum dos sistemas. Entretanto impõe grande sobrecarga no desempenho das VMs indo em oposição ao requisito mais importante que a virtualização deve atender para ser empregada em HPC, a ausência

de sobrecarga para não provocar impactos significativos no sistema. Embora não satisfaça este requisito, a virtualização total é bastante segura, pois proporciona completo isolamento entre o ambiente hospedeiro e os ambientes virtuais (YOUNGE, 2011).

A penalização no desempenho provocada pela virtualização total pode ser mitigada com o uso de virtualização assistida por *hardware*, que possui um conjunto de instruções especificas para virtualização presente em quase todos os processadores atuais e em alguns elementos de IO (YE, 2010). Essa abordagem frequentemente é utilizada conjuntamente com a virtualização total, mantendo suas vantagens e atenuando seus inconvenientes. Já na paravirtualização os dispositivos de *hardware* não são emulados e sim acessados mediante o emprego de *drivers* paravirtualizados especiais, obtendo um melhor desempenho quando comparado com o obtido pela virtualização total, mesmo quando assistida por *hardware* (NUSSBAUM, 2009).

A principal diferença em relação a virtualização total é que os *kernels* dos sistemas operacionais convidados devem ser modificados para providenciar novas chamadas de sistemas. Ela aumenta o desempenho, pois reduz o acesso e o consumo da CPU, entretanto reduz a segurança, a confiabilidade e dificulta a administração do ambiente por sempre ter que modificar os sistemas convidados.

A virtualização pode flexibilizar a distribuição dos recursos de um *datacenter*, onde cada VM tem como requisitos determinadas quantidades desses recursos, em geral, de forma isolada. Cada servidor pode alocar várias VMs, sendo possível executá-las paralelamente e de maneira isolada. Cada VM é composta por um SO que pode ser diferente do sistema do servidor que a executa e, no caso típico, contém uma ou mais aplicações (tarefas, *tasks*) para serem executadas. Portanto, todas as VMs ficam executando independentes entre si, o que permite o isolamento do trabalho executado e do seu desempenho. O isolamento das VMs também facilita o tratamento de falhas, como falhas de *software* que neste caso não afeta o restante dos recursos e VMs que executam no mesmo servidor. Para alguns casos mais complexos como falhas de *hardware*, uma possibilidade é a migração de VMs (BARBOSA, 2014).

A migração de VMs é uma das grandes vantagens da virtualização e com ela, é possível consolidar recursos por entre os servidores do *datacenter*. A virtualização pode facilita a gestão dos servidores, por meio desse procedimento. As migrações podem ser utilizadas para consolidar as VMs num menor número de servidores, reduzindo a quantidade de recursos subutilizados. Esta consolidação permite controlar a distribuição das cargas de

trabalhos criando possibilidades, como a utilização de técnicas de gestão de energia, com o objetivo de reduzir custos operacionais (BARBOSA, 2014).

Existem dois modelos de migração amplamente referenciados na literatura são o de *stop-and-copy* e o de *live migration*. Basicamente com *stop-and-copy* a VM tem a sua execução suspensa completamente no servidor. A partir daí, uma cópia dessa VM é efetuada e enviada para o servidor de destino e, após a conclusão da cópia, a execução da VM continua nesse servidor. Já a *live migration*, durante uma fase inicial da migração, efetua cópias da memória paginada (pré-cópia) para um servidor de destino. Ela pode ter que retransmitir as páginas de memória modificadas pela VM durante este processo. Em seguida essa VM é suspensa e a execução das páginas de memórias que estavam em uso no momento desta parada são copiadas. No fim deste processo é reiniciada a execução no servidor de destino (CLARK, 2005). A vantagem dessa estratégia está no fato de a VM e todas as aplicações nela executadas continuarem a executar enquanto a imagem da memória é transferida, proporcionando um menor *downtime* dos serviços. Existe também a *live migration* com a estratégia baseada na pós-cópia com o propósito de reduzir o tempo de realocação das VMs durante o processo de migração. Essa abordagem é caracterizada pela execução da VM no hospedeiro de destino antes da transferência de suas páginas de memória. Apesar da pós-cópia reduzir o tempo de realocação da VM, provoca uma degradação no desempenho das aplicações, visto que ela é interrompida toda vez que uma página de memória não está disponível e somente é reinicializada quando a página de memória é recuperada a partir do hospedeiro de origem. Para atenuar o *downtime* sofrido pela VM depois de sua realocação no hospedeiro de destino, a técnica de pré-paginação trabalha junto com mecanismo de paginação sob demanda. Esta técnica é utilizada para esconder a latência das falhas de página ocorridas quando a VM acessa uma página de memória que ainda não está disponível (HINES, 2009).

2.3 COMPUTAÇÃO EM NUVEM E NUVENS VERDES

Com a evolução das tecnologias de processadores incorporando tecnologias de virtualização e pelo aumento na capacidade de comunicação de dados, o modelo da Computação em Nuvem surgiu fornecendo aos seus usuários soluções que flexibilizam e facilitam o uso de recursos outrora disponíveis somente em *datacenters* de grandes organizações. O termo "Computação em Nuvem" está em ascensão nas pesquisas feitas no Google de acordo com o Google Trends, desde meados de 2007, conforme apresentado na Figura 2.1.

FIG. 2.1: Referencias ao termo “Computação em Nuvem” no Google

Esse conceito, até pouco tempo abstrato, dispõe, hoje em dia, de uma grande quantidade de ferramentas e serviços para diversos tipos de necessidades, seja em função do negócio (aplicativos e softwares) ou em função de capacidade computacional (processamento e *storage*). As nuvens computacionais contribuem como fonte de recursos adicionais, principalmente quanto à sua velocidade de resposta ao permitir agregar rapidamente mais recursos em função da demanda necessária (elasticidade), permitir que seus usuários sejam capazes de montar e alterar a característica da infraestrutura virtualizada em função do tipo da necessidade (flexibilidade), permitir acessar a aplicações remotamente dissociando o espaço geográfico onde os recursos e pessoas que os utilizam se encontram (portabilidade) e permitir o compartilhamento dos recursos, contribuindo assim para otimização do uso dos mesmos.

A Computação em Nuvem é um modelo que convenientemente possibilita, por meio de uma rede, acesso sob demanda à um *pool* de recursos computacionais configuráveis (por exemplo, redes, servidores, armazenamento, aplicações e serviços) que podem ser rapidamente provisionados e liberados com um esforço mínimo de gestão ou interação com o provedor de serviços. Este modelo promove disponibilidade e é composto de cinco características essenciais, três modelos de serviço, e quatro modelos de implementação. As características referenciadas são: auto-atendimento sob demanda; amplo acesso a rede; *pool* de recursos; elasticidade rápida; e serviços mensuráveis. Os modelos de serviço disponíveis são classificados em: *Software-as-a-Service* (SaaS); *Platform-as-a-Service* (PaaS); e *Infrastructure-as-a-Service* (IaaS). Os modelos de implantação são classificados em: Público; Privado; Comunitário; e Híbrido (MELL, 2011).

Além das definições básicas da Computação em Nuvem, também existe uma sugestão

de arquitetura baseada em um modelo ator-papel que estabelece elementos centrais desse modelo para programas de gerenciamento de TI e suas interações (LIU, 2012). Na Figura 2.2 é apresentado um esquema que representa os esses elementos.

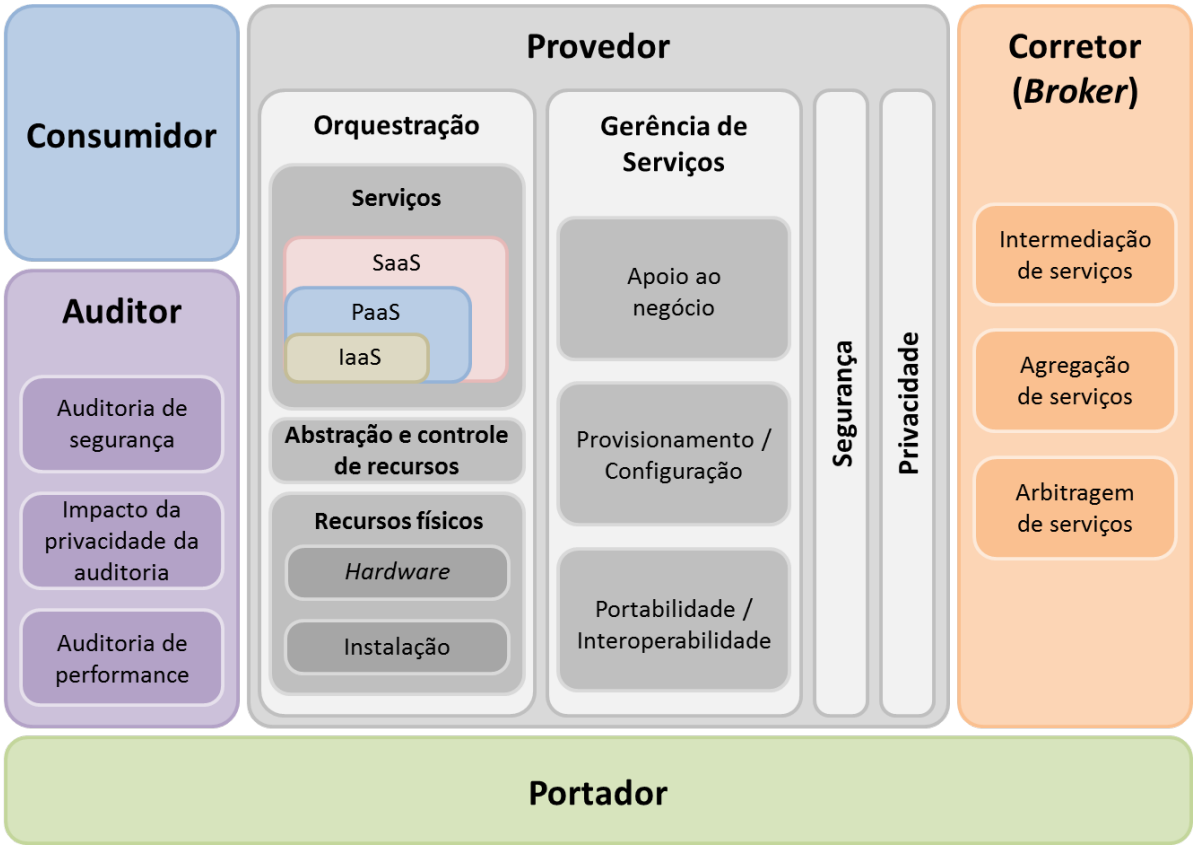


FIG. 2.2: Modelo de Referência Conceitual do NIST – Adaptado de (LIU, 2012)

Como as Nuvens Computacionais são uma realidade, atualmente, e suas possibilidades refletem em vantagens financeiras para muitas organizações, a crescente aderência a esta tecnologia pode ser observada. Nesta direção, aplicativos e serviços são fornecidos por meio das redes de computadores, facilitando a vida dos consumidores na hora de planejar e utilizar as tecnologias e ambientes para trabalho necessário, bem como seus bolsos, pois o modelo financeiro (*pay-as-you-go*), que as nuvens públicas implementam, fazem os mesmos pagarem somente pelo que utilizam, como serviços básicos de água ou luz (AZEVEDO, 2012). Este é um ponto de destaque deste modelo computacional, pois em vista disso é possível não só obter economia financeira, mas também otimização de recursos.

Ao planejar os requisitos de infraestrutura, as organizações são obrigadas a utilizar critérios de pico de carga para dimensionar seus investimentos em recursos. Isto significa

que os recursos devem ser escolhidos de acordo com a demanda que se prevê atender. Na Figura 2.3 está sendo representado um cenário que ilustra uma demanda por recursos em um *datacenter* ao longo do tempo. Para o ambiente atender as necessidades, um investimento em bens de capital precisa ser feito. Esse investimento chama-se *Capital Expenditure* (CAPEX) e está diretamente relacionado aos recursos que o ambiente oferece, pois esse é o foco em um ambiente de nuvem.

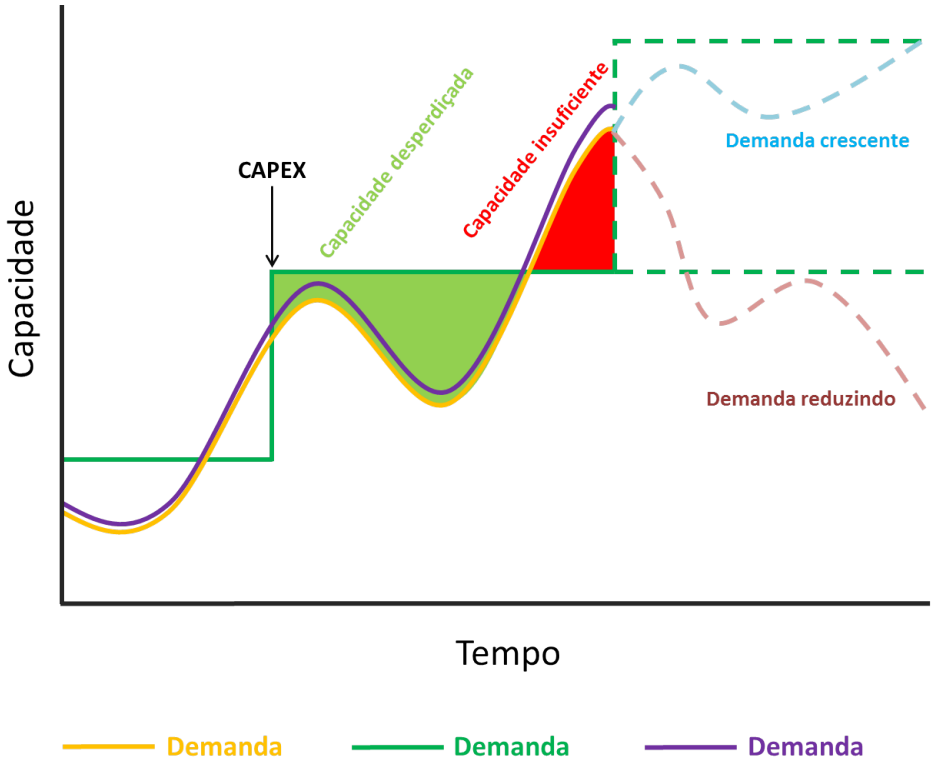


FIG. 2.3: Curvas de Capacidade x Utilização – Adaptado de (KEPES, 2010)

A linha verde é a infraestrutura física local de capital intensivo e mostra sua relação com a demanda (linha amarela). Esta infraestrutura física pode deixar a organização com excesso de capacidade,, ou seja, com recursos sendo subutilizados ou sobrecarregados, levando à degradação do serviço. A linha roxa, que segue de perto a demanda real mostra um dos benefícios da propriedade de elasticidade de uma infraestrutura de nuvem. Esta propriedade significa que os serviços podem ser executados sobre os recursos que podem ser escalados em função da demanda, para cima ou para baixo, rapidamente.

Devido a grande quantidade de carga energética para operar um *datacenters* em nuvem, soluções que trazem economia de energia à esses ambientes são cada vez mais necessária devido a crescente demanda por seus recursos. Essas soluções trazem aspectos

da TI verde para as Nuvens Computacionais e isso faz com que esse modelo passe a se chamado Nuvem Verde.

TI Verde é o termo utilizado para referenciar as atitudes sustentáveis ligadas a área de TI. Como a principal abordagem quanto a esse termo é o consumo energético, as medidas adotadas estão voltadas para o consumo de energia nos *datacenters*. Porém como a questão sustentável envolve o meio ambiente de forma geral, tudo o que envolve a TI precisa ser inserido no contexto, ou seja, desde a consciência dos usuários e organizações até a questão do processo de produção, consumo e descarte de produtos e serviços. Outras questões como o consumo de energia da infraestrutura de comunicação e o impacto da emissão de dióxido de carbono (CO2) precisam ter maior atenção e devem ser alvos das ações sustentáveis a serem adotadas. A primeira questão justifica-se com o fato de cada vez mais serviços e recursos estarem disponíveis na Internet, sendo acessados, inclusive, por meio de redes móveis. Portais de notícia; televisão, rádio e mídia em geral; aplicativos e softwares; arquivos e computadores virtuais; são alguns exemplos que contribuem para uma maior utilização das infraestruturas de comunicação e isso faz aumentar o consumo de energia nesse meio (WESTPHALL, 2013).

Se por um lado a demanda por serviços de nuvens computacionais é crescente, estabelecendo um pressuposto de que cada vez mais a TI representada por estas grandes organizações irá consumir mais energia e emitir mais carbono à atmosfera, por outro, a abordagem deste modelo computacional pode representar uma interessante alternativa para a utilização eficiente dos recursos computacionais, uma vez que permite consolidar a carga de trabalho de uma grande quantidade de usuários, assim como pelo fato de seus recursos favorecerem a consolidação da carga de trabalho do próprio *datacenter*, admitindo, por exemplo, redução de utilização de espaço, menos emissão de calor, e consequentemente otimização da infraestrutura de refrigeração, e uma solução rápida para os sistemas UPS problemáticos, ou seja, quanto menos servidores, menor a demanda dos UPS. Todavia, para isso, é preciso colocar o foco do gerenciamento dos recursos na questão energética, sem onerar o desempenho e os níveis de entrega de serviço acordados com os consumidores. Estes são os desafios das Nuvens Verdes (WESTPHALL, 2013).

Atualmente o grande foco das Nuvens Verdes está baseado fundamentalmente na aplicação de critérios de provisionamento, alocação, redimensionamento e migração de VMs para obtenção de uma consolidação de carga nos servidores físicos de forma eficiente. Porém, outros aspectos precisam ser levados em conta quando se analisa o modelo ener-

gético das nuvens. Um exemplo é a infraestrutura de rede. Como a Computação em Nuvem é um modelo fortemente dependente das tecnologias de comunicação e redes. A infraestrutura de rede e de telecomunicações contribui com uma grande parte do consumo de energia e das emissões de gases da TI além de possuir características diferenciadas, motivo pelo qual os esforços para torná-la mais eficiente e ambientalmente correta constituem uma área especial de estudo que tem sido identificada sob o nome de Redes Verdes (*Green Networking*) (BIANZINO, 2012).

Diante da forte dependência das redes computacionais pelas nuvens, os princípios das Redes Verdes são fundamentais para o conceito das Nuvens Verdes. Um dos aspectos relacionados a isso é o tamanho dos arquivos manipulados no ambiente de nuvem. Se forem muito grandes, a rede pode se tornar um dos principais contribuintes para o consumo de energia do modelo de nuvem, assim, seria mais verde executar o serviço localmente do que nessa suposta nuvem (GARG, 2011).

Visto a quantidade de aspectos que precisam ser tratados num ambiente de nuvem computacional e mais, quando se precisa inserir a este ambiente os princípios da TI Verde, isto deixa claro que os desafios das Nuvens Verdes são problemas complexos devido a quantidade de componentes e esferas envolvidas. Para que todos sejam tratados de maneira eficaz, torna-se prioritário estabelecer métricas e conjuntos de *benchmark* que permitam relacionar as tarefas realizadas com a pegada ambiental, seja em níveis de consumo elétrico ou mesmo a previsão de emissão de gases poluentes. Para isso, todos os componentes passíveis de serem monitorados precisam incorporar funcionalidades para informar a energia consumida de acordo com suas atuações (WESTPHALL, 2013).

2.4 ALOCAÇÃO E ESCALONAMENTO DE RECURSOS EM NUVENS

A alocação de recursos é referenciada em várias áreas da computação e na Computação em Nuvem é um mecanismo que tem como objetivo garantir que os requisitos de aplicações sejam atendidos corretamente pelo fornecedor da infraestrutura. A alocação de recursos deve também considerar o estado corrente de cada recurso, permitindo o uso de algoritmos para melhor alocar os mesmos, físicos e/ou virtuais, às necessidades das aplicações (BARBOSA, 2014).

Uma das partes mais desafiadores da computação distribuída é o problema de escalonamento desses recursos. O objetivo do escalonamento é determinar uma atribuição de tarefas a elementos de processamento com o intuito de otimizar certos índices de

desempenho, como tempo de processamento, leitura e escrita em disco ou memória (EL-REWINI, 2005). Um escalonador é definido como um mecanismo de controle e gestão utilizado para gerenciar de maneira eficiente o acesso aos recursos por seus vários consumidores (CASAVANT, 1988). Na Figura 2.4, estão sendo ilustrados os componentes principais desse mecanismo.

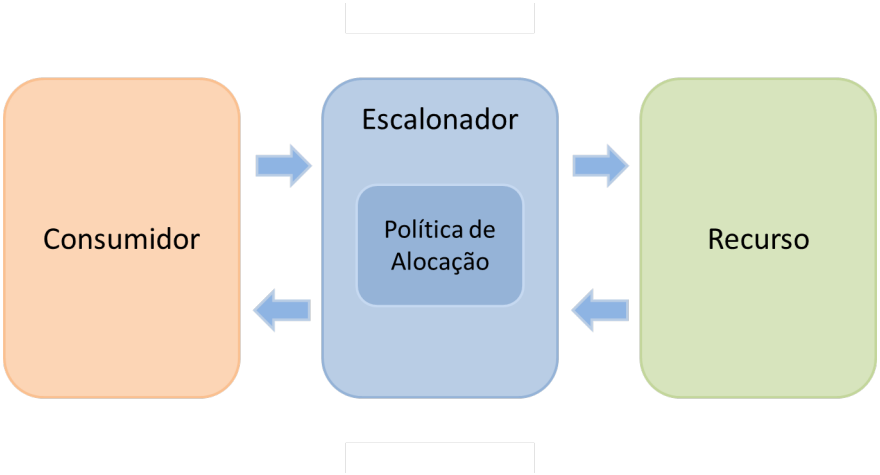


FIG. 2.4: Componentes do sistema de escalonamento – Adaptado de (BACHIEGA, 2014)

Relacionado a Computação em Nuvem, podem-se destacar dois tipos de escalonadores que atuam na execução do ambiente, o escalonador do gestor e o escalonador do virtualizador. O primeiro trabalha com a finalidade de escalonar instâncias de VMs para os servidores o segundo atua dentro do servidor, no sistema operacional, compartilhando e dividindo o tempo de processamento das tarefas (BACHIEGA, 2014).

Duas propriedades que devem ser avaliadas em qualquer sistema de escalonamento de recursos:

- a gerência de recursos deve satisfazer os consumidores;
- consumidores não devem ser prejudicados por problemas associados.

Já sob o paradigma das nuvens, uma terceira propriedade precisa ser adicionada para satisfazer também os provedores que oferecem os serviços aos consumidores. Sendo assim:

- a gerência de recursos deve satisfazer os provedores de serviços.

As nuvens possuem a necessidade de redução dos custos operacionais, o aumento da sua eficiência, assim como o melhoramento da sua performance. Os sistemas atuais são

cada vez mais complexos, o que traz à introdução de meios que elevam essa complexidade como por exemplo, SLAs, para avaliar a satisfação dos objetivos, garantir a qualidade do serviço (QoS) e também o desenvolvimento de técnicas e modelos tolerantes a falhas (BARBOSA, 2014). Para que essas propriedades (dos consumidores e provedores de serviços) não se sobreponham, devem existir SLAs entre elas. Esses SLAs garantem que os recursos estarão disponíveis para os consumidores ou parte deles, no qual os consumidores aceitam participar de um ambiente compartilhado ou não, dependendo do tipo de acordo estabelecido.

Existem diversas propriedades que podem fomentar um algoritmo de escalonamento. Nessas condições torna-se essencial analisar e investigar quais os objetivos que este deve possuir para atender um serviço ou tarefa. Atualmente, no geral, os algoritmos de escalonamento de nuvens *opensource* são determinísticos, ou seja, as informações para determinar o nó computacional com mais recursos não levam em consideração as condições físicas do nó computacional no momento do escalonamento(BACHIEGA, 2014).

O escalonamento de processos tem um impacto substancial no desempenho quando as estratégias de escalonamento não-adaptativo são utilizadas. Assim, se faz necessário criar um escalonador dinâmico e adaptativo (DANDAMUDI, 1994).

3 REVISÃO DA LITERATURA

Atualmente, a Computação em Nuvem é uma área da Ciência da Computação que está em ascensão em relação às pesquisas que envolvam o tema "Sustentabilidade". Essa área é considerada prioritária quando se pensa na implementação de sistemas computacionais, segundo (WESTPHALL, 2013). Nesse artigo, são abordados os conceitos e o estado da arte das tecnologias de Computação em Nuvem e da TI Verde que em conjunto definem e viabilizam o que é chamado de *Green Cloud Computing*.

Os autores, a partir de uma pesquisa realizada sobre trabalhos relacionados às duas áreas, constatam que existem necessidades de maiores esforços para aplicar os conceitos de TI Verde ao ambiente de nuvens computacionais. Os princípios da *Green Cloud Computing* maximizam as capacidades da aplicação da estratégia de TI Verde sobre os *datacenters* dos provedores de nuvens. Um exemplo disso são algoritmos que focam na economia de energia mantendo os níveis de desempenho acordados. O objetivo deste modelo é manter a qualidade dos serviços requisitados pelos usuários, minimizando o uso dos recursos. Porém, os autores alertam que para isto ocorrer, ainda é preciso um esforço para melhorar a integração de todos os elementos do sistema.

Embora o objetivo da TI Verde seja atribuir aos ambientes computacionais suas medidas sustentáveis para diminuir as emissões de gás carbônico e o impacto ambiental, verifica-se com os trabalhos relacionados neste artigo que a grande maioria dos esforços ainda se concentra apenas nos aspectos referentes à eficiência energética e à redução dos custos operacionais.

Em (GARG, 2011) os autores dão destaque a atenção que a questão energética precisa, pois afirmam que *datacenters* são grandes vilões quando se pensa em sustentabilidade na área da Tecnologia da Informação (TI). É necessária grande carga e energia para mantê-los sempre operando de forma eficiente.

Manter um *datacenter* funcionando requer um alto consumo de energia. Atualmente, um típico *datacenter* com 1000 racks necessita de 10 Megawatts de energia para operar. Os autores afirmam ainda que o *datacenter* mais eficiente construído com as melhores taxas de utilização só mitigaria, em vez de eliminar, as emissões nocivas de carbono. Com isso, os autores propõem um *framework* intermediado por um *cloud broker* "verde" que

é capaz de calcular a previsão de consumo energético e emissão de CO2 que o serviço solicitado irá consumir.

Neste trabalho, os autores ainda destacam que aspectos à margem dos *datacenters*, em alguns casos, podem contribuir com uma ineficiência energética, precisando assim ser tratados. São eles: arquivos muito grandes que geraram mais tráfego de banda nas redes, aumentando o consumo de energia nesses dispositivos; replicação de arquivos em vários servidores, que melhora a segurança do sistema, porém fazem com que mais recursos necessitem ser consumidos. Cada componente, sejam dispositivos computacionais ou de rede que são utilizados para acessar os recursos de um *datacenter* remotamente, contribuem diretamente com o consumo de energia que envolve esses ambientes. Além disso, outros componentes como refrigeração e dispositivos elétricos também tem grande parcela de consumo nos *datacenters*, embora não estejam envolvidos diretamente com o provisionamento dos serviços de nuvem.

Em (BUYYA, 2010) é visto que a gerência de energia é um importante atributo quando se fala em Computação em Nuvem. Reduzir o uso de energia é um desafio complexo por conta das possibilidades que uma nuvem computacional oferece. A solicitação de aplicações, dados e recursos é intensa e de grande volume fazendo com que os servidores que atendem essas requisições tenham a necessidade de processá-las rapidamente e em um curto período de tempo. A economia de energia pode ser alcançada por meio de várias estratégias como a consolidação de máquinas virtuais (VMs) de acordo com a utilização dos recursos, a equalização ou mesmo hibernação das CPUs quando os servidores estiverem desocupados, a realocação das cargas de trabalho ou VMs, dentre outras.

Em (POVOA, 2013) os autores abordam a importância da medição e compreensão de impacto que recursos computacionais exercem sobre o consumo energético. Isso, segundo os mesmos, é uma importante informação para que processos e infraestruturas sejam aprimorados, buscando redução de gastos e beneficiando o meio ambiente. Porém, para se obter essas informações, uma forma é a utilização de sensores e aparelhos de medição de consumo energético, porém esses possuem alto custo. Diante disso, o trabalho apresenta uma abordagem de baixo custo para a obtenção dessas informações, por meio de uma equação de regressão que prediz a quantidade de energia consumida em determinado instante. Os autores ainda constatam que o trabalho é uma proposta de ferramenta eficaz para a medição de consumo energético não sendo necessários, então, os sensores e aparelhos para a execução dessas tarefas.

Em (BELOGLAZOV, 2012), é proposto um sistema de gestão eficiente dos recursos energéticos para *datacenters* em nuvens que reduz custo operacional e proporciona qualidade de serviço (QoS). A economia de energia é conseguida por meio de consolidação de VMs de acordo com o status dos recursos, topologias de redes virtuais e temperaturas dos *hosts* de computação. São apresentados resultados de heurísticas de avaliações feitas por simulação, onde realocações dinâmicas acontecem por meio de migrações de VMs baseadas no estado das CPUs. Os resultados apontam redução significativa no consumo de energia e mantimento da QoS.

Em (BELOGLAZOV, 2010) foram avaliadas heurísticas para realocação dinâmica de VMs com o objetivo de minimizar o consumo de energia fornecendo QoS. Essas avaliações consistiram da proposta de três políticas de seleção de VMs para migração: a *Minimization of Migrations* (MM) que minimiza a quantidade de migrações no ambiente para evitar *overhead* de migração; *Highest Potential Growth* (HPG) que migra VMs com possuem as mais baixas taxas de utilização de CPU relativamente ao que foi requisitado, minimizando um aumento total potencial de utilização e de violação de SLA; e *Random Choice* (RC) que migra o número necessário de VMs de acordo com uma variável aleatória uniformemente distribuída.

A política implementada e validada foi a MM que trabalha com dois *thresholds*, mínimo e máximo, para otimizar o consumo de energia e diminuir as violações de SLA. Se a utilização da CPU de um *host* reduzir a baixo do *threshold* mínimo, todas as VMs que nele residirem serão migradas desse *host* e o *host* será desligado com o objetivo de eliminar o consumo de energia desse servidor que encontra-se perto de *idle*.

Se a utilização for acima do *threshold* maior algumas VMs são migradas do *host* para outro, reduzindo o uso do *host* origem visando prever potenciais violações de SLA. Os resultados mostraram que a técnica de realocação dinâmica de VMs, bem como o desligamento dos servidores ociosos pode trazer economia de energia a *datacenters*.

Foi utilizado o CloudSim para validação dos resultados pois o mesmo já possui outras políticas já implementadas, facilitando assim a validação. Os resultados dessa técnica trouxeram economia de até 83% quando comparada a uma política sem consciência energética (NPA), 66% quando comparada a técnica de DVFS e 23% quando comparada a técnica com *threshold* simples (ST).

Em (SINHA, 2011), os autores propuseram uma politica de alocação de VMs dividindo o trabalho de escalonamento das VMs em duas partes. A primeira é a seleção de VMs

a serem migradas, onde essa seleção acontece por meio da utilização de um *threshold* dinâmico e a segunda e a utilização da ordenação *Best Fit Decreasing* para a alocação das VMs. O uso do *threshold* dinâmico, baseado na da utilização da CPU, tenta reduzir o consumo de energia do *datacenter* em nuvem, tratando as cargas de trabalho dinâmicas e imprevisíveis . Além disso, também foi considerada a questão da QoS para os usuários, por meio da minimização das violações de SLA.

Os resultados mostraram as diferenças de custo com e sem o uso de política de migração, bem como as diferenças quanto ao uso dos *thresholds* estabelecidos. A consolidação dinâmica da VM e o desligamento dos servidores ociosos maximiza a eficiência energética do ambiente. O trabalho foi dividido em algumas etapas que inicialmente foi a análise conceitual da *live migration* e sua implementação no CloudSim. Posteriormente estudaram as implementações já existentes no próprio *framework*, ou seja DVFS e NPA. Estas duas não implementam migrações. Em seguida, utilizou-se a implementação de *threshold* único sobre o simulador. Assim, uma atribuição estática de valor de *threshold* limite foi feito sem nenhuma definição para um *threshold* de limite inferior. Em seguida, foi implementada a migração com dois valores estáticos de *threshold*, um inferior e outro superior. Finalmente, o trabalho avançou na direção da implementação do próprio conceito de *threshold* dinâmico, utilizando as teorias de *threshold* expostas no mesmo.

Foram comparados com as estratégias DVFS e NPA. Concluiu-se que, ao usar políticas de eficiência energética para migrações, o consumo de energia pode ser minimizado, resultando em diminuição do consumo elétrico do *datacenter*. Em NPA é usado o montante máximo de poder entre todas as teorias levadas em consideração, resultando em mais custos. Em DVFS pode-se usar menos energia, mas para o cenário real isso pode mudar devido a toda dependência ser limitada a tensão e frequência dos recursos. Em sequência, quanto ao *threshold* único, com ele foi violado um número máximo de SLA com o consumo nominal de energia. Em seguida, no *threshold* duplo as violações de SLA caíram em cerca de 25% do *threshold* único. Finalmente, quanto ao *threshold* dinâmico também foram reduzidas um pouco mais as violações de SLA em comparação com o *threshold* duplo, bem como o consumo energético.

A ideia de usar limites máximos e mínimos (*thresholds*) em algoritmos de escalonamento também pode ser encontrada em (METKAR, 2013), onde os autores reforçam a importância da tecnologia de virtualização e uma de suas principais características, a migração (*live migration*). Segundo os autores, essa técnica é amplamente utilizada para

tolerar falhas, balancear cargas de trabalho e consolidar recursos. Com isso, ela também é fator primordial na minimização de consumo de energia dos *datacenters*.

Em (SINGH, 2013) afirma-se que o crescimento da demanda por poder computacional levou à criação de grandes *datacenters*. Esses ambientes consomem enormes quantidades de energia elétrica, resultando em alto custo operacional e emissão de dióxido de carbono. Além disso, esses ambientes precisam estar aptos a fornecer qualidade na entrega dos serviços (QoS) aos seus clientes, resultando assim na necessidade de lidar com desempenho e economia de energia. Os autores também propõem uma politica de gestão de recursos para *datacenters* que implementem nuvens computacionais. Essa politica foi chamada de *Minimum CPU Utilization* (MCU) e engloba um algoritmo de escalonamento de tarefas que considera que se tarefas forem alocadas em VMs mais cedo, essa carga é melhor gerida e isso reduz as migrações, reduzindo também violações de SLA. Sua validação e comparação foi também executada no CloudSim e teve como métodos para comparação as políticas NPA, DVFS e MC, descrita no trabalho citado anteriormente.

Em (QUANG-HUNG, 2013), os autores afirmam que a popularidade das nuvens trouxe usuários de HPC para esses ambientes virtualizados. Isso torna a relação de minimização de consumo energético e qualidade de serviço um ponto chave, atualmente, no contexto do paradigma das nuvens. De um lado, provedores tentam minimizar o consumo por meio da utilização de um menor número de servidores. Do outro, usuários querem cada vez mais recursos para executar aplicações de alto desempenho. Neste trabalho relacionado, os autores propõem uma ideia de escalonador que visa tratar essa problemática escolhendo um melhor servidor, energeticamente eficiente, baseado em uma métrica de desempenho por Watt. Segundo os autores, o método proposto pode reduzir até 35% o consumo total de energia se comparado com o estado da arte das heurísticas de alocação. Entretanto, neste trabalho, migrações de VMs e requisições de recursos maiores do que as capacidades dos *hosts* não foram consideradas.

Em (QUANG-HUNG, 2014) os autores afirmam que a redução do consumo de energia para sistemas em nuvem é de alta prioridade para qualquer provedor de nuvem. Neste trabalho foi estudado a alocação consciente da energia de máquinas virtuais (VMs) em sistemas de nuvem HPC. Apresentou-se um exemplo que mostra que o uso de heurísticas como a *Best-Fit Decreasing*, para reduzir o número de servidores que atendem as demandas, não consegue levar a um mínimo de consumo de energia no ambiente. Por outro lado, observou-se que minimizar o consumo total de energia é equivalente a minimizar a

soma do tempo total de conclusão de todas as tarefas nos servidores. Com base nesta constatação, foram propostos os algoritmos MinDFT-ST e MinDFT-FT para alocar as VMs em servidores de tal maneira que isso minimizasse os tempos totais de conclusão de todas as tarefas nos servidores. Os resultados das simulações mostraram que os algoritmos podem reduzir o consumo total de energia em 22,4% e 16,0%, respectivamente, em comparação com o estado da arte das heurísticas que consideram economia de consumo energético.

Segundo (BARBOSA, 2014), o rápido crescimento do paradigma da Computação em Nuvem implica no fato de que muitas tecnologias relacionadas ao mesmo ainda não sejam maduras o suficiente, abrindo portas para uma investigação maior desse universo tecnológico. Os autores deste trabalho afirmam que um dos grandes problemas desse modelo é o seu consumo energético, onde existe a necessidade de reduzi-lo não só para recuperar o investimento realizado como também por causas ambientais.

Este trabalho aborda a consolidação de máquinas físicas do *datacenter* e tem como objetivo otimizar o uso desses recursos físico de forma a minimizar a sua subutilização, mas sem causar prejuízo significativo na performance do ambiente. Algumas estratégias são exploradas e permitam essa consolidação com o objetivo de compreender melhor suas influencias no desempenho das nuvens computacionais e determinar quais as características que mais afetam os resultados. Essas estratégias são divididas em três categorias e constituem-se de algoritmos e técnicas como os algoritmos de alocação de VMs, os algoritmos de seleção de VMs e as técnicas de gestão de energia.

Os autores não utilizaram um simulador de nuvens computacionais disponível no mundo acadêmico, pois alterações e extensões seriam necessárias de acordo com o levantamento executado e o tempo para obter o conhecimento total da ferramenta. Por isso, promover as alterações necessárias em qualquer um deles passou a ser uma tarefa inviável, portanto um simulador que atendesse os requisitos foi desenvolvido. Nele foram simuladas várias estratégias, tendo em vista diferentes cenários, combinações de algoritmos de alocação e seleção com as técnicas de gerenciamento de energia, bem como foi analisado o conjunto de algoritmos de alocação dinâmica de VMs com diferentes cargas de trabalho, usando as técnicas de gestão de energia, além de a possibilidade de usar migrações para beneficiar o consumo energético.

Como resultado de uma análise comparativa entre os dados das várias execuções, apontando as características e diferenças entre os experimentos, foi mostrado que com

as técnicas de gestão de energia é possível reduzir o consumo energético sem piorar o desempenho, enquanto que o uso de migrações de VMs, com objetivo de melhorar o consumo energético, não obteve bons resultados nas simulações.

No presente trabalho, o foco foi atacar a problemática da relação desempenho-energia por meio de uma estratégia de alocação de VMs que consolida os recursos de acordo com as características de utilização dos componentes dos servidores da nuvem. Essa estratégia considera as diferenças das características do consumo de recurso que as VMs de dois tipos distintos (CPU-*bound* e IO-*bound*) demandam, para com isso tentar balancear desempenho e consumo energético minimizando violações de SLA. Migrações e extrapolação de recursos foram consideradas dadas as possibilidades avaliadas quanto as características computacionais das tarefas executadas pelas VMs, na tentativa de aproveitamento de recursos, visando balancear o desempenho e a economia de energia, bem como reduzir violações de SLA no ambiente de nuvem. Tratar a questão de tarefas que demandam processos de IO também é um item diferenciado a ser destacado nesse trabalho, devido aos grandes *datacenters* de hoje em dia possuírem não só serviços virtualizados em nuvem e HPC, mas também outros que envolvem *storages* e Big Data como a análise e transformação de grandes massas de dados.

4 MÉTODO DE ESCALONAMENTO PROPOSTO

Neste capítulo são apresentados os detalhes de cada passo executados até chegar ao método de escalonamento. Esses passos consistiram da definição de um ambiente de execução de experimentos desde a configuração dos recursos (máquinas reais, máquinas virtuais, método de medição de consumo), da escolha dos *benchmarks*, da analise dos dados dos experimentos e do desenvolvimento do protótipo do algoritmo com as regras estabelecidas a partir da avaliação das informações.

No escopo deste trabalho, as avaliações foram importantes para projetar a execução do método em recursos de ambientes distribuídos. Com a análise do comportamento dos *benchmarks* em VMs executando em um único equipamento pôde-se obter uma base para estimativas e previsões nos demais que formariam um *datacenter*. Sendo assim, foi possível concentrar informações importantes nas quais o método se apoia, pois sabe-se que a disposição das VMs em relação aos servidores, em um ambiente de nuvem, pode afetar o desempenho e o consumo energético do *datacenter*.

4.1 CONSIDERAÇÕES INICIAIS

O que fundamentou o desenvolvimento do método de escalonamento foi o resultado dos experimentos executados. Com os experimentos buscou-se características de consumo de recursos e impactos no que se refere ao desempenho e consumo de energia quanto a execução das aplicações dentro de VMs em um *host* físico.

Em HPC, a execução das aplicações estão geralmente ligadas à utilização de ambientes de *clusters*. Com o amadurecimento das tecnologias de virtualização, a criação, manutenção e acesso à esses ambientes passou a ser facilitada, aumentando, assim, disseminando ainda mais à HPC. Entretanto, no escopo deste trabalho não foram avaliados *clusters* virtuais executando aplicações HPC, pois, como a abordagem dessa pesquisa é inicial, uma avaliação de VMs simples, que executam aplicações com características computacionais comuns à HPC, foi suficiente para o cumprimento dos objetivos.

Contudo, é preciso destacar a importância de se conhecer o comportamento dos itens avaliados para que o método a ser criado mantenha forte relação com o ambiente de execução e suas características, nesse caso evidencia-se a questão da importância da es-

pecificação na criação de um método e o passo de avaliação do ambiente ao qual o método será aplicado.

Um exemplo pode ser descrito quanto a ocorrência dos *swaps* ou atrasos de comunicação de rede em ambientes de *clusters* virtuais. Neste caso, isso refletiria no consumo e desempenho das VMs integrantes deste *cluster*, mais ainda se as mesmas estivessem concorrendo ao mesmo recurso. Por conta dessas ocorrências, as VMs levam mais tempo para serem executadas por completo e, consequentemente, mais energia do servidor será despendida. Por isso ser uma questão relevante à execução do ambiente, um método de escalonamento que visa o balanceamento de desempenho e consumo energético precisa considerar isso. Portanto, avaliar esse tipo de característica e introduzi-la no método é essencial para a obtenção de um bom resultado.

Então, partindo desse princípio, para o desenvolvimento do algoritmo de escalonamento que considera a melhor utilização dos recursos do servidor, tanto objetivando desempenho como eficiência energética, foi necessário executar as medições e analisá-las para observar as características do ambiente. Com essas informações foi possível relacionar as possibilidades da execução de cada cenário para atender os objetivos estabelecidos nesse trabalho. Logo, considerando as características do recurso computacional disponível e os dados das medições (carga, tempo de execução, consumo energético, tamanho do ambiente virtualizado) aferidos no mesmo, pôde-se definir a estratégia de escalonamento.

4.2 DEFINIÇÕES E CARACTERÍSTICAS DOS EXPERIMENTOS

Para verificar as diferenças de consumo de recursos e de energia de cada máquina virtual nos servidores, considerando os tipos de aplicações (CPU-*bound* e IO-*bound*) que executam nas mesmas, foram estabelecidos três diferentes ambientes para cada tipo de aplicação. Esses ambientes consistem de VMs com:

- 1 processador virtual (vCPU) e 768 MB de RAM virtual;
- 2 vCPUs e 1536 MB de RAM;
- 4 vCPUs e 3072 MB de RAM; e
- Sistema operacional CentOS.

Essa verificação dos recursos consumidos, tanto em nível de ocupação dos componentes como o gasto energético, possibilitou estabelecer as condições e limites de distribuição das

VMs no servidor, considerando seus tipos. Como cada tipo de aplicação tem um comportamento computacional diferente da outra, essas informações foram os parâmetros para o desenvolvimento inicial do método de escalonamento bem como do algoritmo. Assim, a escolha de verificar e registrar as diferenças das aplicações tornou viável, por meio de simulações, definir o que é mais importante computacionalmente na questão do provisionamento de recursos de forma sustentável, logo possibilitou melhorias e ideias de estratégia quanto ao escalonamento desses tipos de aplicações em uma nuvem computacional energeticamente eficiente do ponto de vista do escalonamento.

A execução dos testes ocorreu em uma máquina com as seguintes características:

- Processador Intel Core i7-2600;
- 8 GB de memória RAM;
- 500 GB de espaço em disco rígido;
- Rede Gigabit Ethernet; e
- Virtualizador KVM.

Para capturar o consumo energético poderiam ter sido utilizados sensores específicos como Wattímetros, possibilidade descartada devido aos custos de aquisição serem elevados e a necessidade do servidor a ser medido estar fisicamente no mesmo local que o recurso humano.

Outra alternativa seria utilizar a *Intelligent Platform Management Interface* (IPMI), que é uma interface padronizada para gerência de *hardware* utilizada por administradores de sistemas para monitoramento de servidores (ROUSE, 2007b) (INTEL, 2014). Essa interface coleta as informações utilizando sensores que são os responsáveis por disponibilizar informações em tempo real (LIMA, 2012). Os administradores podem utilizar as mensagens da IPMI para monitorar o estado da plataforma por meio de informações como temperatura, tensão, estado da ventilação e fontes de energia. Também é possível efetuar operações de recuperação remota, configurando o desligamento e reinício. Porém, essa interface está disponível em servidores com tecnologias atuais, mas precisa ser configurada por meio de ferramentas de *software*.

No caso deste trabalho, o computador utilizado não é exatamente um servidor e não possui essa interface então abandonou-se a ideia de utilizá-la. Porém o que se busca

são as diferenças de consumo energético em relação ao tipo de computação, portanto a preocupação com a arquitetura do equipamento não foi levada em consideração apesar de ser um ponto a ser estudado futuramente.

Diate disso, foi encontrada na literatura uma alternativa para a execução das coletas, a IAEE (DUSSO, 2011), uma ferramenta de *software* para estimar o consumo energético de um computador. Esta ferramenta foi desenvolvida na plataforma Java e utiliza agentes para coletar os dados. Uma instância atua como mestre recebendo as informações das outras instâncias, os agentes, que atuam em cada servidor. Os agentes enviam os dados de seus respectivos hospedeiros à instância mestre, em um intervalo de tempo. Essa ferramenta considera os dados a partir de um modelo energético de alto nível para efetuar a estimativa de consumo. Esse modelo é representado pela equação a seguir.

$$P_{TOTAL} = U_{CPU} \times TDP_{CPU} + U_{DISK} \times TDP_{DISK} + K_{MEM} + K_{NET} + K_{MISC}$$

Na expressão acima, U_{CPU} e U_{DISK} representam respectivamente o percentual de utilização do processador e o percentual de utilização do disco. Esses percentuais são multiplicados pelos seus respectivos valores de *Thermal Design Power* (TDP), que representa a potência máxima que cada componente dissipa ao executar aplicações. Estes valores de TDPs são obtidos nas especificações dos fabricantes dos componentes de *hardware*.

K_{MEM}, K_{NET} e K_{MISC} representam respectivamente a constante de energia consumida pelas memórias RAM, a placa de rede e os outros componentes de *hardware*.

P_{TOTAL} é a potência total do *hardware* diante do consumo de cada componente atribuído as variáveis descritas.

Para calcular a energia consumida em determinado período de tempo basta multiplicar o P_{TOTAL} pelo intervalo de tempo desejado. Esse modelo é preciso o suficiente (aproximadamente 10% para a maioria dos *workloads*) para ser considerado como uma opção viável para mensurar o consumo energético de sistemas computacionais (ALLES, 2013).

Esse modelo não provê a precisão que outros modelos conseguem, mas se mostra uma opção interessante pois é não-intrusiva, de baixo custo (não precisa de instrumentos de medição), rápida e de alta portabilidade (utiliza variáveis e propriedades já disponíveis em sistemas operacionais) (ALLES, 2013).

O IAEE é uma ferramenta que foi desenvolvida para a coleta do consumo energético de um sistema de banco de dados (DUSSO, 2012). O princípio de utilização dessa ferramenta

para este trabalho foi o mesmo. Durante a execução dos *benchmarks* o monitor IAEE foi executado em paralelo para aferir a execução e o consumo dos componentes de *hardware* citados.

Sabe-se que alguns componentes não consomem energia proporcionalmente ao seu nível de utilização, ou seja, eles gastam a mesma quantidade de energia independente do quão requisitados são (DUSSO, 2012). Esse é o caso das placas de memória que precisam ser completamente energizada independente de estarem usando 0% ou 100% de sua capacidade, ao contrário das CPUs modernas que possuem a capacidade de desligar partes e gerenciar a própria energia de acordo com o nível de utilização.

Nesse trabalho, os valores adotados para as potências dos componentes estáticos foram de 24,47 W para o total de pentes de memória, 0,66 W para o componente de rede e 100 W para miscelâneas. O disco possui um TDP de 7,2 W e apesar de não ter um consumo estático, a sua variação é muito pequena em relação a CPU, logo, para efeitos de cálculos, o seu consumo foi considerado estático também. Esses valores são somados ao valor médio da CPU e essa soma é multiplicada pelo tempo de execução. Por fim obtém-se o gasto de cada execução de VM que servirá para alocar e realocar as mesmas quando possível e necessário. Dessa forma, o modelo adotado na implementação da IAEE foi simplificado e alterado, assim a constante K_{MISC} passa a abranger todos os componentes com consumo estático, inclusive o disco. O modelo de alto nível então se dá por:

$$P_{TOTAL} = U_{CPU} \times TDP_{CPU} + K_{MISC}$$

O *benchmark* selecionado para representar as aplicações do tipo CPU-*bound* foi o HPL do pacote Linpack (PETITET, 2012) (utilizado pelo TOP 500 (TOP500.ORG, 2013) para a elaboração de seus *rankings*) enquanto que o escolhido para representar as aplicações IO-*bound* foi o IOZone (NORCOTT, 2006). O HPL resolve problemas de álgebra linear densa ao qual abrange cálculos de vetores e matrizes. Essa aplicação estabelece um padrão computacional intenso de CPU e é referente a muitas aplicações HPC. Ela possui a maior abrangência de campo de atuação, ou seja, é a que corresponde a um maior número de problemas em variadas áreas de pesquisa. O valor de carga definido para executar os experimentos com esse *benchmark* foi baseado na recomendação de (CLUSTERING, 2014).

Quando se aumenta o número de processadores para executar a tarefa, a recomendação é também aumentar a carga (e consequentemente a quantidade de memória), pois

isso otimiza o trabalho computacional. Entretanto, estabeleceu-se a melhor carga para a máquina virtual menos potente e depois repetiu-se essa carga para as demais, independente da quantidade de processadores e memória, para obter também uma relação de desempenho (tempo de execução) por custo energético (total de Watts).

O problema resolvido pelo HPL pode ser executado de forma paralela, dividindo-se o mesmo em partes e cada processo executando a sua parte. Enquanto a carga no HPL é o tamanho das matrizes, no IOZone é o tamanho do arquivo a ser escrito e lido.

O IOZone simula acessos a arquivos, leituras e escritas em alguns diferentes modos. Para esse *benchmark*, a ideia de não variar a carga entre os diferentes ambientes foi mantida pelas mesmas razões. O IOZone executa várias operações, aleatórias ou sequenciais, de leitura e escrita de arquivos em disco. Ele também é capaz de paralelizar a resolução do problema, logo variou-se também o número de processos da mesma forma que no HPL, aferindo-se as informações das execuções da tarefa para comparação entre os ambientes.

Quanto a carga estabelecida para as aplicações HPL e IOZone, essas foram fixadas. Isso limitou o universo dos testes e consequentemente a abrangência do próprio trabalho, mas o objetivo aqui foi encontrar uma estratégia que conseguisse usar menos energia possível com o maior desempenho possível.

4.3 ANÁLISE DAS MEDIÇÕES

As medições possibilitaram a observação do comportamento das VMs, contribuindo assim com a formulação de uma ideia central do método de escalonamento para ambientes que alocam VMs que executem essas aplicações (CPU-*bound* e IO-*bound*).

Na Tabela 4.3 são apresentados os dados das medições referente aos dois *benchmarks*, no que diz respeito a execução dos mesmos para a resolução de suas tarefas de forma exclusiva em um servidor.

No gráfico da Figura 4.3 são apresentados os ambientes e sua relação de tamanho (tamanho das bolhas) com o consumo (eixo y) pelo tempo (eixo x).

Como a avaliação abrangeu duas aplicações de características computacionais diferentes, normalizar o desempenho de ambas não foi necessário, pois não existe a necessidade de comparar seus desempenhos entre si. Foram apenas aferidos o seus desempenhos na mesma escala de tempo de execução para que a avaliação, análise e exposição dos resultados ficasse melhor visualmente.

Pelas suas próprias naturezas, a ideia de aferir as duas de forma separada foi que seus

TAB. 4.1: Dados dos *benchmarks* HPL e IOZone

Tipo	vCPU	vRAM (MB)	Tempo de execução (decimal)	CPU (W)	HD (W)	Memória (W)	Rede (W)	Misc (W)	Consumo total (Wh)
CPU-bound	1	768	2,68	35,98	7,2	24,47	0,66	100	**451,07**
	2	1.536	1,96	44,76	7,2	24,47	0,66	100	**347,10**
	4	3.072	1,53	81,3	7,2	24,47	0,66	100	**326,85**
IO-bound	1	768	2,8	36,04	7,2	24,47	0,66	100	**471,44**
	2	1.536	2,58	38,33	7,2	24,47	0,66	100	**440,30**
	4	3.072	2,35	35,66	7,2	24,47	0,66	100	**394,78**

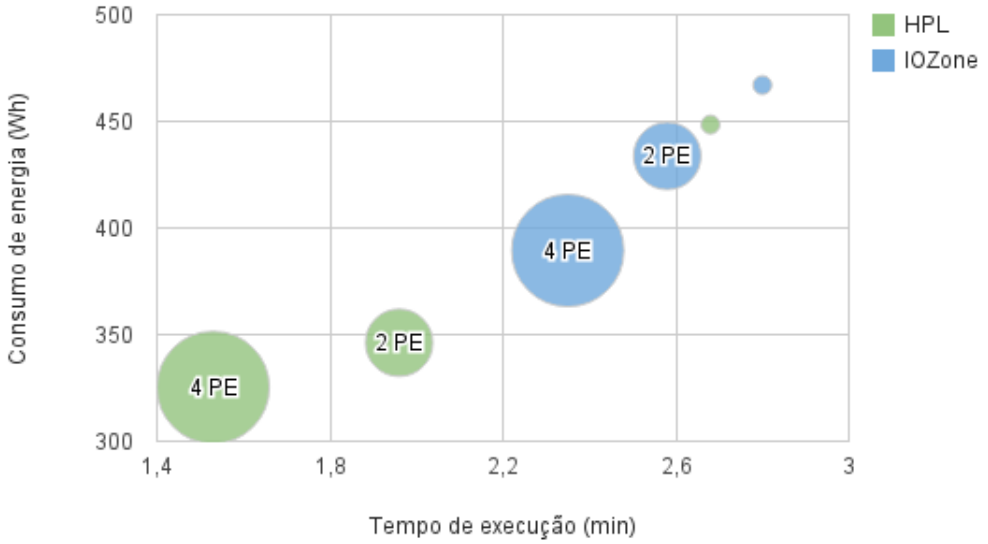


FIG. 4.1: Relação tempo de execução/energia dos ambientes e *benchmarks*

comportamentos eram requisito para ser tratado pelo método do escalonador. Ou seja, o mesmo precisa escalonar as aplicações independentemente da relação de suas cargas de trabalho. O foco estudado está na relação entre desempenho e energia despendida, considerando as diferenças computacionais das aplicações. Suas execuções foram aferidas individualmente e simultaneamente para poder verificar o quanto uma pode afetar no desempenho da outra, quando em concorrência pelos recursos do servidor.

Na tabela 4.3 é apresentada a estimativa que possibilitou a comparação de algumas execuções de cada ambiente no servidor, estabelecendo as previsões de gasto e consumo das VMs na execução das tarefas.

Após essa coleta foi necessário também medir algumas possibilidades dos ambientes

TAB. 4.2: Conjunto de ambientes em execução

Benchmark	Ambiente	Tempo de execução (decimal)	CPU (W)	Consumo de energia (Wh)	Nº de instâncias p/ servidor	Ocupação da RAM do servidor (%)	Consumo energético p/ VM (Wh)
HPL	1 vCPU e 768 MB de RAM	2,68	35,98	451,07	1	10,67	451,07
		2,68	47,98	483,23	2	21,34	241,61
		2,68	59,98	515,39	3	32,01	171,79
		2,68	71,98	547,55	4	42,68	136,88
	2 vCPUs e 1536 MB de RAM	1,96	44,76	347,09	1	21,34	347,09
		1,96	64,76	386,29	2	42,68	193,14
	4 vCPUs e 3072 MB de RAM	1,53	81,3	326,85	1	42,68	326,85
IOZone	1 vCPU e 768 MB de RAM	2,8	36,04	471,43	1	10,67	471,43
		2,8	38,04	477,03	2	21,34	238,51
		2,8	40,04	482,63	3	32,01	160,87
		2,8	42,04	488,23	4	42,68	122,06
	2 vCPUs e 1536 MB de RAM	2,58	38,33	440,30	1	21,34	440,30
		2,58	52,33	476,42	2	42,68	238,21
	4 vCPUs e 3072 MB de RAM	2,35	35,66	394,77	1	42,68	394,77

executando em concorrência, superlotando recurso do servidor.

Nas tabelas 4.3 e 4.3 são apresentadas as degradações de desempenho quando os ambientes relacionados são executados concorrentemente com ambientes iguais, para cada tipo de aplicação. Isso destaca a importância do escalonador não alocar VMs que possam atrapalhar outras que estejam executando suas tarefas, mas para isso é preciso ter o conhecimento do tipo de recurso que está sendo utilizado e assim otimizá-los, visto que aplicações que consumam os mesmos recursos, se executadas sobre os mesmos terão seus desempenhos afetados e consequentemente gastarão mais energia.

TAB. 4.3: Concorrência de execução de VMs CPU-*bound*

Benchmark	Instâncias de VMs	Recursos utilizados		vCPUs	Tempo (minutos)	Perda (%)
		Núcleos de CPU	Dispositivos IO			
HPL	1	4	-	4	1,53	-
	2	8	-	8	4,71	208
	4	16	-	16	11,12	627

A medição apresentada na tabela 4.3 mostra uma possibilidade que precisa ser considerada. O escalonamento pode alocar VMs de tipos diferentes ainda que o número de núcleos de processamento do servidor seja extrapolado. A condição é que VMs que sejam intensivas de CPU executem seus processo nas CPUs e as VMs que sejam intensivas de IO, apesar de virtualmente ocuparem esses mesmos núcleos, devido a sua natureza, não

TAB. 4.4: Concorrência de execução de VMs IO-*bound*

Benchmark	Instâncias de VMs	Recursos utilizados		vCPUs	Tempo (minutos)	Perda (%)
		Núcleos de CPU	Dispositivos IO			
IOZone	1	-	1	4	2,35	-
	2	-	2	8	4,78	103
	3	-	3	12	5,51	134
	4	-	4	16	6,15	162

atrapalham o processamento (no caso elas não ocupam o núcleo e sim o componente de IO). A perda de desempenho é inexpressiva, diferentemente dos mesmos tipos de VM quando executam concorrentemente.

TAB. 4.5: Concorrência de execução de VM CPU-*bound* com VM IO-*bound*

Benchmark	Instâncias de VMs	Recursos utilizados		vCPUs	Tempo (minutos)	Perda (%)
		Núcleos de CPU	Dispositivos IO			
HPL	1	4	-	4	1,57	2,7
IOZone	1	-	1	4	2,38	1,3

Nessa analise é compreendido que o aproveitamento inteligente dos componentes de *hardware* é um fator significativo para escalonar aplicações de naturezas distintas. Enxergar o comportamento das mesmas e, assim então, buscar a sua alocação ajuda a economizar energia ao passo que novos servidores não precisam ser ligados para atender a essas demandas.

4.4 DESCRIÇÃO DO MÉTODO

Baseado nas medições e analises, o método foi desenvolvido e ele é constituído de três caraterísticas principais. São elas:

a) Empilhamento de VMs:

   Para haver empilhamento de VMs, ou seja, VMs dividirem tempo de CPU do servidor na execução de suas tarefas com outras VMs, uma regra precisa ser respeitada. Essa regra estabelece que:

   - Somente uma VM IO-*bound*, independente de seu tamanho, pode ser executada por vez;

- Cada núcleo de processamento só pode estar referenciado a uma VM CPU-*bound* por vez;
- A quantidade total de memória real disponível no servidor não pode ser excedida pelas instâncias das VMs em execução.

Então, um exemplo de configuração de VMs alocadas que pode gerar empilhamento de VMs é quando um determinado servidor possuir somente VMs do tipo CPU-*bound* em execução e a próxima VM da fila a ser escalonada for uma VM IO-*bound*. Caso exista espaço de memória disponível nesse servidor, essa VM IO-*bound* pode ser instanciada ainda que as VMs CPU-*bound* estejam ocupando todos os núcleos de processamento, ou seja, caso exceda a capacidade de processamento do servidor. VMs IO-*bound* quase não consomem processamento, logo o empilhamento não onera o trabalho das VMs CPU-*bound*. O empilhamento economiza a energia do *datacenter* pois evita o ligamento de novos servidores para executar mais VMs além de não afetar de forma impactante o desempenho das tarefas a serem executadas.

b) Reorganização de recursos por concentração:

A reorganização dos recursos também evita o ligamento de novos servidores para execução de outras VMs. Caso exista a possibilidade, as VMs em execução nos servidores são realocadas de forma a se conseguir mais espaço concentrado em um servidor, podendo assim alocar as próximas VMs da fila que precisem de espaço para tal. Essa reorganização se dá respeitando-se os três pontos descritos na regra do item 1 desse método. Um exemplo dessa reorganização está sendo demonstrado no cenário da Figura 4.2:

Os quadrados e retângulos verdes representam as VMs CPU-*bound*, os azuis as VMs IO-*bound*. Os caracteres "S" e "M" representam respectivamente VMs com 1 processador virtual e 768 MB de RAM e 2 processadores virtuais com 1536 MB de RAM. Esta ilustração representa apenas uma configuração de disposição de VMs nos servidores que permite a reorganização de recursos, portanto, caso os servidores possuam recursos desocupados, que no seu somatório atendam aos requisitos da próxima VM da fila, e caso as VMs em execução nesses servidores possam ser realocadas entre os próprios servidores, então uma rearrumação acontece de forma a deixar um servidor mais desocupado do que o outro para que possa receber a VM da fila em questão.

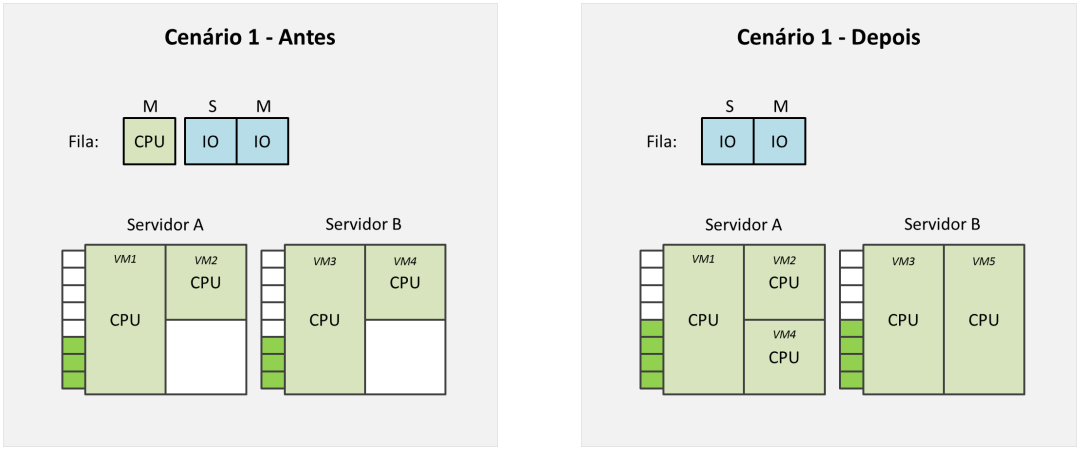


FIG. 4.2: Cenário 1 antes e depois da reorganização por concentração

A VM CPU-*bound* da fila pôde ser escalonada para o Servidor B a partir do momento que a VM4 migrou para o Servidor A, deixando desocupado dois processadores no Servidor B. Futuramente, neste cenário ainda, o escalonador poderá escalonar as próximas VMs da fila (VMs IO-*bound*) uma para cada servidor devido a ideia de empilhamento vista no item anterior, não havendo necessidade de ligar outro servidor para atender a demanda.

Na Figura 4.3, a seguir, está representado um segundo cenário de rearrumação, de maneira mais abrangente. Nesse caso podemos notar que essa reorganização dos recursos, respeitando as regras de alocação, conseguiu esvaziar quatro dos nove servidores ligados.

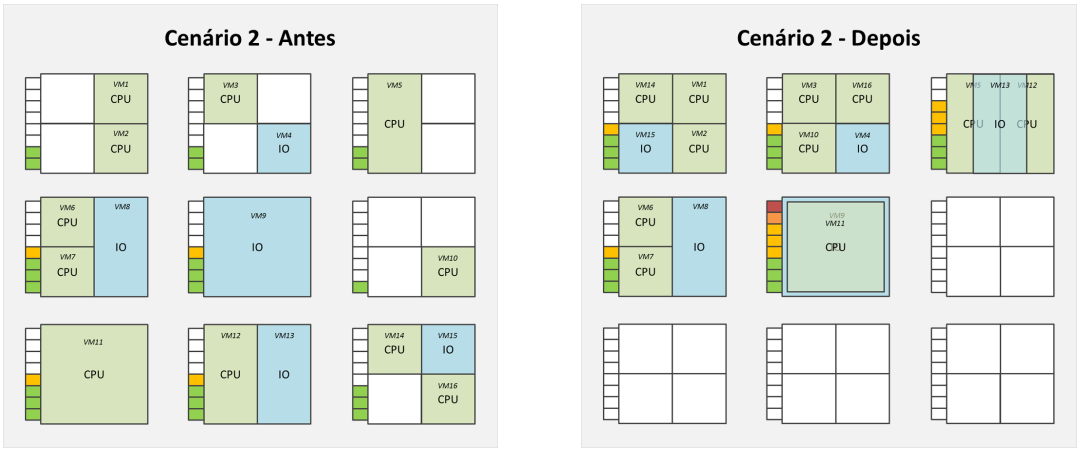


FIG. 4.3: Cenário 2 antes e depois da reorganização por concentração

c) Desligamento de servidor:

Na medida em que as VMs vão terminando de executar suas tarefas e vão sendo desligadas, os servidores começam a ganhar espaço. Esses espaços começam a ocasionar uma fragmentação de recursos. Ou seja, uma quantidade Q de recursos fica disponível, mas se não houver uma reorganização, podem não ser aproveitados. Isso foi visto no item anterior. Então, a etapa seguinte, após a reorganização, é desligar os servidores que estão ociosos.

Para estabelecer o desligamento desses servidores, um pré-escalonamento é executado em memória atribuindo, por meio de um mapeamento, as VMs da fila que estão aguardando aos servidores necessários. A cada *round* do escalonador essa ação é executada estabelecendo-se o destinos das VMs e, portanto, viabilizando a contagem de servidores a serem desligados, baseado no cálculo dos recursos que precisam estar disponíveis. Isso se justifica pelo fato de que religar um servidor custa energia e tempo, este que acaba se adicionando ao tempo de execução de uma VM que necessitou do recurso e o mesmo não estava pronto, ainda, para atendê-la.

Na Figura 4.4 são mostrados quais servidores seriam desligados. Pode-se notar que os recursos virtuais são formados por 3 VMs sendo 2 de tamanho M (uma CPU-*bound* e outra IO-*bound*) e 1 de tamanho P (IO-*bound*). Entretanto, pelas características dos ambientes, os recursos reais são formados por:

- 2 vCPUs para execução de tarefa CPU-*bound* referente a VM CPU-*bonud*;
- 2 IO *devices* para execução de tarefas IO-*bound* referente a VM IO-*bonud*;
- 3848 MB de RAM referente ao total de RAM requisitada pelas 3 VMs;

Essa ideia de estratégia de desligamento, até o presente momento, não foi implementada, entretanto faz parte de trabalhos futuros que visam aprimorar esse método.

Acredita-se que pela representação das características computacionais estabelecidas pelos *benchmarks*, qualquer ambiente virtualizado que execute VMs com aplicação de mesmas características das relacionadas neste trabalho possa ser controlada e escalonada por esse método de escalonamento que foi desenvolvido e continua sendo aprimorado, visto que computacionalmente suas características de consumo aos recursos seriam correspondentes. mas até o presente momento, nenhum teste de validação com estas outras aplicações foi executado.

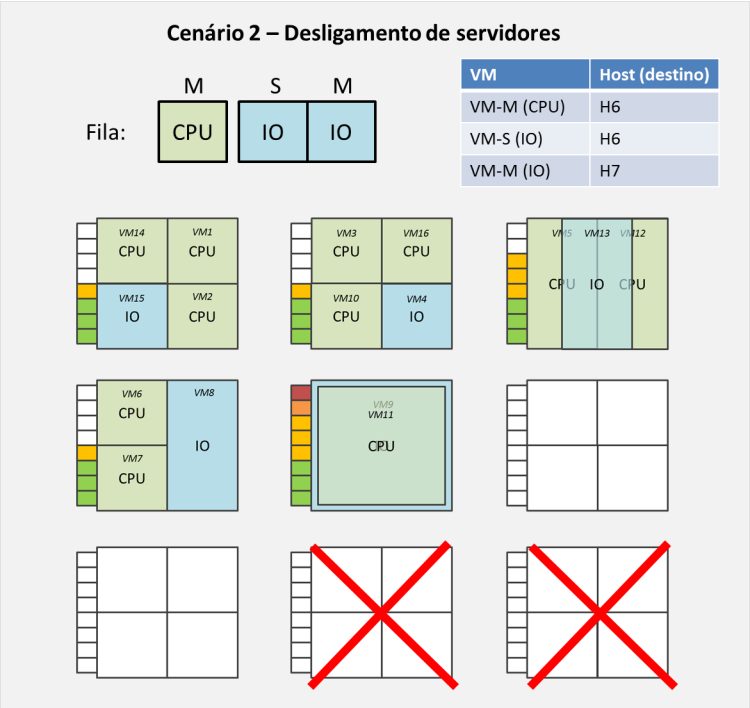


FIG. 4.4: Desligamento de servidores após a reorganização dos recursos no Cenário 2

4.5 ALGORITMO DESENVOLVIDO

O algoritmo apresentado a seguir, na Figura 4.5, é a ideia central do método de escalonamento descrito. As chamadas e sub-rotinas presentes no método são abstratas e foram implementadas na validação ou adaptadas para trabalhar de acordo com as características do simulador apresentado mais a frente, na seção 5 deste trabalho.

A questão do desligamento dos servidores ociosos também depende da forma como o simulador trabalha, e no caso desse trabalho, não foram implementados por conta das características do mesmo.

Esse algoritmo foi desenvolvido para simular e acompanhar o escalonamento de VMs em servidores. Todo o seu processo aconteceu em memória com objetos Threads do Java.

A ideia de implementação em memória, simulando um ambiente simplificado somente com essas VMs e servidores, teve o objetivo de analisar o comportamento desses componentes, não considerando, nesse ponto, nenhuma outra arquitetura ou característica relacionada a um ambiente ou *framework*, pois, nesse caso, outras implementações e adaptações necessitariam ser implementadas. Um exemplo disso é a própria implementação no simulador, que aconteceu posteriormente, para poder executar as validações. Nela, o algoritmo implementado foi modificado em alguns detalhes para poder adaptar-se às

```
loudVMsQueue(); loudServersQueue(); int vmIndex = 0;
loopVMs:
while (queue.size() != 0) {
    if (onlineServers.size() == 0) {
        powerOnAServer();
        continue loopVMs;
    }
    loopServers:
    for (int serverIndex = 0; serverIndex < onlineServers.size(); serverIndex++) {
        if (queue.size() != 0){
            if (thereIsFreeMemToVM(onlineServers.get(serverIndex), queue.get(vmIndex))){
                if (thereIsFreeCpuToVM(onlineServers.get(serverIndex), queue.get(vmIndex))){
                    if (!thereIsRunningIOVM(onlineServers.get(serverIndex))){
                        onlineServers.get(serverIndex).runVM(queue.get(vmIndex));
                        queue.remove(vmIndex);
                        continue loopVMs;
                    } else if (queue.get(vmIndex).getType().equals("io-bound")){
                        if (onlineServers.lastIndexOf(onlineServers.get(serverIndex)) == serverIndex)
                            powerOnAServer();
                        continue loopServers;
                    } else {
                        onlineServers.get(serverIndex).runVM(queue.get(vmIndex));
                        queue.remove(vmIndex);
                        continue loopVMs;
                    }
                } else {
                    if (canSobreposit(queue.get(vmIndex), onlineServers.get(serverIndex))){
                        onlineServers.get(serverIndex).runVM(queue.get(vmIndex));
                        queue.remove(vmIndex);
                        continue loopVMs;
                    } else
                        continue loopServers;
                }
            } else if (onlineServers.lastIndexOf(onlineServers.get(serverIndex)) == serverIndex) {
                if (waitForTime() <= onlineServers.get(serverIndex).BOOT_TIME) {
                    try {
                        Thread.sleep(waitForTime());
                    } catch (InterruptedException ex) {}
                } else
                    powerOnAServer();
                continue loopServers;
            } else
                continue loopServers;
        }
        vmIndex++;
    }
}
```

FIG. 4.5: Algoritmo em Java de escalonamento de VMs CPU-*bound* e IO-*bound*

particularidades do mesmo. Como exemplo, uma dessas mudanças foi o processo de ligamento dos servidores, pois esse processo não existe. Por padrão, é necessário definir a quantidade de servidores que estarão disponíveis para atender ao ambiente, ou seja, os *hosts* que compõem o *datacenter*, mas os mesmos não possuem nenhum comando ou tratamento para verificar seus *status*.

Portanto, com a implementação inicial em Java Threads foi possível assistir a interação das VMs com os servidores de acordo com suas características, apenas para garantir o

comportamento do método. No fluxograma da Figura 4.6, a seguir, pode-se compreender o comportamento do método.

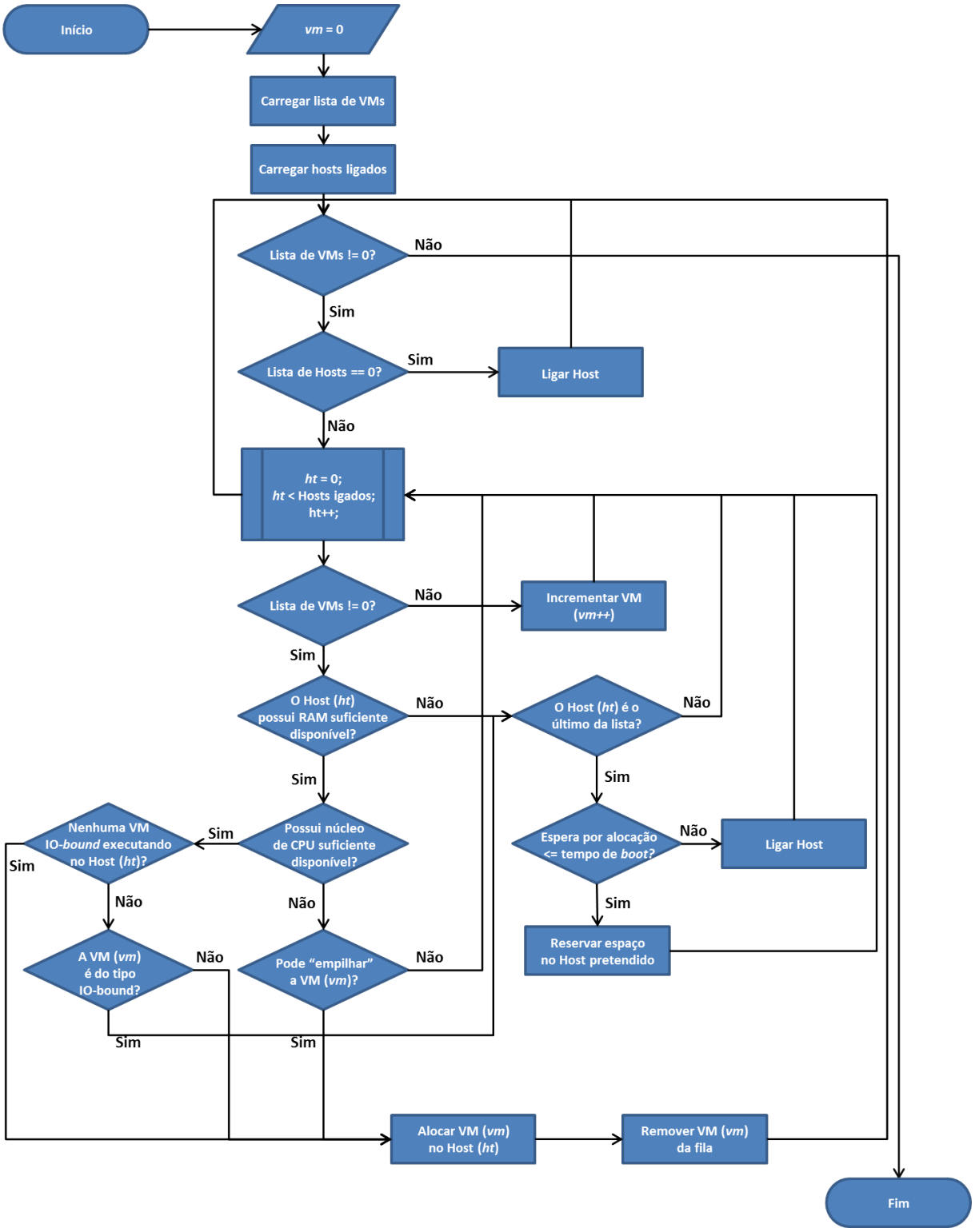


FIG. 4.6: Fluxograma do método de escalonamento

Enquanto a fila de VMs a serem escalonadas não chegar ao fim, se nenhum servidor estiver ligado então liga-se um novo servidor.

Para cada servidor ligado, se a fila de VMs a serem escalonadas não estiver vazia então verifica-se se existe memória RAM livre no servidor em questão para alocar a VM. Existindo memória RAM, verifica-se se existe CPU livre pra executar a VM. Caso existam núcleos de CPU suficiente disponíveis, verifica-se se não existe VM IO-*bound* já executando. Caso não exista essa quantidade de núcleos de CPU disponíveis, aloca-se a VM em questão no servidor e remove-se a mesma da fila de escalonamento.

Caso, neste servidor, exista alguma VM IO-*bound* executando, então verifica-se se a VM da fila, na posição atual, é IO-*bound*. Caso seja, então é preciso ligar um novo servidor para alocá-la. Caso não seja, escalona-se a VM para o servidor em questão e remove-se a mesma da fila de escalonamento.

Se não existir CPU disponível e se a VM da fila puder ser alocada por empilhamento, então escalona-se a VM para esse servidor e remove-se a mesma da fila de escalonamento. Caso não possa ser alocada por empilhamento, então procura-se um novo servidor na lista. Porém se não existir memória RAM livre no servidor, se a VM não foi escalonada e se a lista dos servidores ligados chegou ao última posição, então verifica-se se o tempo de espera por um servidor será menor do que o tempo de *boot* de um novo servidor. Sendo menor, então reserva-se o espaço necessário para a VM nesse servidor alvo e espera-se pelo tempo de desalocação do recurso para poder alocá-la. Caso contrário, se o tempo de espera for maior, liga-se um novo servidor para alocá-la.

Como esse algoritmo de escalonamento foi construído baseado na metodologia de avaliação descrita e na analise dos dados coletados, isso reforça a ideia de que o conhecimento prévio das características do ambiente é de extrema importância para se identificar os pontos-chave dos problemas a serem minimizados ou mesmo resolvidos. Nele só não foi implementada ainda a ideia de desligamento dos servidores ociosos.

5 VALIDAÇÃO E RESULTADOS

Para validar o método de escalonamento desenvolvido nesse trabalho e poder comparar seus resultados, a ideia inicial de implementar o algoritmo em Java Threads não se mostrou uma boa opção pelo fato de não simular um ambiente de nuvem, desprezando assim componentes e aspectos importantes como características de servidores, *brokers* e redes, bem como cálculo de SLA, alocação de instruções em grandeza de milhões de instruções por segundo (MIPS), impacto de migrações, dentre outros.

Na implementação em Java Threads foi desenvolvido somente o comportamento básico de um escalonador, considerando o escopo do trabalho, em que só objetos envolvidos no escalonamento, como servidores, VMs e tipos de aplicação (CPU-*bound* e IO-*bound*) fossem controlados de forma a restringir e diminuir a complexidade de um ambiente completo de nuvem em favor da observação do comportamento desses componentes. Porém, com isso, percebeu-se que para efeito de validação não seria suficiente, visto que acompanhar as variáveis e componentes de um ambiente de nuvem agregaria mais valor. Além disso, implementar outros algoritmos de escalonamento em Java Threads, para efeito de comparação, seria um esforço sem compensação, visto que esse ambiente simulado é bem restrito em relação aos componentes, como descrito.

Logo, duas opções restaram em relação a essa validação: a simulação em um *framework* disseminado no mundo acadêmico ou a criação de um ambiente de nuvem real e a implementação do escalonador nesse ambiente. O primeiro foi escolhido devido ao tempo e aos baixos custos de implantação, implementação e utilização, bem como a ferramenta escolhida ser bastante disseminada no mundo acadêmico, agregando assim maior confiabilidade aos dados e resultados.

O simulador escolhido para esse trabalho foi o CloudSim (CALHEIROS, 2011) em sua versão 2.1.1. CloudSim é um simulador reconhecido academicamente com citações e relacionamento em mais de 3800 referências de trabalhos indexados pelo Google Scholar (GOOGLE, 2004).

O CloudSim inclui várias funcionalidades que são essenciais ao desenvolvimento deste trabalho, no entanto, foi necessário implementar algumas outras funcionalidades que não estavam presentes no mesmo, como por exemplo, as diferenças comportamentais de apli-

cações CPU-*bound* e IO-*bound*, assim como seus impactos quando em concorrência de processamento de IO (ponto-chave desse trabalho).

5.1 ASPECTOS DO SIMULADOR

O CloudSim é um *framework* que possibilita a modelagem e simulação de ambientes de nuvens em seu nível de infraestrutura como serviço (IaaS). Este simulador está focado no provisionamento de recursos para execução de tarefas (*cloudlets*) que rodam em VMs que por sua vez executam em servidores que formam um *datacenter*. Além disso possui suporte para a questão energética já contendo alguns componentes que trabalham com essa característica (BARBOSA, 2014). O CloudSim é um *software* livre, disponibilizado sob a licença GNU LGPL (COMPUTING, 2010).

Quanto aos eventos do simulador, o CloudSim estabelece a criação de um ou mais *datacenters* que contém servidores (*hosts*) onde as VMs dos usuários serão alocadas. Cada servidor (*host*) possui uma capacidade de processamento pré-determinada, medida em milhões de instruções por segundo (MIPS) e podendo ser com apenas um ou vários núcleos de processamento, memória e armazenamento.

Um servidor pode incorporar uma ou mais VMs de acordo com as políticas de alocação definidas pelo administrador da nuvem (*time-share* ou *space-share*). De acordo com a definição, os servidores podem alocar as VMs por compartilhamento de tempo ou por compartilhamento de espaço, ou seja: de acordo com a primeira política, VMs podem compartilhar o processador executando instruções dividindo o seus tempos de execução, ainda que seja em um mesmo núcleo; já pela segunda política elas compartilham o processador de acordo com o espaço disponível nos núcleos em razão da quantidade de instruções requisitada no momento.

Uma política de alocação é definida como as operações relacionadas a uma VM durante seu ciclo de vida que constitui-se da seleção de um servidor, criação, migração e destruição. De maneira similar, cada VM possui capacidades de processamento, memória, largura de banda e número de processadores definidos no momento da sua criação. Ainda, pode executar uma ou mais tarefas (*cloudlets*), obedecendo as políticas de provisionamento de aplicações (CAMPOS, 2013).

Para alocar as VMs e as tarefas, um *broker* precisa ser definido e o mesmo é o responsável por gerir a lista de VMs no ambiente de nuvem bem como a lista de tarefas, ou seja, o *broker* é o componente que demanda as VMs para os servidores que às executarão, bem

como as tarefas que serão executadas pelas VMs. Portanto, além da lista de VMs, uma lista de tarefas também é necessária, pois as mesmas executarão dentro das VMs.

Para a avaliação dos resultados, o CloudSim disponibiliza os seguintes atributos:

- Tempo de execução total do ambiente: O CloudSim calcula o tempo total de execução do ambiente desde o início da simulação até o fim da execução da última tarefa (*cloudlet*), executada em uma VM que fora enviada pelo *broker* ao *datacenter*. Esse tempo é calculado em segundos e equivale ao tempo em que todas as VMs terminam de executar suas tarefas;

- Tempo de execução de cada VM: No CloudSim, cada VM possui uma ou mais tarefas (*cloudlets*) que podem possuir tamanhos diferentes. Esse tamanho de tarefa é formado pela quantidade de instruções que ela precisa executar. Além disso, duas politicas de escalonamento dentro das VMs podem ser selecionadas elas são *space-shared* e *time-shared* (CALHEIROS, 2011). Com isso, a estimativa do tempo de execução de uma VM é obtida pela soma dos tempos das execuções de todas as tarefa de acordo com a capacidade de processamento das VMs em seus respectivos servidores;

- Quantidade de VMs executadas: Ao estabelecer a lista de VMs que o *datacenter* recebe por meio do *broker*, no inicio de sua execução o CloudSim tenta alocar, por meio de alguma politica de alocação de VMs pré-definida, as VMs nos servidores. Dependendo das restrições e características dessa politica, o *broker* consegue enviar as VMs para os seus servidores destinos ou, simplesmente, o mesmo adia sua alocação. Ao adiar a alocação, o CloudSim considera que aquelas VMs adiadas não irão executar mais naquela simulação. Com isso, obtém-se a quantidade de VMs que conseguiu executar e finalizar seus trabalhos;

- Total de energia consumida: O CloudSim acumula o consumo energético de cada servidor ao longo da execução da simulação. É pré-definido um valor de potência em kWh para os servidores e uma porcentagem de consumo estático desses servidores. Já o restante é referente ao consumo variável dos servidores, isso equivale a utilização dos processadores de cada servidor durante todo o tempo da execução da simulação; Nesse trabalho, inicialmente foi estipulado 230 W de potência por servidor e 70% dessa potência foi considerada estática. Os outros 30% equivale a

variação causada pela utilização da CPU, de forma linear. Essa potência foi baseada no consumo do servidor utilizado para a coleta dos dados das aplicações, mas não possui grande relevância pois o importante são os valores comparativos de cada simulação e estratégia;

- Quantidade de migrações de VM: Dependendo da estratégia de escalonamento, suas politicas de aplicação de VMs podem migrar as VMs de um servidor para outro, a cada *round* de escalonamento. Cada migração é contabilizada gerando um total ao final da simulação;

- Quantidade de violações de SLA: Cada tarefa que executa em sua respectiva VM, que requisita uma quantidade de instruções, mas não pode ser atendida integralmente gera uma quantidade de violações de SLA. Para efeito de cálculo, o CloudSim utiliza essa quantidade de requisições não alocadas para gerar o número de violações de SLA baseado em cada instrução não alocada;

- Porcentagem de violações de SLA: Esse atributo é referente a quantidade de violações de SLA que ocorreram a cada *round* de escalonamento em relação as instruções que foram atendidas sem restrição;

- Média global de violações de SLA: Nesse atributo, o CloudSim contabiliza as alocações que geraram violações de SLA em relação as que não geraram.

Quanto a questão do escalonamento de VMs e de suas politicas de alocação, em seu pacote já existe uma implementação de escalonamento simples, aqui chamada de "*Single*" e outra de escalonamento visando eficiência energética, aqui chamada de "*Default*". Uma terceira possibilidade já implementada no CloudSim é o recurso de DVFS que usa a politica de alocação do escalonamento *Default*, entretanto sem migração de VMs, fazendo com que o único recurso de economia de energia seja o ajuste das frequências dos servidores.

A implementação do escalonamento *Single* do CloudSim recebe uma fila de VMs e aloca as mesmas na medida da capacidade dos servidores. Nessa implementação não existe migração de VMs e a política de alocação encontra um servidor com o menor número de processadores em uso e então a VM em questão é alocada. Não existe uma otimização de alocação nessa política. Já a implementação do escalonamento *Default* recebe uma fila de VMs e aloca cada uma de acordo com o consumo energético de cada servidor, buscando o menor consumo entre eles. A cada *round* de escalonamento, a política de

alocação busca otimizar o consumo energético com a realocação das VMs nos servidores. Migrações podem ocorrer devido a esse fator. Já a terceira abordagem, DVFS, atua como a anterior aqui descrita, mas não executa migrações.

5.2 ALTERAÇÕES NO SIMULADOR

Apesar de o CloudSim possuir muitos itens que se encaixam com os aspectos desse trabalho, algumas adaptações precisaram ser desenvolvidas para que as simulações se enquadrassem no contexto dessa pesquisa.

A principal adaptação foi implementar uma forma de diferenciar o comportamento de aplicações CPU-*bound* do comportamento de aplicações IO-*bound*, considerando as características de ocupação e consumo dos recurso já existentes no simulador. Esses comportamentos são diferenciados no que diz respeito ao consumo das unidades de processamento dos servidores. Como visto anteriormente, aplicações IO-*bound* não consomem tando núcleo de processamento de um processador quanto aplicações CPU-*bound*. Entretanto, consomem o componente de IO ao qual precisam ocupar para executar. O CloudSim não implementa a abstração para esse tipo de componente, apenas as CPUs. Portanto, foi preciso adaptar o simulador para considerar essas diferenças.

A forma de adaptá-lo não foi implementando objetos pra processamento de IO, ou seja, abstrações de dispositivos de IO. Essa ideia seria mais pertinente, mas demandaria mais tempo e conhecimento sobre a estrutura do simulador. Portanto, essa questão foi tratada de outra maneira: diferenciando no nível da tarefa o tipo de aplicação que a mesma se enquadraria e, assim, passando essa informação ao objeto que representa a VM.

Assim, foi criado mais um atributo no objeto que representa a tarefa (*cloudlet*) e na própria VM. Ao atribuir uma tarefa que tivesse as características de aplicação CPU-*bound*, a VM responsável por executá-la teria uma configuração para usar 95% de capacidade de um servidor, enquanto que uma VM que recebesse uma aplicação IO-*bound* teria o restante, ou seja, 5% de capacidade de um servidor. Isso foi baseado nos dados dos experimentos que ajudaram a identificar essas diferenças de características dos dois tipos de aplicação e assim, tornando possível ajustar o simulador com essas marcas. Isso foi uma representação aproximada de quanto uma aplicação CPU-*bound* requisita de processamento em relação a uma aplicação IO-*bound*, da ótica do processamento de tarefas pelas CPUs. Isso se fez necessário pois, como observado nas execuções dos experimentos, existe degradação de desempenho quando duas ou mais VMs do tipo IO-*bound* estão executando

em um mesmo servidor que possua apenas um componente desse tipo para executar as tarefas, bem como aplicações CPU-*bound* concorrem por tempo de CPU, devido a esses componentes serem dividido entre mais de uma VM que possua tarefas de mesmo tipo. Isso ocasiona violações de SLA, comprometendo assim a entrega do serviço e demandando mais energia.

Para poder tratar essas diferenças em comparação com os outros métodos já existentes na ferramenta, outra alteração também foi aplicada no CloudSim: a degradação de desempenho de VMs IO-*bound*. Foi implementada uma rotina que causasse a degradação de desempenho em VMs que tivessem que concorrer por um mesmo dispositivo IO. Como já citado, originalmente o CloudSiim não implementa essa abstração então, baseado também nos experimentos anteriores descritos nesse trabalho, uma VM IO-*bound* que fosse alocada em um servidor que já estivesse executando outra VM IO-*bound* causaria essa degradação. Com isso, foi possível verificar a diferença de um método de escalonamento que considera essa característica e um que não considera. O CloudSim já implementava uma degradação de desempenho quanto a migração das VMs, mas, por padrão, não implementa degradação por concorrência de recursos IO.

Sendo assim, no presente trabalho, o método exposto utiliza a informação do tipo de aplicação que uma VM executa. Isso é parte fundamental para que esse método de escalonamento atue da maneira prevista, tentando reduzir o consumo de energia ao minimizar a perda de desempenho e a concorrência por recursos na execução das VMs. A violação de SLA é ocasionada pela perda de desempenho ao alocar menos recursos que o previsto, portanto tratar a relação desempenho-consumo foi o objetivo maior aqui.

Aproveitando as implementações de estratégias de escalonamento já existentes no CloudSim, e aplicando as adaptações no simulador, foi possível estabelecer a execução de algumas comparações com o método exposto nesse trabalho, contribuindo com o prosseguimento da pesquisa e o aprimoramento da mesma. Para isso, o método de escalonamento apresentado nesse trabalho foi implementado e, aqui foi chamado de "*Custom*". Uma outra abordagem também foi criada e mistura aspectos do algoritmo *Custom* com o *Default*, formando assim um escalonamento chamado de "*Hybrid*".

5.3 CONFIGURAÇÕES E CARACTERÍSTICAS DAS SIMULAÇÕES

Considerando o objetivo de validar o método desenvolvido até este momento do trabalho em um simulado, foram estabelecidas algumas características para variar o ambiente de

execução, observar os resultados e viabilizar as comparações e relações. Portanto, o ambiente de execução foi configurado com as seguintes variações de atributos para cada execução de simulação:

- 1 *datacenter*;
- 5 métodos de escalonamento: *Single*, *Default* e DVFS, esses já implementados pelo CloudSim, e *Custom* e *Hybrid*, o primeiro sendo o método puro apresentado neste trabalho e o segundo a combinação da estratégia do *Default* com o *Custom*;
- 50 VMs, cada uma rodando uma tarefa (CPU-*bound* ou IO-*bound*); e
- 50 tarefas (uma para executar em cada VM) variadas em relação ao tipo de computação, sendo 25 CPU-*bound* e 25 IO-*bound*.

Em relação aos métodos de escalonamento, para cada um, as variáveis que foram consideradas nas execuções foram:

- o intervalo de atuação do escalonamento (*round*), variando em 1 minuto, 3:45, 7:30, 15, 30 e 1 hora; e
- a quantidade de servidores para atender as demandas, variando em cinco, dez, 20 e 40 servidores.

O consumo energético dos servidores que compõem o *datacenter* no CloudSim é descrito por uma relação linear entre o consumo energético estático do servidor e a sua variação de utilização da CPU. A potência média consumida por um servidor com CPU em *idle* é de 70% da energia consumida pelo servidor em uso total (SINHA, 2011). Assim, o consumo de energia variável se dá pela variação da utilização de CPU $P(u)$ como mostrado na equação a seguir, onde u é a utilização da CPU e P_{max} é o consumo tota de um servidor, onde foi considerado de 214 W para cada servidores, uma estimativa baseada no somatório dos recursos já descritos na seção 4 deste trabalho:

$$P(u) = P_{max}(0.7 + 0.3u)$$

Entretanto, para estimar o consumo energético ainda é preciso adicionar a variação de utilização da CPU ao longo do tempo (intervalo de tempo *t1* e *t2*), ou seja, $u(t)$. Então, o total de energia consumida (E), é dado por:

$$E = \int_{t1}^{t2} P(u(t))\, dt$$

Portanto, o total de energia é calculado baseado nesse modelo (SINHA, 2011).

Além do consumo energético, o CloudSim calcula também a taxa de violações de SLA. Como já citado, o SLA calculado é considerado sobre as instruções requisitadas e não atendidas no momento dessas requisições. Para esse atributo, três cálculos são executados. Os cálculos desses valores considera a quantidade de VMs que causaram violação de SLA, a quantidade de MIPS não alocados requisitados por todas as VMs em função dos que foram alocados e a quantidade de MIPS não alocados em função da quantidade de VMs que causou violação de SLA.

O primeiro cálculo disponibilizado é a quantidade de violações de SLA (QV). Ela é calculada pelo somatório das alocações em que em algum momento tiveram instruções não alocadas (qtd). Seu cálculo é simples e pode ser representado pela seguinte equação:

$$QV = \sum_{qtd=0}^{N}$$

O segundo é a média geral de violações de SLA (MGS) e ela é calculada pelo somatório das diferenças das instruções requisitadas (RQ_{MIPS}) pelas instruções alocadas (AL_{MIPS}), divididas pelas próprias instruções requisitadas (RQ_{MIPS}). Para esse caso, a equação se dá por:

$$MGS = 100 \times \frac{\sum RQ_{MIPS} - \sum AL_{MIPS}}{\sum RQ_{MIPS}}$$

Já o terceiro cálculo disponível é a média de violações de SLA (MTS). Ela representa a média total das violações de SLA que foram causadas no ambiente pelas VMs. Seu cálculo se dá pelo somatório das MIPS que não foram demandadas mas não alocadas (AL_{MIPS}), pelo somatório das alocações em que houve violação de SLA (QV).

$$MTS = \frac{\sum RQ_{MIPS} - \sum AL_{MIPS}}{QV}$$

Um atributo chamado de *threshold* atua nas estratégias *Default*, *Custom* e *Hybrid*, definindo um limite máximo apenas para a utilização da CPU, tornando as VMs eleitas para migração quando essa utilização no servidor em questão se encontrar abaixo do valor do *threshold* definido. Isso faz com que as VMs que não atuem com no mínimo o *threshold* estabelecido não executem no recurso, deixando de causa sub provisionamento no ambiente.

As estratégias de escalonamento possuem políticas de alocação de VMs que causam variação no desempenho, consumo e SLA no ambiente. Essas diferenças consistem em:

- *Single* – Este algoritmo seleciona do conjunto de todos os servidores disponíveis, os que possuem capacidade de recursos disponível para executar as VMs, alocando-as no servidor com menos capacidade disponível possível. Essa estratégia chama-se *Common Best-fit*;

- *Default* – Essa estratégia aloca as VMs de acordo com o servidor que possui o menor aumento de consumo energético no momento do escalonamento. Após uma verificação da lista de servidores e de suas cargas, a política de alocação seleciona o servidor que encontra-se com o menor consumo, alocando a VM em questão nesse recurso;

- *Custom* – Este é o método implementado neste trabalho. Sua política de alocação evita alocar VMs que concorram pelo mesmo dispositivo de processamento de acordo com suas características de execução de tarefas (CPU-*bound* ou IO-*bound).* Por essa estratégia ser mais restritiva em relação as outras, a vazão de alocação é menor;

- *Hybrid* – Este método combina as estratégias *Custom* e *Default.* A alocação das VMs ocorre em primeiro momento com a política de alocação da estratégia *Custom*,

entretanto, se não conseguir alocar as VMs com essa política, é então acionada a estratégia *Default*. Isso faz com que a vazão de alocação de VMs aumente em relação a estratégia *Custom*;

- DVFS – Nessa estratégia, os servidores ajustam a frequência da sua CPU baseado na sua eficiência energética segundo a equação, mas sempre escolhendo uma frequência que permita a CPU produzir os MIPS necessários para as tarefas que esta tiver alocadas.

As simulações foram executadas da seguinte maneira: para cada uma das estratégias de escalonamento (*Single*,*Default*, *Custom*, *Hybrid* e DVFS) variou-se o número de *hosts* (5, 10, 20 e 40) e o intervalo de tempo de atuação do *round* dos escalonadores (1 minuto, 3:45, 7:30, 15, 30 e 1 hora). Foram executadas um total de 120 simulações, gerando-se os dados apresentados nos gráficos a seguir.

5.4 RESULTADOS OBTIDOS

Os resultados obtidos nesse trabalho se referem aos dados das simulações executadas no CloudSim, contando com as alterações efetuadas nesse *framework*. Experimentos em um ambiente de nuvem real ainda não foram executados devido a maior dificuldade de se dispor de um ambiente como esse. Portanto, os testes comparativos das execuções dos métodos existentes no CloudSim foram consolidados via esta simulação e seus resultados estão sendo apresentados nessa seção.

Nos gráficos apresentados nas Figuras 5.1, 5.2, 5.3, 5.4, 5.5 e 5.6 pode-se observar que os métodos que visam eficiência energética acabam sendo um pouco melhores que o método de alocação simples, entretanto a medida que o *round* aumenta de intervalo, esses métodos vão perdendo sua eficiência em relação ao de alocação simples. O único método que mantém regularidade em relação ao *round* foi o DVFS.

A politica de alocação do método *Custom* é mais restritiva do que os outros, devido a sua impossibilidade de deixar VMs que executam aplicações IO-*bound* serem alocadas em servidores que já executam uma VM desse tipo, em favor de minimizar a degradação de desempenho. A politica do método *Hybrid* permite a primeira alocação como a *Default*, porém, as realocações ao migrar as VMs atuam como a *Custom*.

Quanto ao tempo de execução da simulação, nota-se nos gráficos das Figuras 5.7, 5.8, 5.9, 5.10, 5.11 e 5.12 que o algoritmo de escalonamento implementado neste trabalho

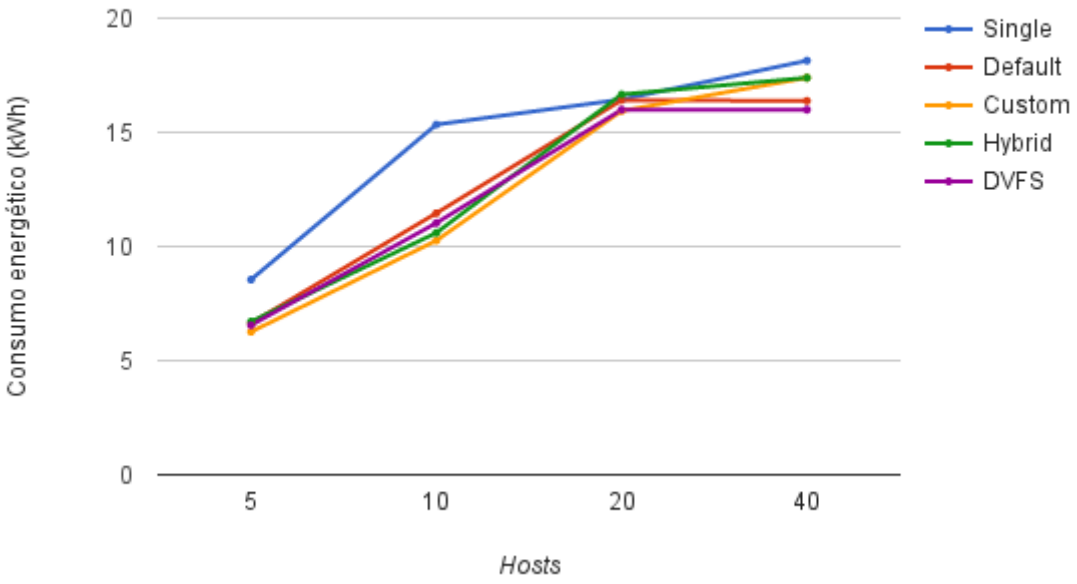


FIG. 5.1: Consumo de energia – Escalonamento com *round* de 00:01:00

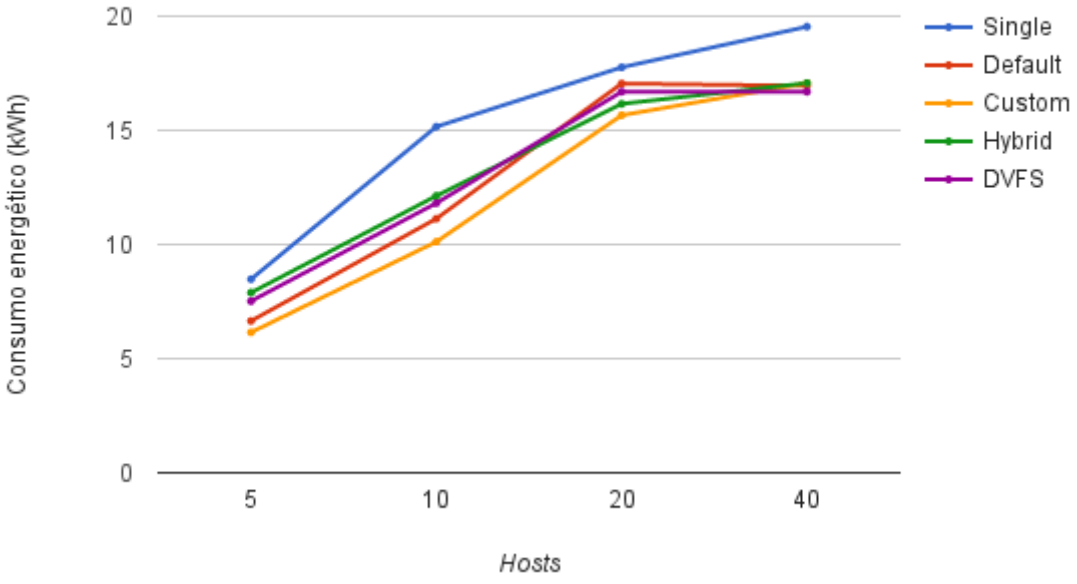


FIG. 5.2: Consumo de energia – Escalonamento com *round* de 00:03:45

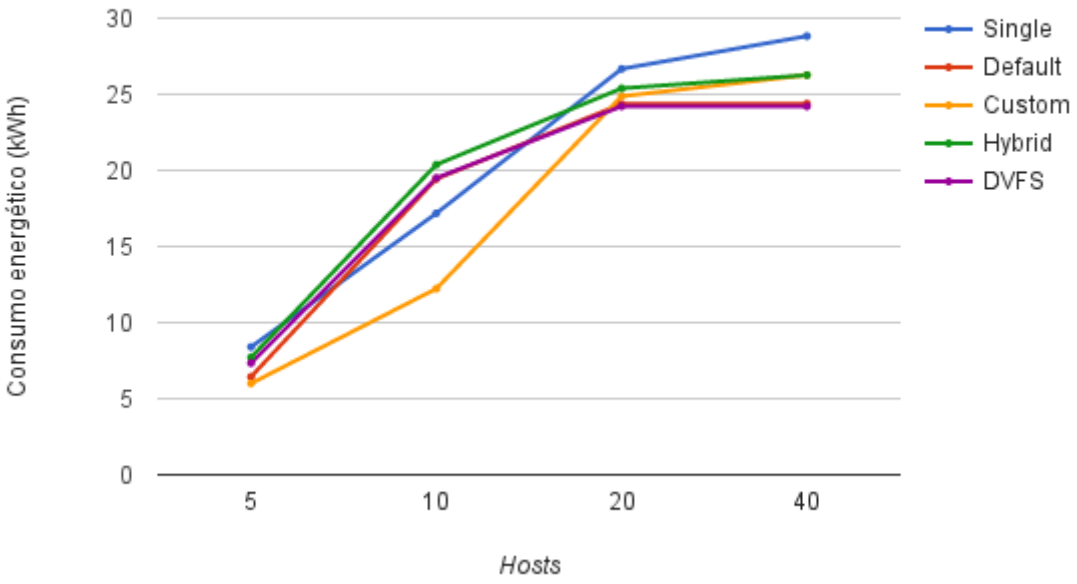


FIG. 5.3: Consumo de energia – Escalonamento com *round* de 00:07:30

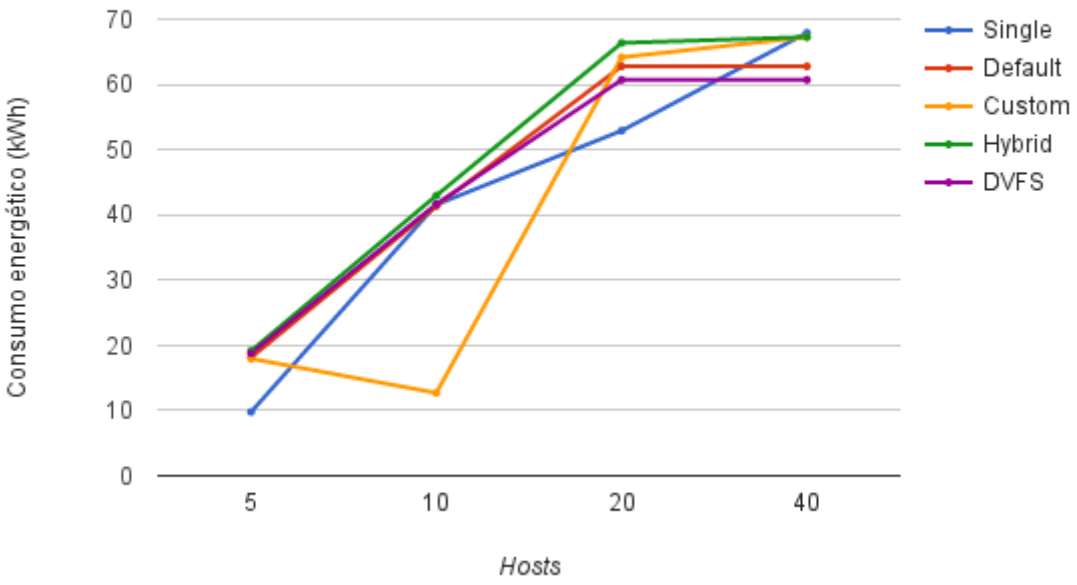


FIG. 5.4: Consumo de energia – Escalonamento com *round* de 00:15:00

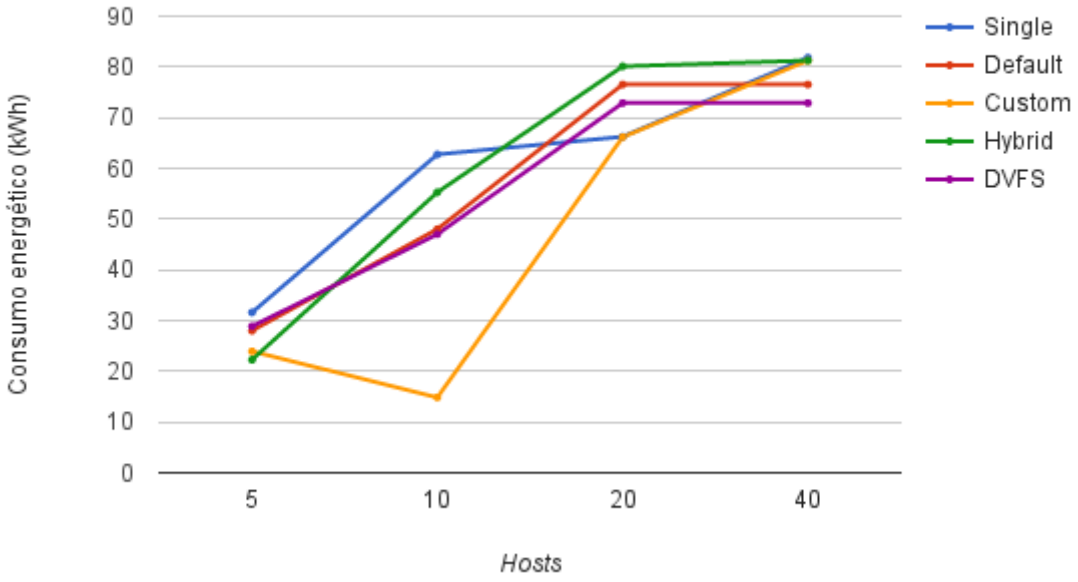


FIG. 5.5: Consumo de energia – Escalonamento com *round* de 00:30:00

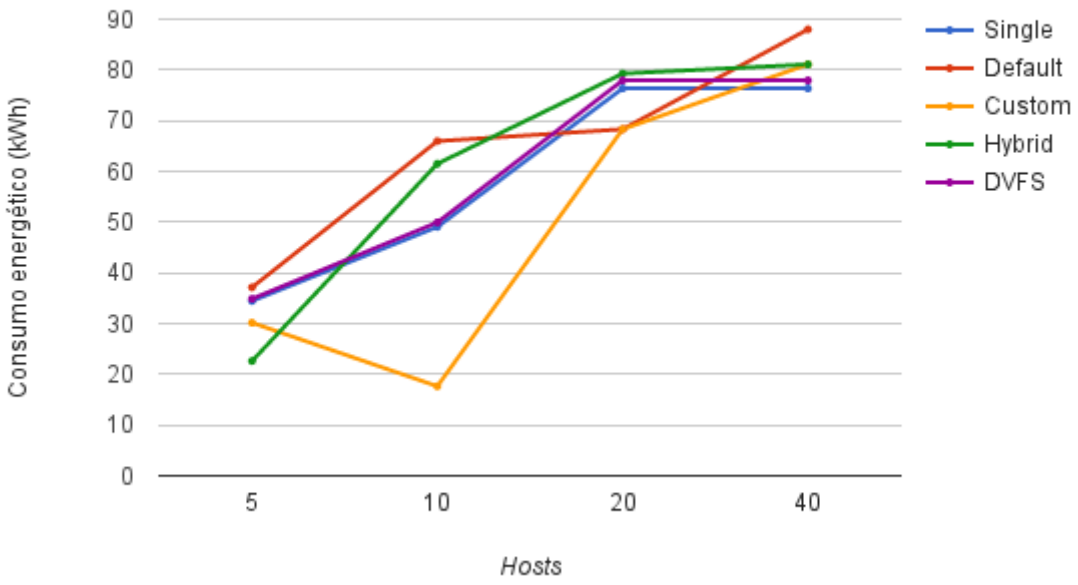


FIG. 5.6: Consumo de energia – Escalonamento com *round* de 01:00:00

possibilita que os tempos de execução das VMs sejam mantidos, em geral, mais baixos que os do método *Default*. Os métodos *Single* e DVFS não executam migrações, portanto, tendem a executar suas VMs de maneira mais veloz. Nota-se também que quanto maior os *rounds* do escalonamento para os métodos que visam o balanceamento da energia com o desempenho, o tempo de execução da simulação tende a aumentar.

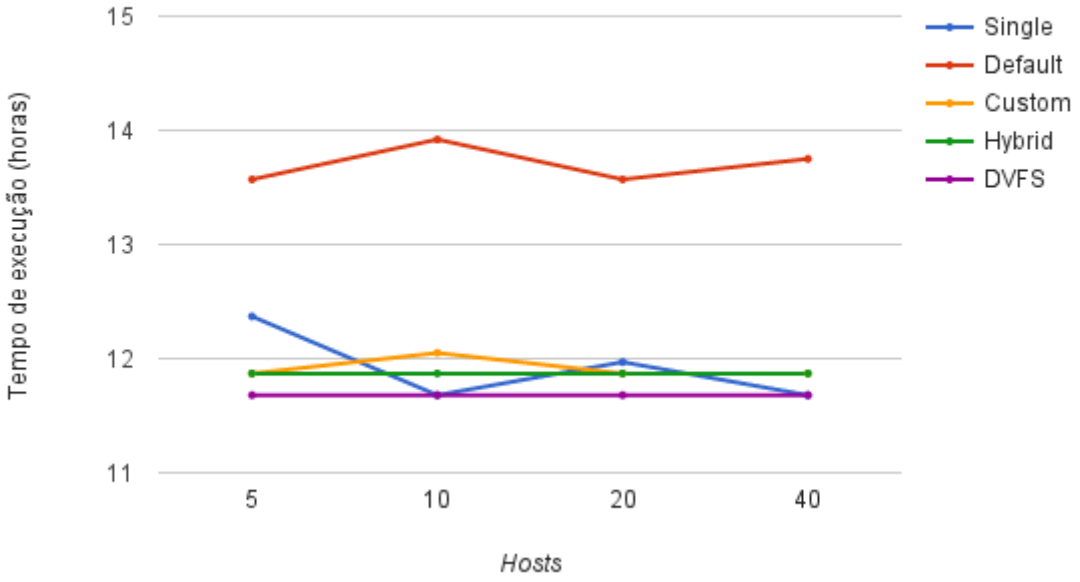


FIG. 5.7: Tempo de execução – Escalonamento com *round* de 00:01:00

Pode-se perceber também que mesmo em comparação com o método *Single*, que não trata a questão energética, o algoritmo desenvolvido mantém bom resultado. Já o algoritmo *Default* tem uma pequena perda em relação aos outros.

Em ambientes de nuvens, o desempenho é um fator que reflete diretamente no SLA e isso faz com que seja necessária uma atenção especial a esse fator. Esse fator é implementado pelo CloudSim em uma função de cálculo que está ligada aos recursos requisitados que foram atendidos e aos que não foram atendidos.

Nos gráficos das Figuras 5.13, 5.14, 5.15, 5.16, 5.17 e 5.18 estão evidenciados as menores médias de violação de SLA conseguidas pelo método desenvolvido neste trabalho. Esses bons resultados estão ligados ao fato de melhor direcionar as VMs de acordo com seu tipo de computação. Com a execução do método *Custom* os recursos requisitados são atendidos em maior quantidade que os demais métodos, ou seja, consegue evitar mais concorrência

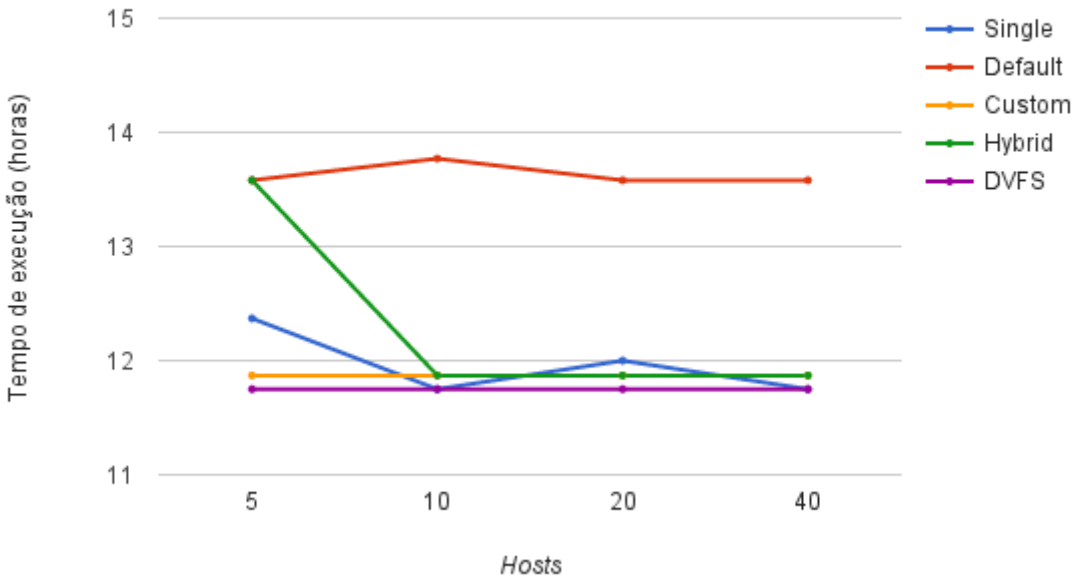


FIG. 5.8: Tempo de execução – Escalonamento com *round* de 00:03:45

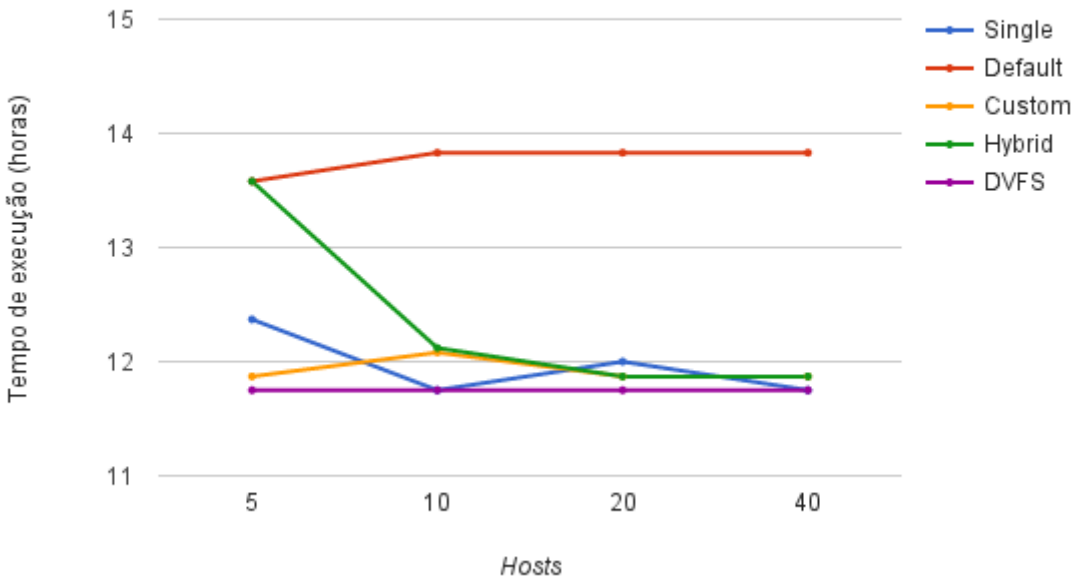


FIG. 5.9: Tempo de execução – Escalonamento com *round* de 00:07:30

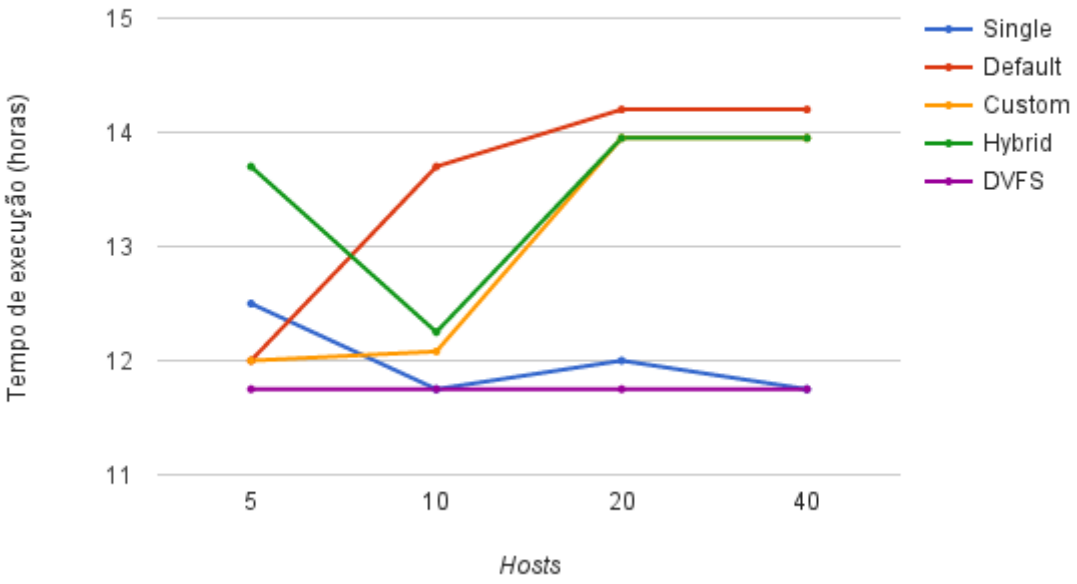


FIG. 5.10: Tempo de execução – Escalonamento com *round* de 00:15:00

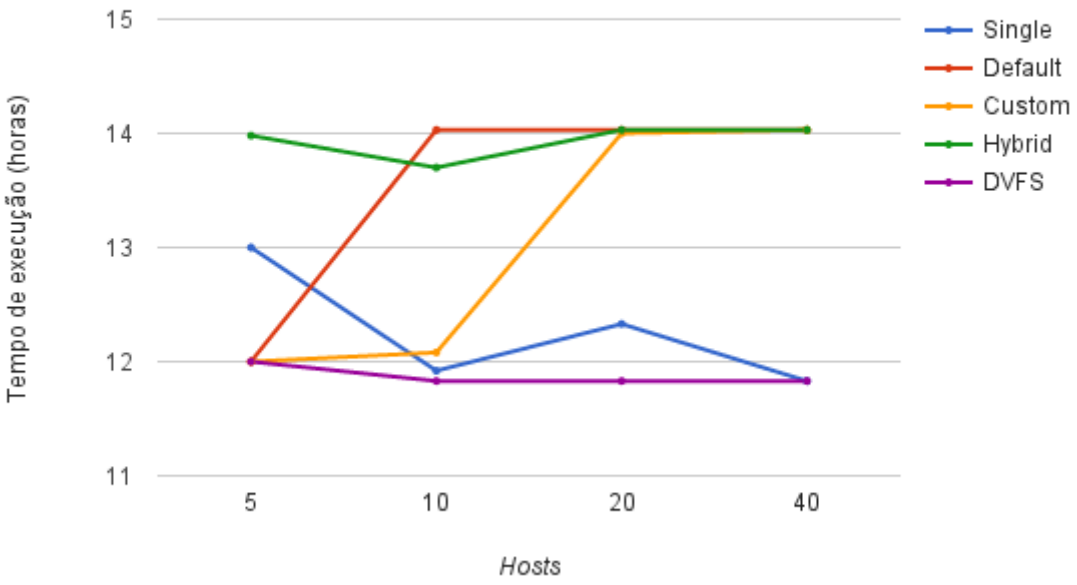


FIG. 5.11: Tempo de execução – Escalonamento com *round* de 00:30:00

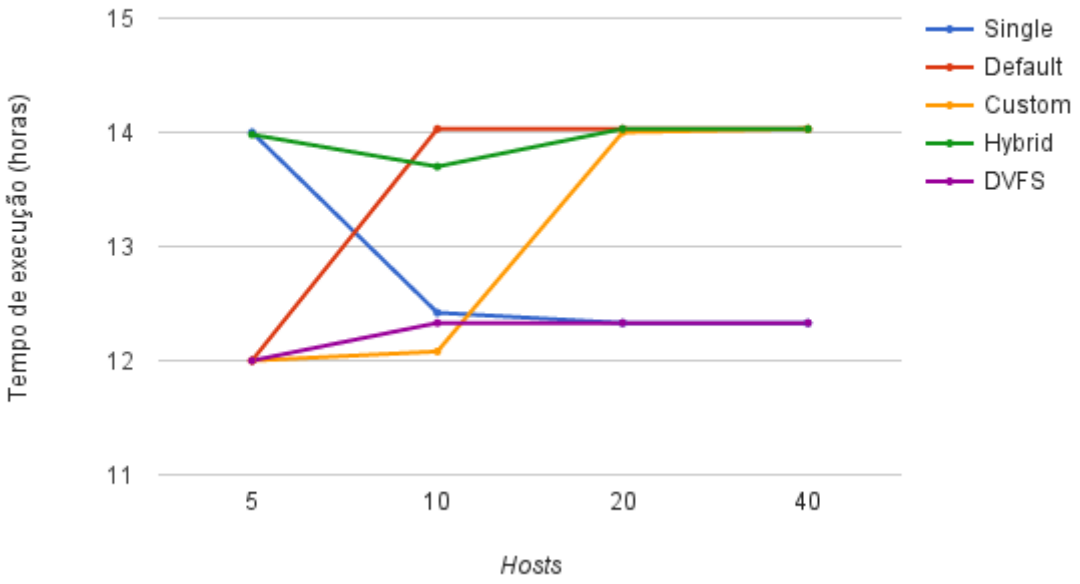


FIG. 5.12: Tempo de execução – Escalonamento com *round* de 01:00:00

pelos mesmos recursos, assim, impactando em menos degradações de desempenho. E esse é um dos objetivos do método implementado.

Já nos gráficos das Figuras 5.19, 5.20, 5.21, 5.22, 5.23 e 5.24 pode-se notar uma menor quantidade de violações por cada violação nos método de escalonamento desenvolvidos nesse trabalho. Ou seja, cada VM que não consegue obter todo a requisição por *round* tem uma média baixa dessas requisições que não são atendidas. Isso é reflexo da politica de alocação das VMs que trata a questão de aplicações CPU-*bound* e IO-*bound*. Como nesse método o mínimo de VM IO-*bound* pode executar concorrentemente com outras do mesmo tipo, em um mesmo servidor, não acontecem tantas degradações de performance e isso resulta em um número reduzido de violações de SLA.

Os outros três métodos (*Single*, *Default* e DVFS) não tratam esse questão de concorrência por recursos de IO, portanto as violações se mostram maiores.

Os métodos implementados nesse trabalho também executam menos migrações de VMs fazendo com que menos tempo se perca com transferências de VMs entre servidores. Isso impacta tanto no desempenho quanto no consumo energético do ambiente.

Nos gráficos das Figuras 5.25, 5.26, 5.27, 5.28, 5.29 e 5.30 pode-se observar a diferença das migrações ocorridas nos métodos *Default*, *Custom* e *Hybrid*. Os outros métodos, por

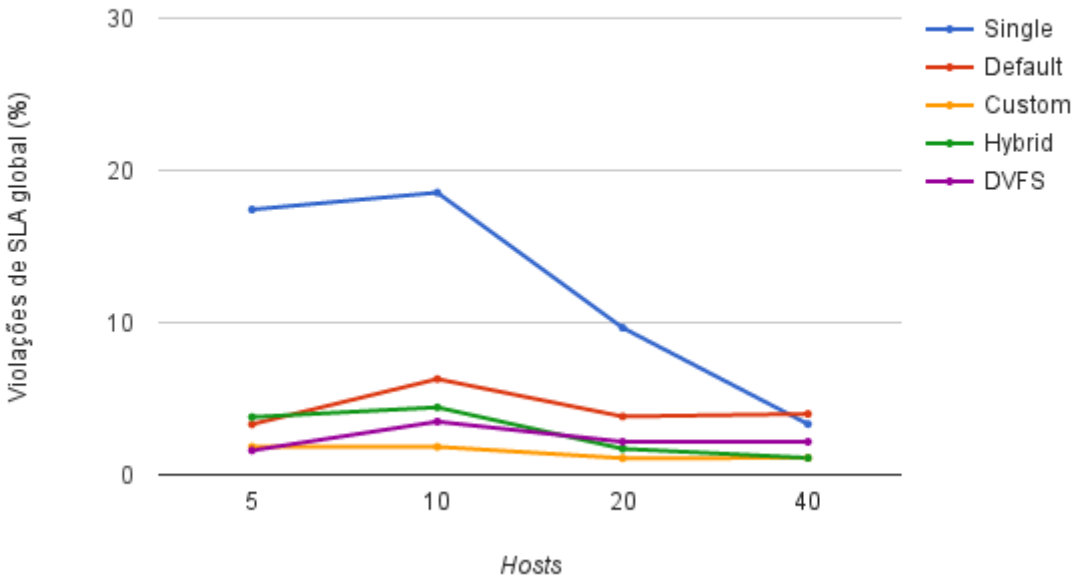


FIG. 5.13: Violação de SLA global – Escalonamento com *round* de 00:01:00

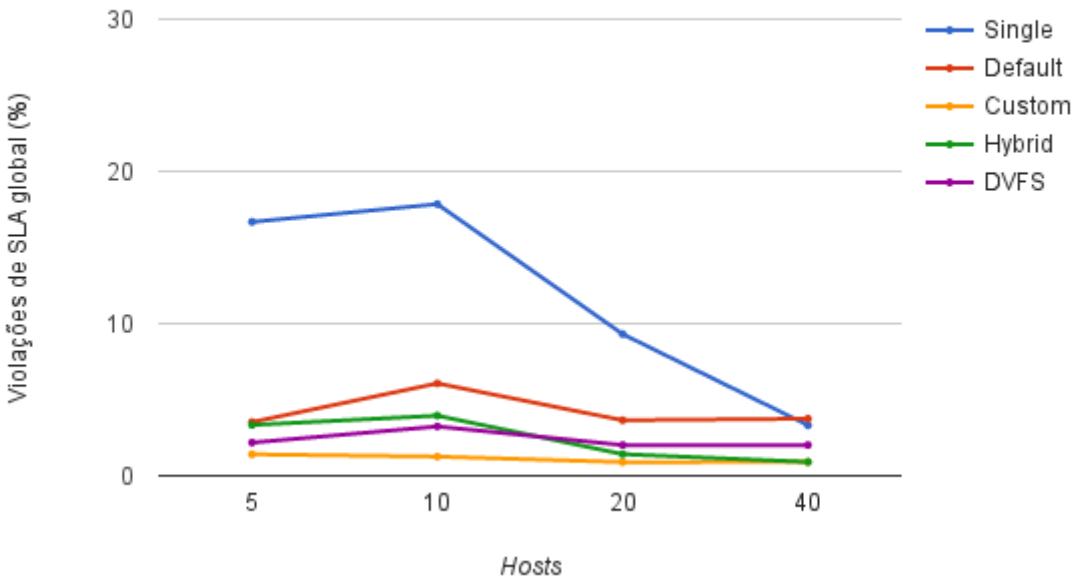


FIG. 5.14: Violação de SLA global – Escalonamento com *round* de 00:03:45

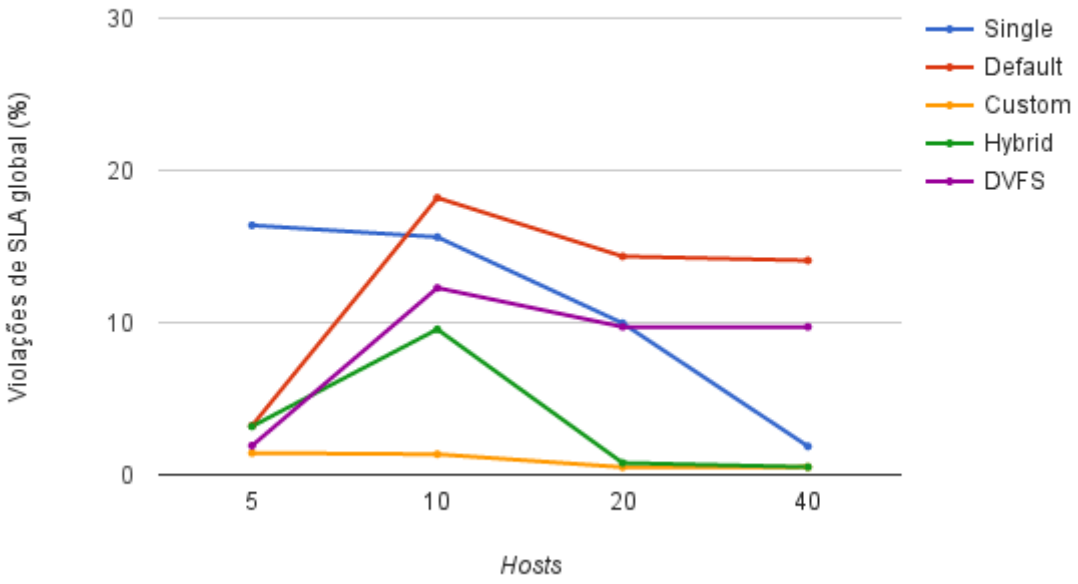


FIG. 5.15: Violação de SLA global – Escalonamento com *round* de 00:07:30

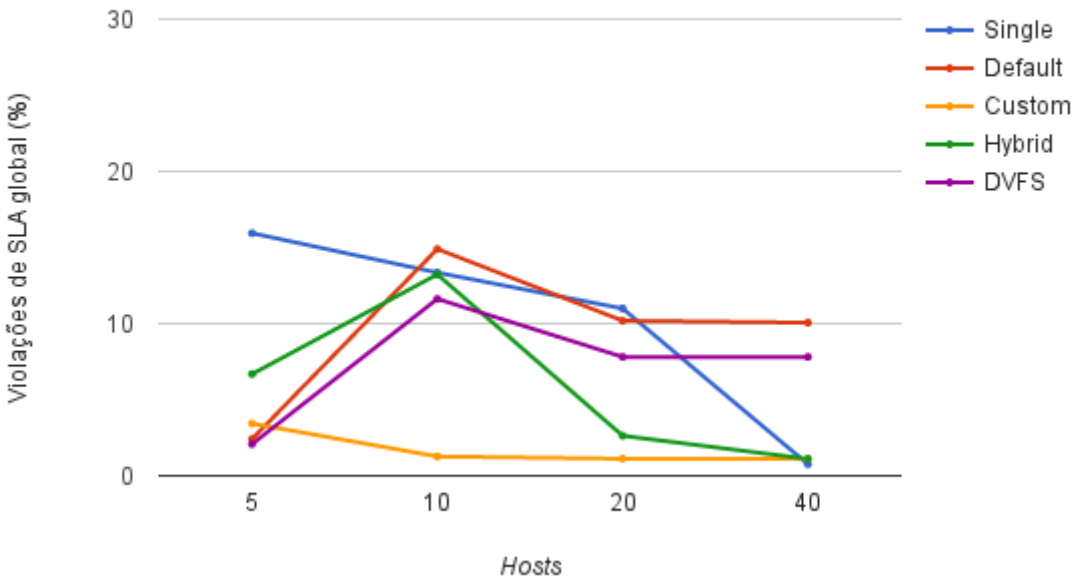


FIG. 5.16: Violação de SLA global – Escalonamento com *round* de 00:15:00

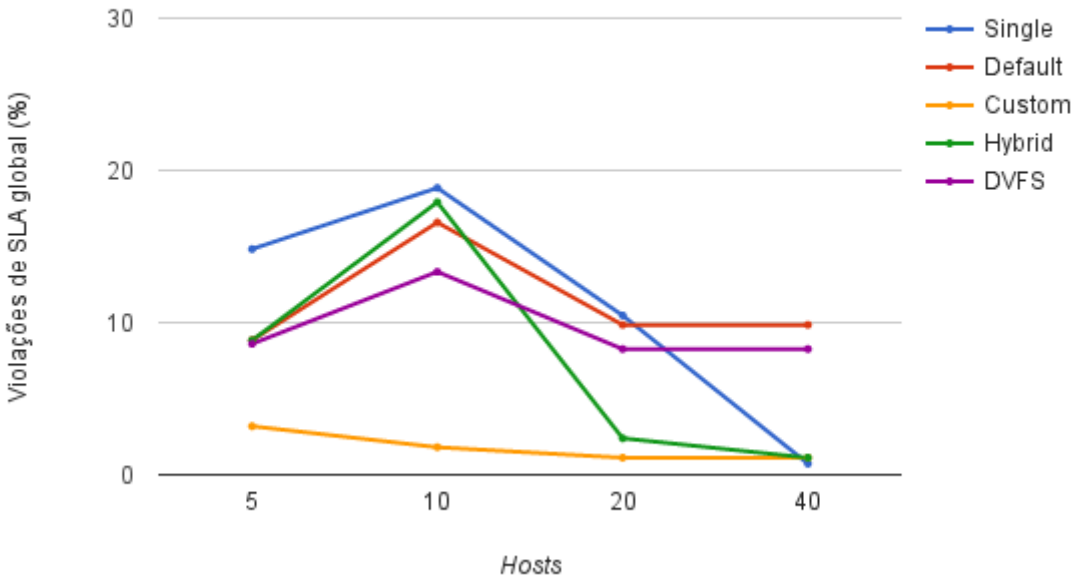


FIG. 5.17: Violação de SLA global – Escalonamento com *round* de 00:30:00

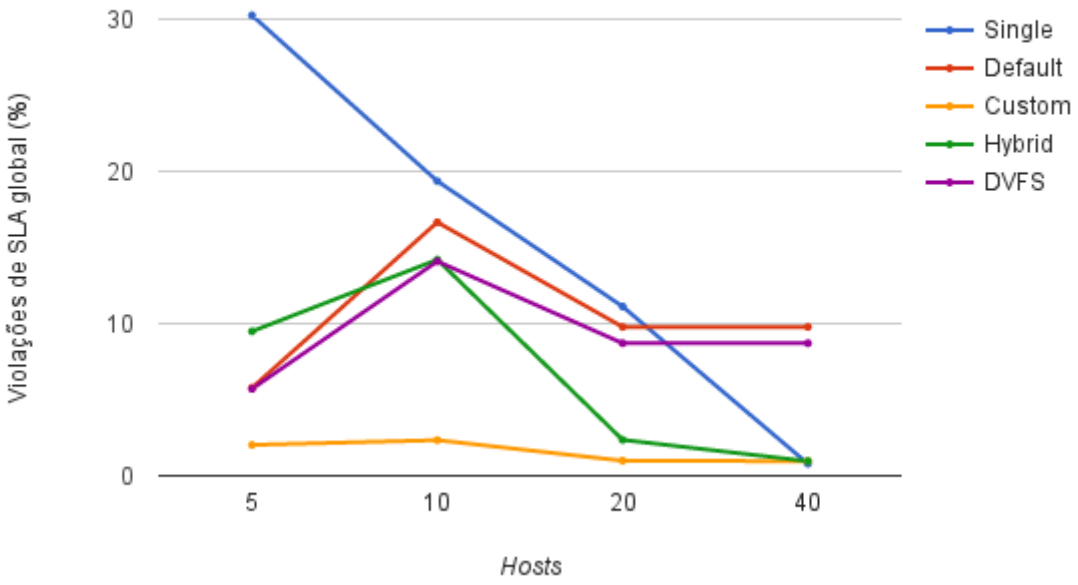


FIG. 5.18: Violação de SLA global – Escalonamento com *round* de 01:00:00

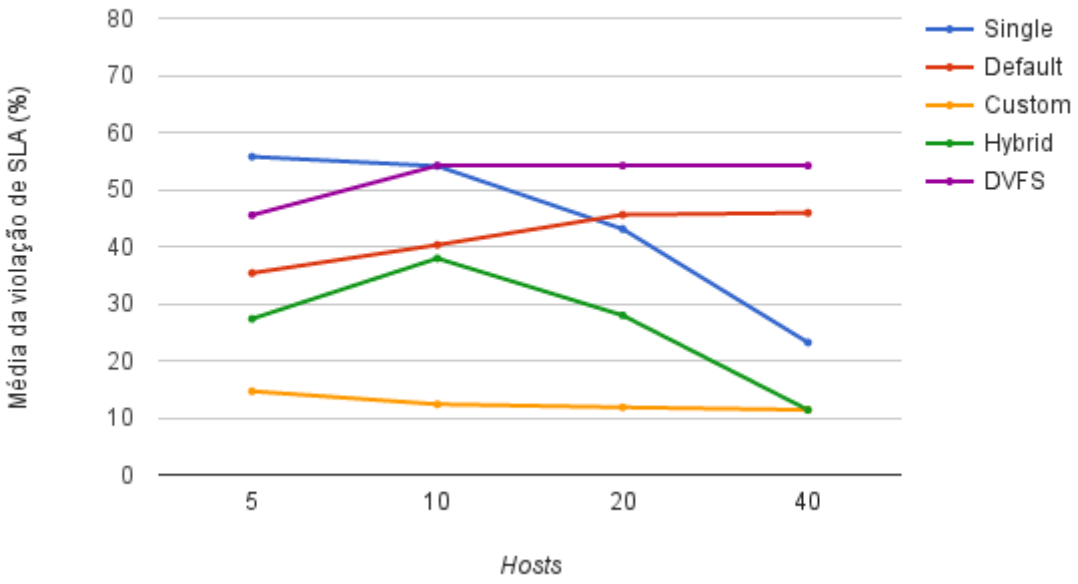


FIG. 5.19: Média das violações SLA – Escalonamento com *round* de 00:01:00

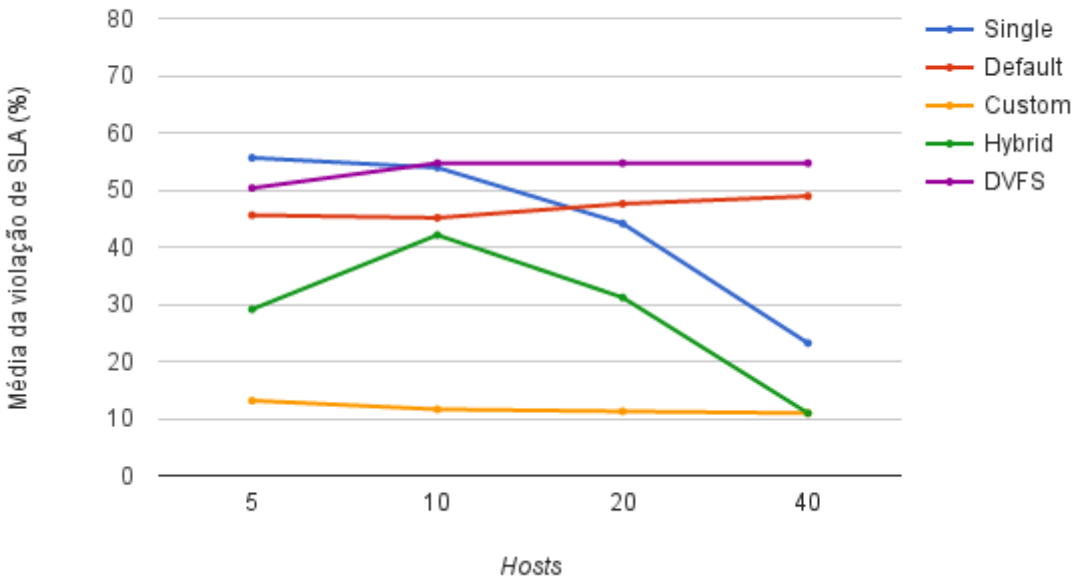


FIG. 5.20: Média das violações SLA – Escalonamento com *round* de 00:03:45

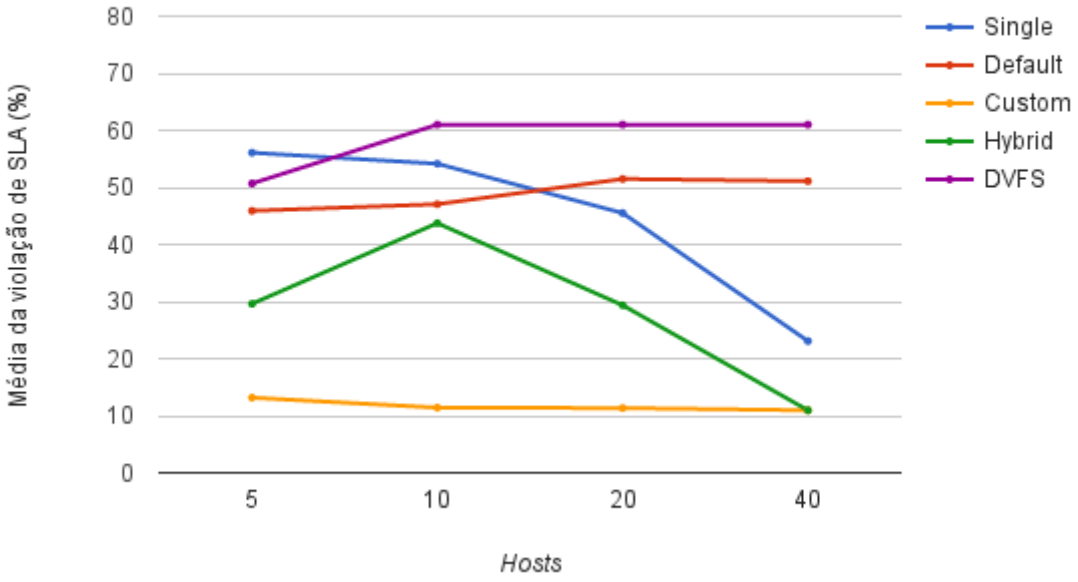


FIG. 5.21: Média das violações SLA – Escalonamento com *round* de 00:07:30

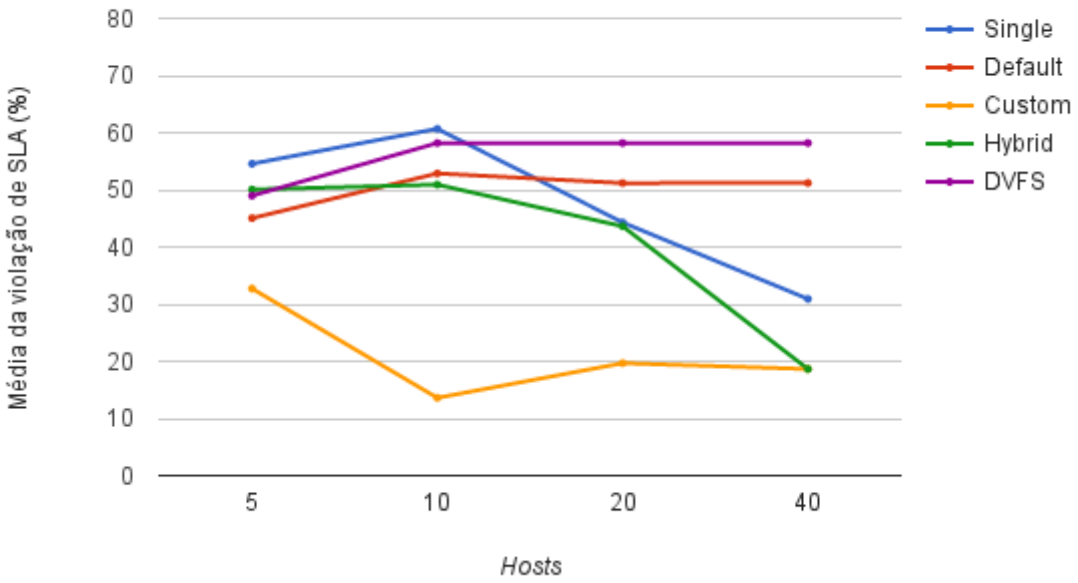


FIG. 5.22: Média das violações SLA – Escalonamento com *round* de 00:15:00

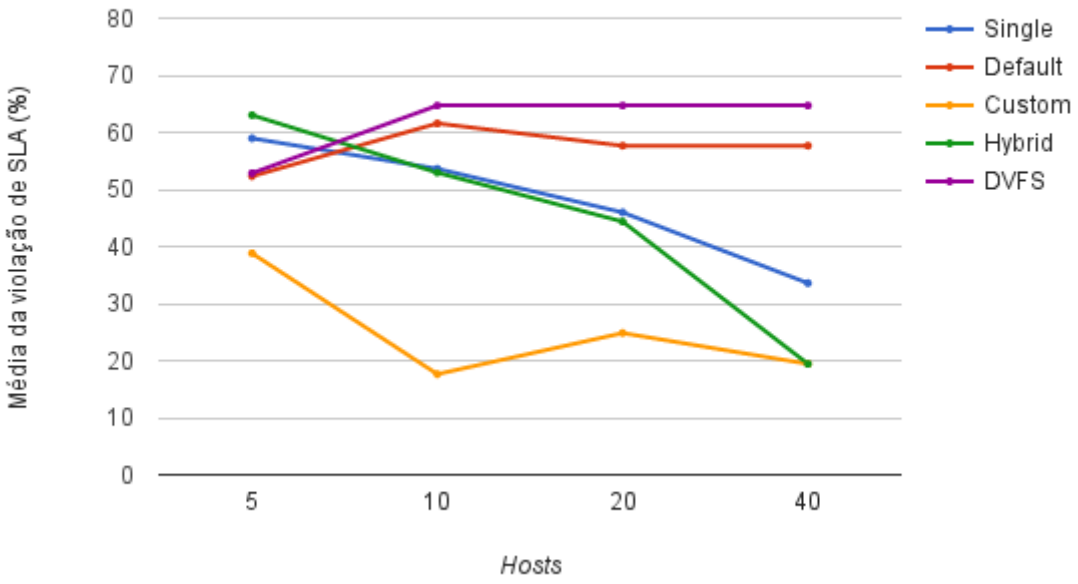


FIG. 5.23: Média das violações SLA – Escalonamento com *round* de 00:30:00

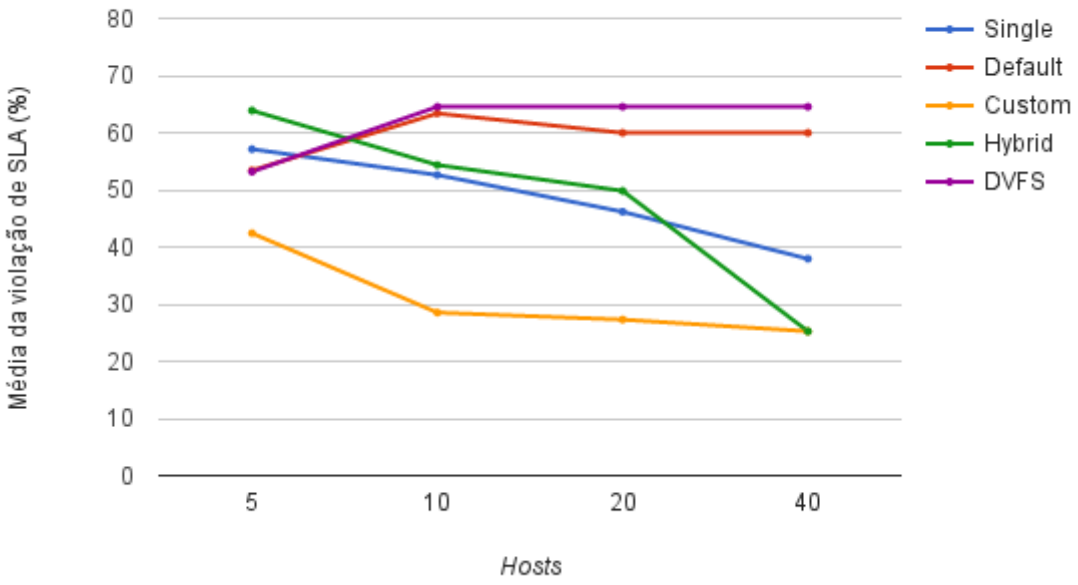


FIG. 5.24: Média das violações SLA – Escalonamento com *round* de 01:00:00

não executar migrações aparecem no eixo x para todas as variações de quantidade de *hosts*.

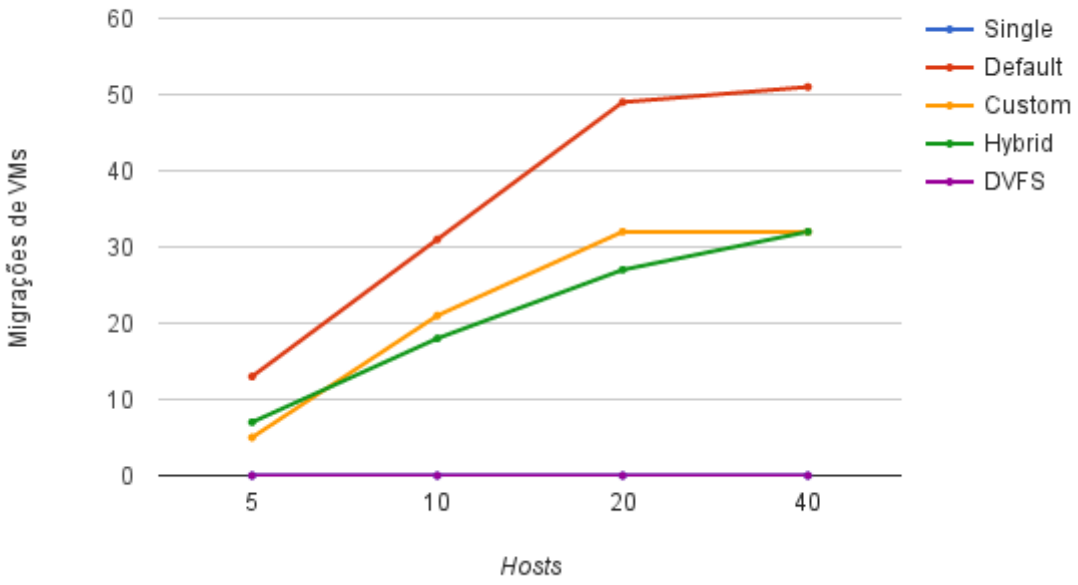


FIG. 5.25: Migrações de VMs – Escalonamento com *round* de 00:01:00

Quanto a questão da vazão de VMs pode-se observar no gráfico da Figura 5.31, que o método desenvolvido nesse trabalho (*Custom*) não consegue atingir a mesma quantidade de VMs executadas que o método *Default*. Entretanto, o método *Hybrid*, por combinar as duas estratégias, consegue.

Apesar do método *Hybrid*, não permitir que VMs IO-*bound* sejam realocadas no mesmo servidor, essa condição só é excedida na alocação inicial, quando as VMs são alocadas da mesma forma que no método *Default*. Já o método *Single* consegue alocar um pouco mais de VMs com menos recursos pois não trata a questão da relação desempenho-consumo como os outros dois métodos. Sua alocação visa executar o máximo de VMs o quanto possível, sem critério adicional.

Por sua estratégia de alocação ser mais restritiva, o método *Custom* é o que obteve o pior resultado nesse caso, mas esses dados contribuem para a avaliação, estudo e pesquisa de melhorias, visando o aprimoramento da ideia.

Nos gráfios das Figuras 5.32 e 5.33, 5.34 e 5.35, 5.36 e 5.37, 5.38 e 5.39 pode-se observar a comparação entre os melhores resultados de cada estratégia de escalonamento

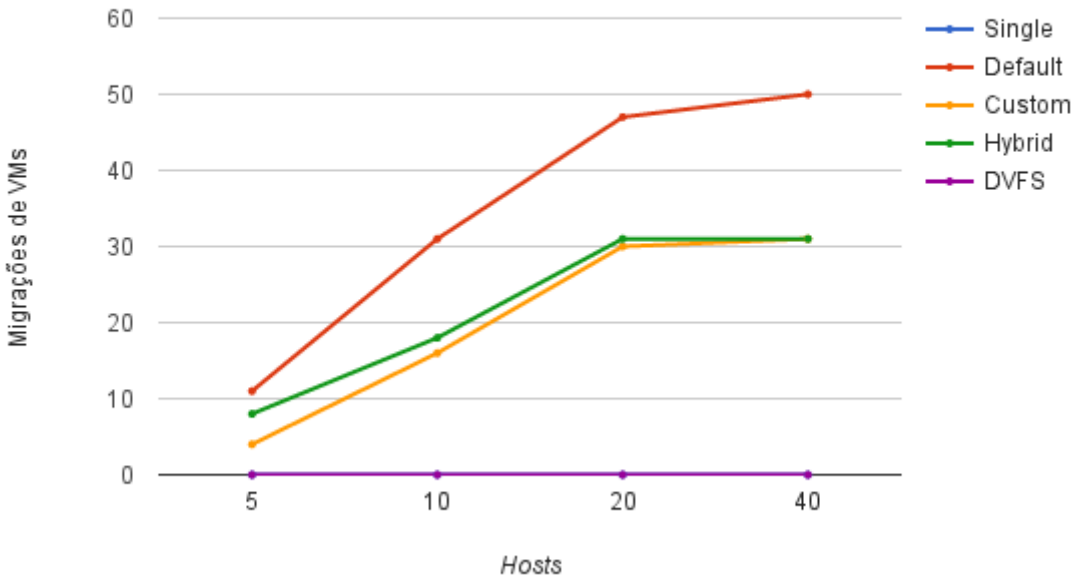


FIG. 5.26: Migrações de VMs – Escalonamento com *round* de 00:03:45

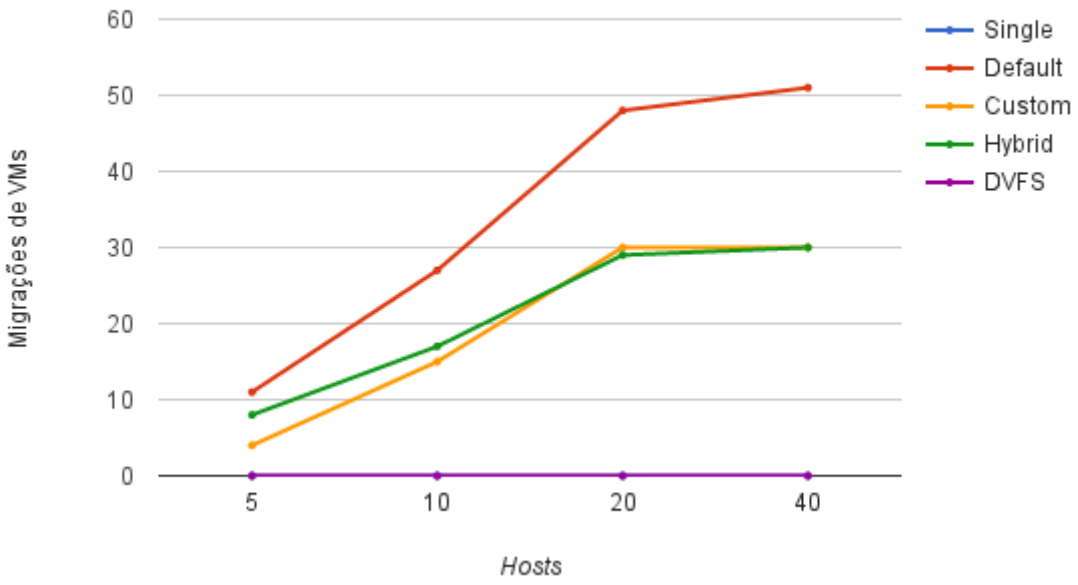


FIG. 5.27: Migrações de VMs – Escalonamento com *round* de 00:07:30

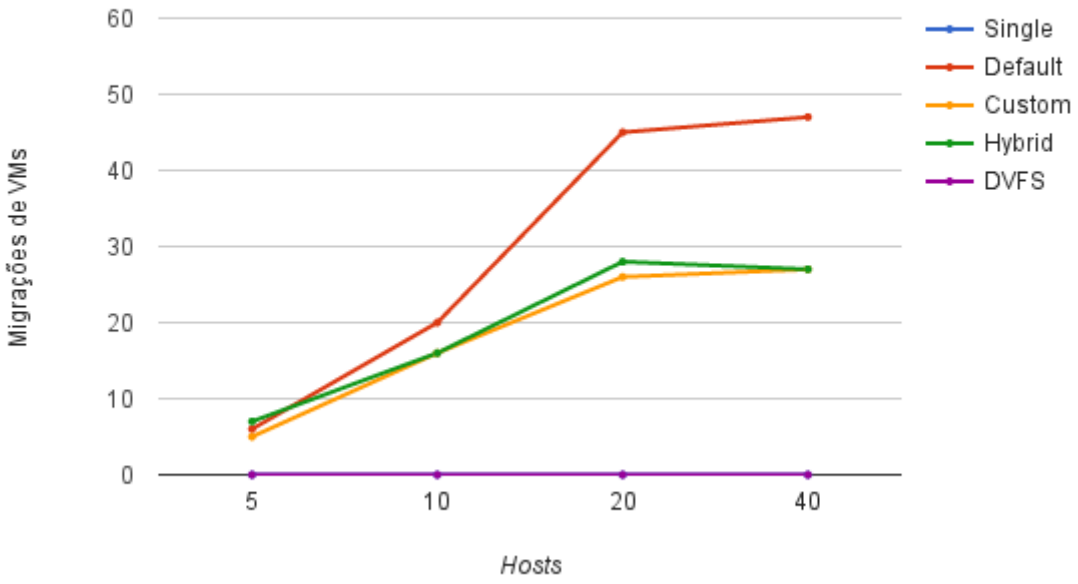


FIG. 5.28: Migrações de VMs – Escalonamento com *round* de 00:15:00

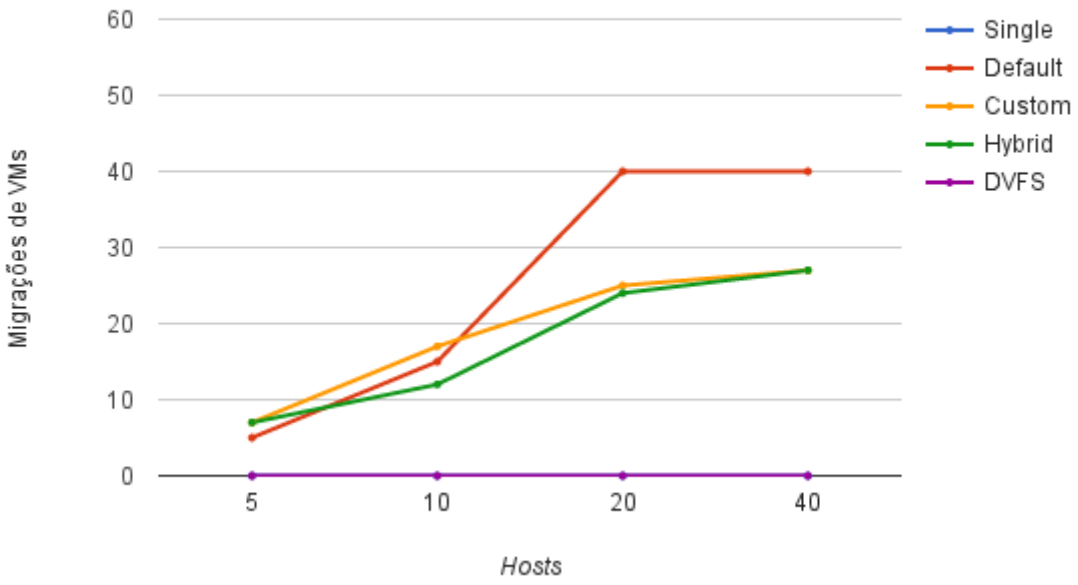


FIG. 5.29: Migrações de VMs – Escalonamento com *round* de 00:30:00

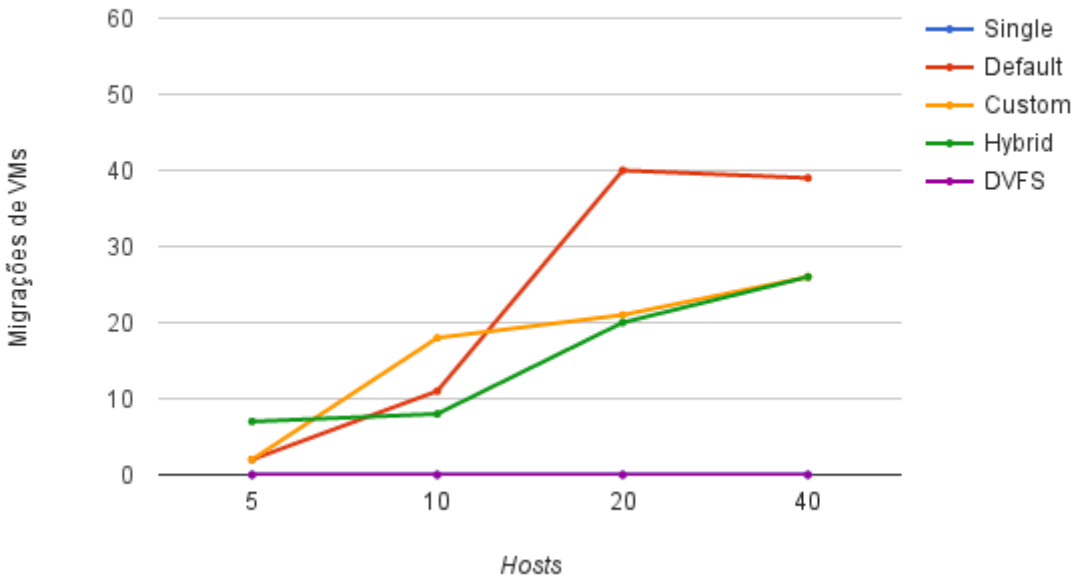


FIG. 5.30: Migrações de VMs – Escalonamento com *round* de 01:00:00

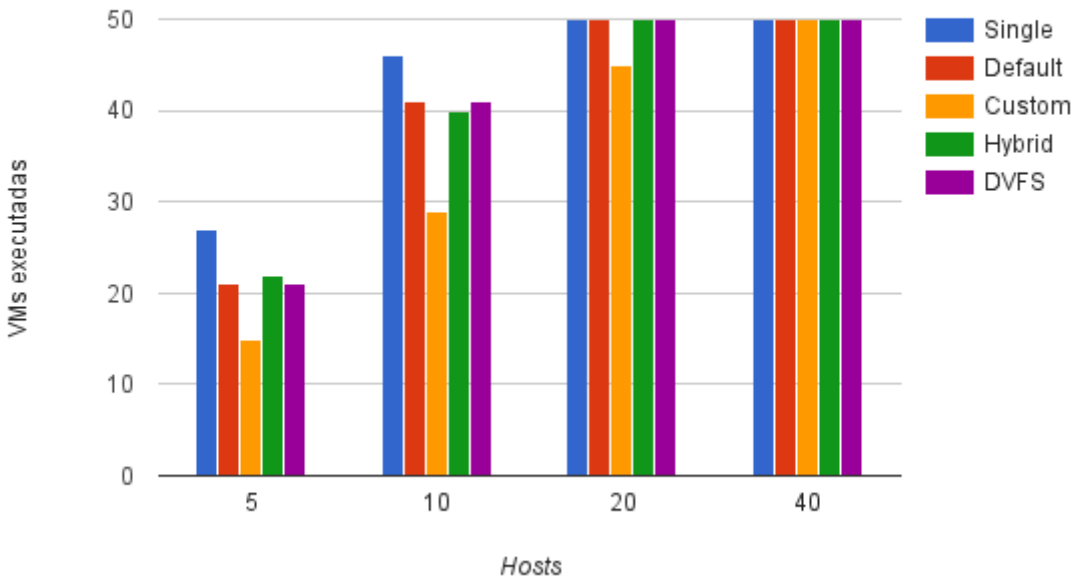


FIG. 5.31: Vazão

relacionada nesse trabalho.

Nos gráficos de bolhas, as bolhas que se aproximam mais do eixo mínimo de x e y significam que obtiveram a melhor relação entre execução e consumo. Os valores dentro das bolhas são informações sobre o *round* de escalonamento, ou seja, como foram escolhidos os melhores resultados entre cada estratégia, cada uma delas obteve esses melhores resultados com configurações de *round* distintas umas das outras.

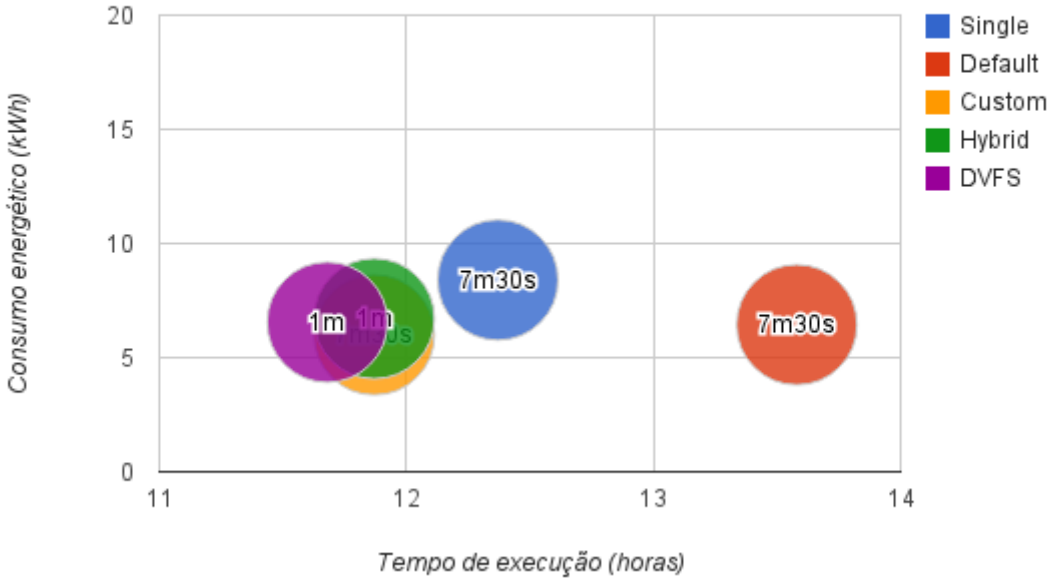


FIG. 5.32: Desempenho por consumo energético com 5 *hosts*

Sendo assim, esses resultados abrem possibilidades para a continuação e aprimoramento do método apresentado. O tempo para desenvolver mais modificações importantes no CloudSim foi pouco, mas o mínimo necessário foi executado, podendo assim destacar o tratamento na estratégia do método quando as diferenças dos tipos de tarefas. O CloudSim não implementa a representação dos componentes de IO quanto a execução de processos, portanto a execução destes necessitava ser representada. Dessa forma, foi criado um atributo chamado de "Tipo de Intensidade" e uma rotina que indicasse e causasse uma penalização nos recursos alocados quando esses recursos eram submetidos a atender as requisições das VMs no ambiente.

Devido ao fato de haver concorrência desses recursos quando VMs de mesmo tipo entravam em execução em um mesmo recurso, degradações de performance passaram a

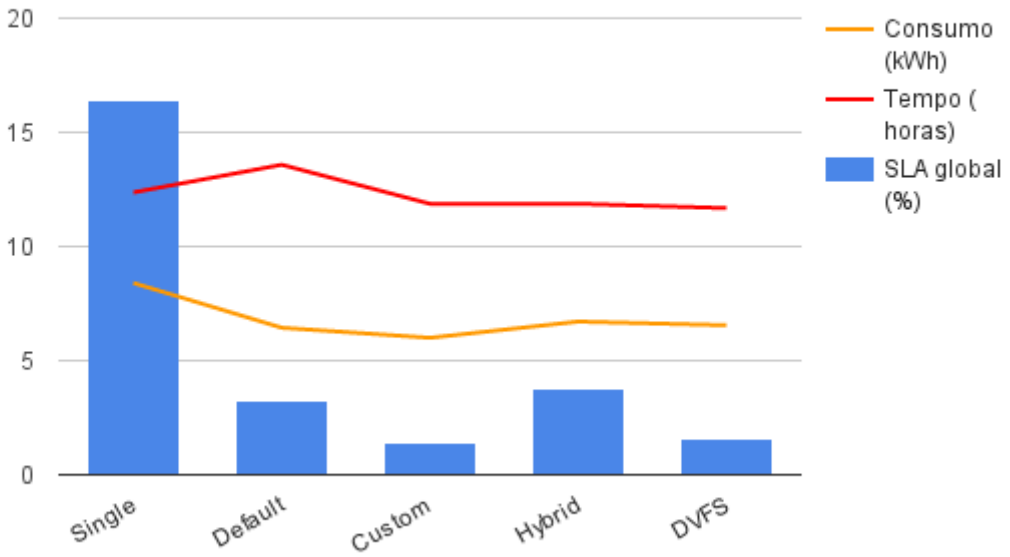


FIG. 5.33: Consumo, desempenho e violação de SLA global com 5 *hosts*

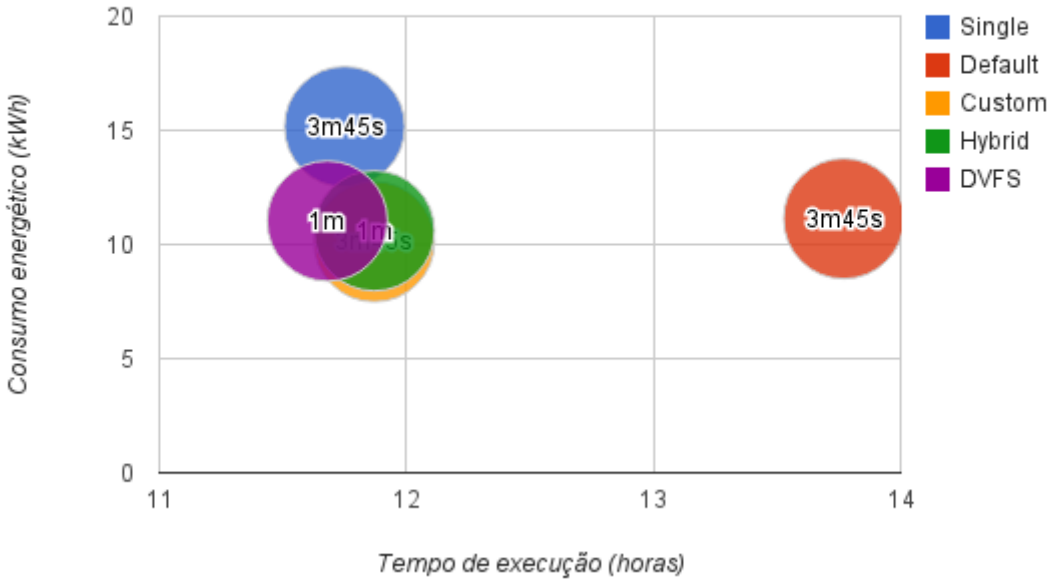


FIG. 5.34: Desempenho por consumo energético com 10 *hosts*

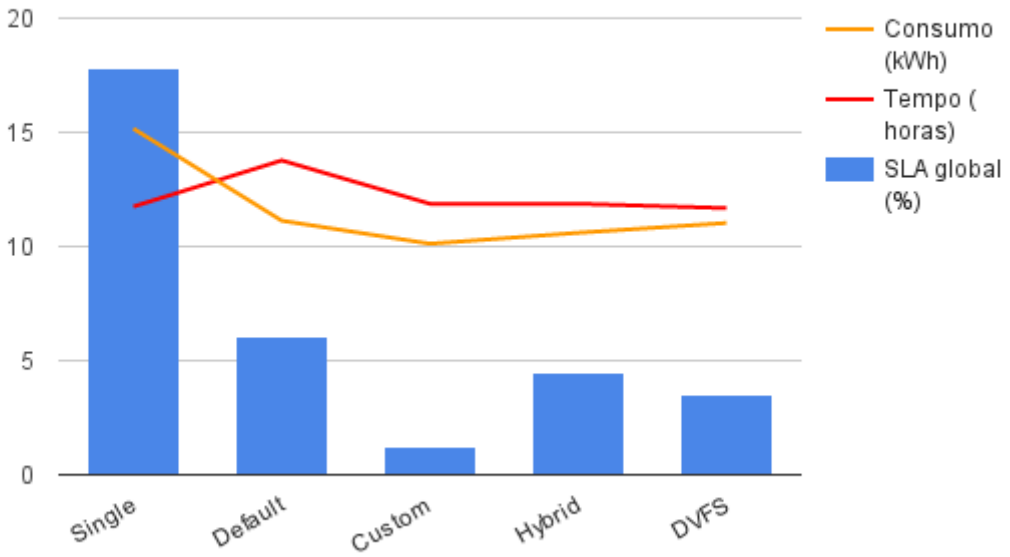


FIG. 5.35: Consumo, desempenho e violação de SLA global com 10 *hosts*

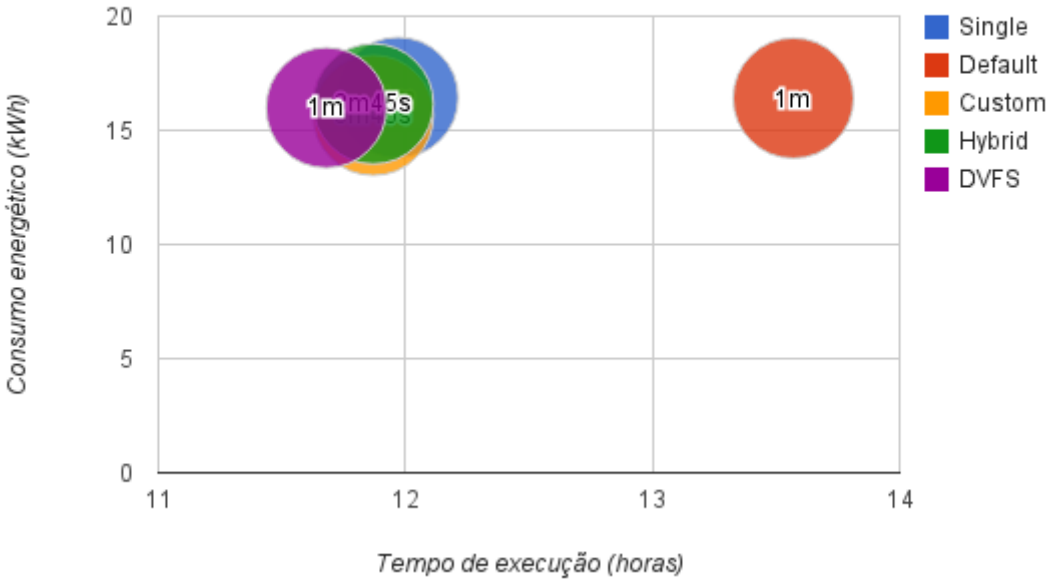


FIG. 5.36: Desempenho por consumo energético com 20 *hosts*

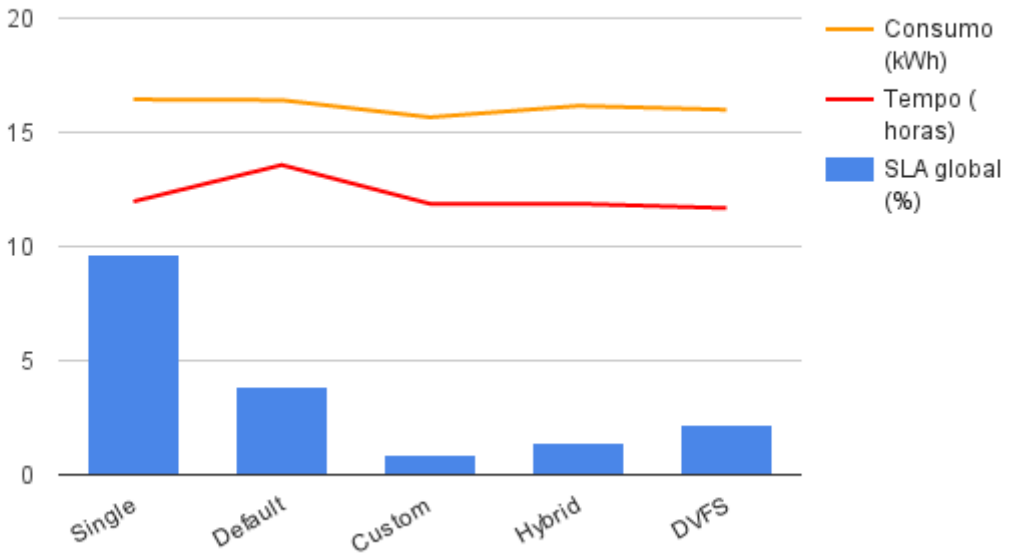


FIG. 5.37: Consumo, desempenho e violação de SLA global com 20 *hosts*

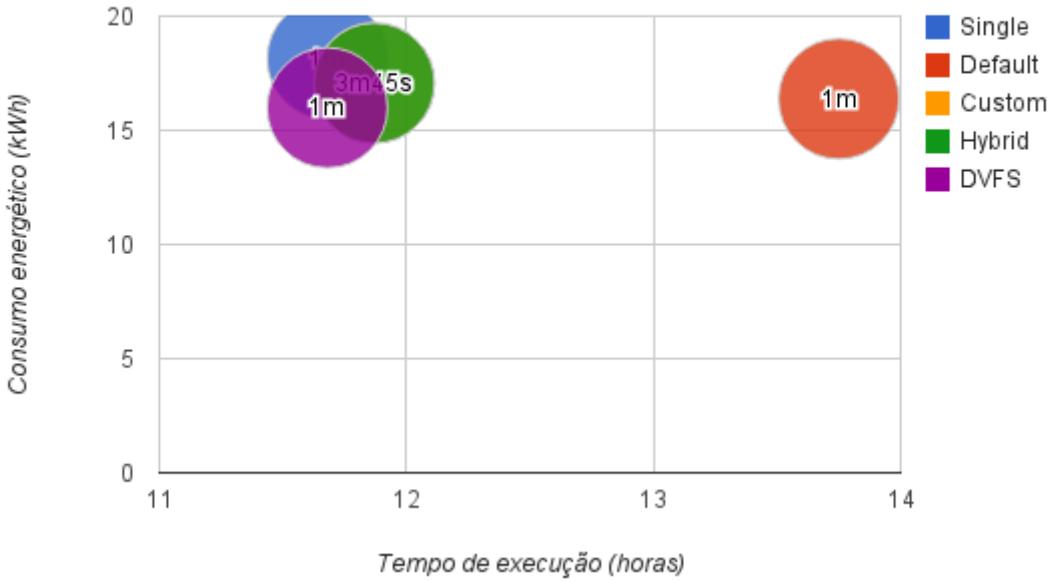


FIG. 5.38: Desempenho por consumo energético com 40 *hosts*

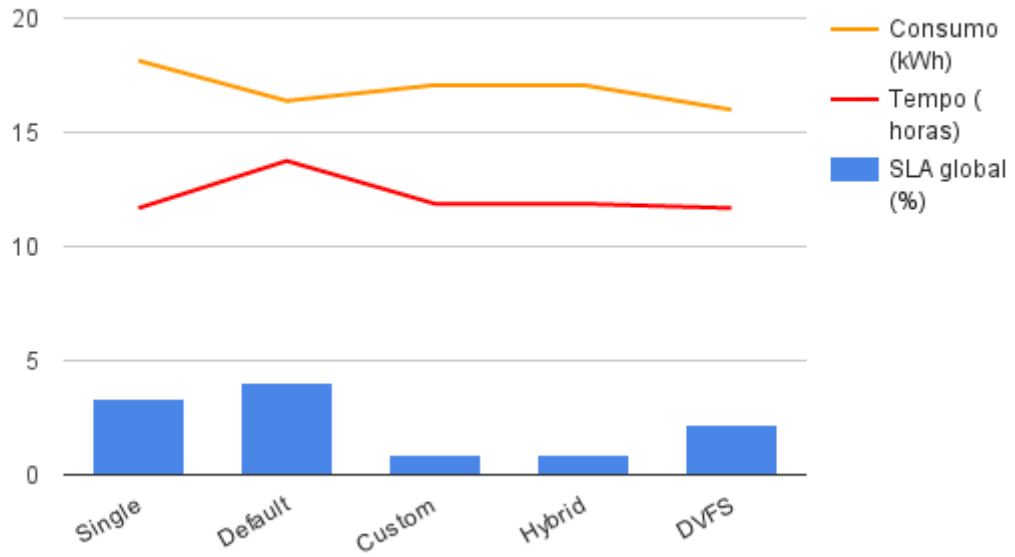


FIG. 5.39: Consumo, desempenho e violação de SLA global com 40 *hosts*

ocorrer e assim foi possível diferenciar as VMs e tratar suas alocações como no algoritmo desenvolvido em Java Threads. O impacto disso foi que após essa modificação simples, os métodos já implementados pelo CloudSim e que não tratavam essa diferença de aplicações, tiveram perdas quando ao desempenho nas simulações e causaram penalidades ao ambiente.

As estratégias de escalonamento, até aqui desenvolvidas, *Custom* e *Hybrid*, como pode-se observar, atingiram bons resultados apesar da vazão da estratégia *Custom* ser a pior quando poucos servidores estão disponíveis para atender a demanda de VMs. Ainda sim, as duas estratégias conseguem obter ótimos valores quanto as violações de SLA se comparadas com as outras três estratégias relacionadas. Como esses resultados foram obtidos por meio do CloudSim alterado, acredita-se que um trabalho mais apurado no simulador, adaptando-o melhor a diferenciar e calcular as características das aplicações quanto ao uso dos recursos de *hardware*, pode provocar ainda algumas alterações nos resultados.

É também válido destacar que por se tratar de uma primeira versão dessas estratégias de escalonamento, e pelas mesmas partirem de uma ideia simples de diferenciar características computacionais para melhor alocar VMs, o método ainda pode evoluir.

6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Nesse trabalho foi desenvolvido e apresentado um método de escalonamento de VMs para ambientes de nuvens com o objetivo de balancear consumo energético e desempenho, melhorando a qualidade de serviço (QoS). Considerando esses tipos de ambientes, a execução de dois tipos de aplicações com características computacionalmente distintas quanto ao uso do *hardware* foi a ideia central para a criação do método de escalonamento. Essa diferença se deu no âmbito de aplicações intensivas de CPU e de IO. Também foi apresentado todo o seu processo de desenvolvimento, que iniciou baseado em dados de consumo coletados experimentalmente, passando pela análise dos mesmos, posteriormente a construção de um novo método, e finalizando, até esse momento, com simulações para validação e resultados.

Durante o trabalho, foram observados e comprovados algumas características que viabilizaram a ideia do método de escalonamento. A de mais destaque é a possibilidade de extrapolar a capacidade do servidor quanto ao número de processadores reais em relação aos processadores das VMs que nele executam, sendo tal fato possível desde que suas características de execução sejam favoráveis, ou seja, VMs CPU-*bound* dependem de processadores livres e não devem ser atrapalhadas, ao passo que VMs IO-*bound* podem executar em processadores ocupados já que não ocupam tempo de CPU significativo, entretanto não podem concorrer com outras VMs IO-*bound*, visto que concorrem pelos componentes de IO. Dessa forma é possível arrumar as VMs em servidores de forma a garantir uma melhor QoS, contribuindo com o SLA e por sua vez com o desempenho dos ambientes virtuais e com a energia. Também é possível economizar com o ligamento de servidores adicionais e reduzir, com isso, a necessidade de migração das VMs, contribuindo com um melhor aproveitamento dos recursos computacionais em relação a demanda.

Pode-se destacar que qualquer esforço de desenvolvimento de escalonadores precisa de uma avaliação prévia no ambiente em que se deseja executar. Aferir suas diferenças e correlacionar os dados dessas medições se mostra de extrema importância para se extrair o melhor desempenho das aplicações e também a melhor estratégia de escalonamento a ser executada no ambiente em questão. Dessa forma, um bom resultado pode ser conseguido devido ao fato de que particularidades sempre serão encontradas e se forem tratadas, os

problemas serão minimizados, conseguindo-se então melhores resultados. A observação do problema foi uma etapa importante no desenvolvimento desse trabalho, a partir dela e das características dos servidores foi desenvolvido o método.

Quanto a implementação do método no presente momento, o mesmo já consegue obter resultados energéticos similares ao método de economia de energia implementado no CloudSim, ainda que seja uma versão inicial de resolução do problema. Para melhorar isso, o CloudSim precisa ser melhor modificado a ponto de trabalhar com os requisitos dos processamentos de IO. Até esse momento, isso não pode ser completamente implementado no simulador pois exige modificações em muitas partes do mesmo. O que se fez, então, foi apenas diferenciar as VMs com seus tipos de aplicações a restringir seus servidores de execução de acordo com as regras do método, buscando a minimizar as degradações de desempenho causadas pelas VMs IO-*bound*.

Outro ponto importante é que esse método tratou a questão do tempo de execução das VMs, tornando-as melhor. Isso significa que as requisições por recursos de processamento, em geral, sofreram menos restrições e foram atendidas de maneira esperada, resultando também em menos violações de SLA, tornando a QoS mais alta.

Com isso, pode-se afirmar que o objetivo do trabalho foi atingido ao conseguir balancear desempenho e consumo de energia. Isso contribui para que clientes e provedores minimizem riscos e tenham disponíveis seus ambientes de execução de aplicações mais estáveis quanto aos resultados por eles esperados.

Pôde-se perceber que a questão das diferenças de tipos de aplicações, se tratadas, de fato contribuem com melhor aproveitamento dos recursos envolvidos. Assim, esforços na direção de desenvolvimento de algoritmos de escalonamento específicos, de acordo com o tipo de padrão computacional a ser executado em um ambiente virtualizado, como o de uma nuvem computacional, traz economias para provedores e consumidores.

Em trabalhos futuros, uma ideia geral de aproveitar o legado da pesquisa desenvolvida até esse momento e aperfeiçoar o método de escalonamento de forma que sua política de alocação consiga, dinamicamente, escalonar VMs de acordo com a utilização dos componentes de *hardware* no instante de suas execuções.

No presente trabalho essa alocação é estabelecida por meio da consideração de que aplicações são 100% CPU-*bound* ou 100% IO-*bound*, entretanto sabe-se que além dessas aplicações existem outras que, em fases de suas execuções, atuam como CPU-*bound* em uns momentos e IO-*bound* em outros. Isso poderia trazer melhor rendimento ao ambiente

computacional, contribuindo ainda com a questão energética e o desempenho do mesmo. Identificar as características do *datacenter* virtualizado quanto ao tipo das aplicações que nele executarão e, com isso, especializar o método de escalonamento e suas políticas de alocação de VMs considerando isso, pode-se obter mais ganhos. Se um ambiente possui mais de uma especificação quanto as aplicações executadas, como por exemplo esse trabalho ao executar aplicações HPC do tipo CPU-*bound* e do tipo IO-*bound*, então, melhorar o método aplicado ao ambiente reflete em melhores resultados para os envolvidos com o mesmo.

Na intenção de aprimorar mais a política de alocação das VMs do método aqui apresentado, pretende-se aplicar uma variação não só da carga como também da distribuição dos ambientes virtualizados em mais de um servidor, gerando assim mais informações reais a serem consideradas no ajuste do método de escalonamento.

Até o presente momento, poucas variações nas simulações foram executadas, logo são necessárias mais delas para identificar as melhores aplicações do trabalho. É preciso, por exemplo, variar a quantidade de VMs, a quantidade relativa de aplicações CPU-*bound* e IO-*bound*, bem como tamanho das tarefas para complementar informações e identificar a tendência das políticas de alocação já medidas.

Para aprimorar as simulações na direção deste trabalho, uma atualização do CloudSim também é necessária. Essa atualização consiste na implementação de novas abstrações de componentes como os dispositivos de IO referentes as necessidades de consumo das aplicações. Na versão do simulador utilizado nesse trabalho, os objetos do CloudSim não contemplam a questão do processo de aplicações em dispositivos de IO, tratado somente a execução de processos em CPUs. Portanto a métrica de processamento em IO precisa ser implementada para detalhar melhor o método apresentado aqui. Com essa implementação também será possível aplicar um modelo de consumo pra cada um dos itens de processamento (CPU e IO), gerando variação de consumo energético, desempenho e violações de SLA. Com essa atualização, o trabalho pode gerar melhores resultados além de novos dados referentes a esses componentes pelos fato de possibilitar melhor apuração dos dados das simulações.

7 REFERÊNCIAS BIBLIOGRÁFICAS

ALLES, F. **Caracterização do consumo energético do hadoop mapreduce**. Dissertação de Mestrado, Universidade Federal do Rio Grande do Sul - UFRGS, 2013. Acessado em 13/11/2014.

ALMEIDA, F., GIMÉNEZ, D., MANTAS, J. M. e VIDAL, A. M. *Introducción a la programación paralela*. Paraninfo Cengage Learning, 2008.

ARMBRUST, M., FOX, A., GRIFFITH, R., JOSEPH, A. D., KATZ, R. H., KONWINSKI, A., LEE, G., PATTERSON, D. A., RABKIN, A., STOICA, I. e ZAHARIA, M. **Above the clouds: A berkeley view of cloud computing**. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009. URL `http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html`. Acessado em 20/01/2014.

AZEVEDO, E., DIAS, C., SIMÕES, R., DANTAS, R., SADOK, D., FERNANDES, S. e KAMIENSKI, C. **Nuvem pública versus privada: Variações no desempenho de infraestrutura para elasticidade**. *Anais do WCGA*, 2012.

BACHIEGA, N. G. **Algoritmo de escalonamento de instância de máquina virtual na computação em nuvem**. Dissertação de Mestrado, Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista Júlio de Mesquita Filho, 2014.

BALIGA, J., AYRE, R., HINTON, K. e TUCKER, R. **Green cloud computing: Balancing energy in processing, storage, and transport**. *Proceedings of the IEEE*, 99(1):149–167, Jan 2011. ISSN 0018-9219.

BARBOSA, P. X. A., BARBOSA, J. M. G. e SAMPAIO, A. M. S. **Estratégias de consolidação de máquinas físicas em infraestruturas de cloud computing**. Dissertação de Mestrado, Faculdade de Engenharia da Universidade do Porto - FEUP, 2014.

BELOGLAZOV, A., ABAWAJY, J. e BUYYA, R. **Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing**. *Future Gener. Comput. Syst.*, 28(5):755–768, maio 2012. ISSN 0167-739X. URL `http://dx.doi.org/10.1016/j.future.2011.04.017`.

BELOGLAZOV, A. e BUYYA, R. **Energy efficient allocation of virtual machines in cloud data centers**. Em *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, págs. 577–578. IEEE, 2010.

BESERRA, D., BORBA, A., SOUTO, S., DE ANDRADE, M. e DE ARAÚJO, A. **Desempenho de ferramentas de virtualização na implementação de clusters beowulf virtualizados em hospedeiros windows**. Em *X Workshop em Clouds, Grids e Aplicações-SBRC 2012*, págs. 86–95, 2012.

BESERRA, D., KARMAN, R., CAMBOIM, K., ARAUJO, J., BORBA, A. e ARAÚJO, A. **Análise do desempenho de sistemas operacionais hospedeiros de clusters virtualizados com o virtualbox**. Em *Anais do XII Workshop de Computação em Clouds e Aplicações–WCGA*, 2014.

BIANZINO, A. P., CHAUDET, C., ROSSI, D. e ROUGIER, J. **A survey of green networking research**. *Communications Surveys & Tutorials, IEEE*, 14(1):3–20, 2012.

BUYYA, R., BELOGLAZOV, A. e 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.

BUYYA, R., YEO, C. S. e VENUGOPAL, S. **Market-oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities**. *CoRR*, abs/0808.3558, 2008. URL `http://arxiv.org/abs/0808.3558`.

CALHEIROS, R. N., RANJAN, R., BELOGLAZOV, A., DE ROSE, C. A. F. e BUYYA, R. **Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms**. *Softw. Pract. Exper.*, 41(1):23–50, janeiro 2011. ISSN 0038-0644. URL `http://dx.doi.org/10.1002/spe.995`.

CAMPOS, P. P. V. **Algoritmos de escalonamento para redução do consumo de energia em computação em nuvem**. Dissertação de Mestrado, Universidade de São Paulo, 2013.

CASAVANT, T. e KUHL, J. **A taxonomy of scheduling in general-purpose distributed computing systems**. *Software Engineering, IEEE Transactions on*, 14(2): 141–154, Feb 1988. ISSN 0098-5589.

CHEN, Y., GMACH, D., HYSER, C., WANG, Z., BASH, C., HOOVER, C. e SINGHAL, S. **Integrated management of application performance, power and cooling in data centers**. Em *Network Operations and Management Symposium (NOMS), 2010 IEEE*, págs. 615–622. IEEE, 2010.

CHENG, Y. e CHEN, W. **Evaluation of virtual machine performance on numa multicore systems**. Em *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference on*, págs. 136–143, Oct 2013.

CLARK, C., FRASER, K., HAND, S., HANSEN, J. G., JUL, E., LIMPACH, C., PRATT, I. e WARFIELD, A. **Live migration of virtual machines**. Em *Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, págs. 273–286, Berkeley, CA, USA, 2005. USENIX Association. URL `http://dl.acm.org/citation.cfm?id=1251203.1251223`.

CLUSTERING, A. **Act knowledge base**, 2014. URL `http://www.advancedclustering.com/act-kb/tune-hpl-dat-file/`. Acessado em 08/04/2014.

COMPUTING, T. C. e LABORATORY, D. S. **Cloudsim: A framework for modeling and simulation of cloud computing infrastructures and services**, 2010. URL http://www.cloudbus.org/cloudsim/. Acessado em 02/02/2015.

DANDAMUDI, S. **Performance implications of task routing and task scheduling strategies for multiprocessor systems**. Em *Proceedings of the First International Conference*, págs. 348–353. IEEE Computer Society, May 1994.

DOE, U. **Department of energy national laboratories and plants**, 2012. URL http://www.nrel.gov/docs/fy13osti/56143.pdf. Acessado em 01/07/2013.

DUSSO, P. **Iaee**, 2011. URL https://github.com/pmdusso/energyagent.

DUSSO, P. **A monitoring system for wattdb: An energy-proportional database cluster**. Dissertação de Mestrado, Universidade Federal do Rio Grande do Sul - UFRGS, 2012.

EL-REWINI, H. e ABD-EL-BARR, M. *Advanced Computer Architecture and Parallel Processing*. Wiley Series on Parallel and Distributed Computing. Wiley, 2005. ISBN 9780471478393. URL https://books.google.com.br/books?id=7JB-u6D5Q7kC.

ENDO, P. T., GONÇALVES, G. E., KELNER, J. e SADOK, D. **A survey on open-source cloud computing solutions**. Em *VIII Workshop em Clouds, Grids e Aplicações*, págs. 3–16, 2010.

FISCHBORN, M. e OTHERS. **Computação de alto desempenho aplicada à análise de dispositivos eletromagnéticos**. 2006.

GARG, S. K. e BUYYA, R. **Green cloud computing and environmental sustainability**. *Harnessing Green IT: Principles and Practices*, págs. 315–340, 2011.

GOIRI, Í., BEAUCHEA, R., LE, K., NGUYEN, T. D., HAQUE, M. E., GUITART, J., TORRES, J. e BIANCHINI, R. **Greenslot: scheduling energy consumption in green datacenters**. Em *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, pág. 20. ACM, 2011.

GOOGLE. **Google scholar**, 2004. URL http://scholar.google.com. Acessado em 30/06/2015.

GREENPEACE. **Cool it leaderboard**. Technical report, Greenpeace International, 2013. URL http://www.rackspace.com/knowledge_center/sites/default/files/whitepaper_pdf/MovingyourInfrastructuretothe Cloud-HowtoMaxi. Acessado em 24/03/2014.

HAQUE, M. E., LE, K., GOIRI, Í., BIANCHINI, R. e NGUYEN, T. D. **Providing green slas in high performance computing clouds**. Em *Green Computing Conference (IGCC), 2013 International*, págs. 1–11. IEEE, 2013.

HINES, M. R. e GOPALAN, K. **Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning**. Em *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, VEE '09, págs. 51–60, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-375-4. URL http://doi.acm.org/10.1145/1508293.1508301.

INTEL. **Intelligent platform management interface**, 2014. URL http://www.intel.com/content/www/us/en/servers/ipmi/ipmi-home.html. Acessado em 17/11/2014.

JOHNSON, E., GARRITY, P., YATES, T. e BROWN, R. **Performance of a virtual cluster in a general-purpose teaching laboratory**. Em *Cluster Computing (CLUSTER), 2011 IEEE International Conference on*, págs. 600–604, Sept 2011.

KARVE, A., KIMBREL, T., PACIFICI, G., SPREITZER, M., STEINDER, M., SVIRIDENKO, M. e TANTAWI, A. **Dynamic placement for clustered web applications**. Em *Proceedings of the 15th international conference on World Wide Web*, págs. 595–604. ACM, 2006.

KEPES, B. **Moving your infrastructure to the cloud: How to maximize benefts and avoid pitfalls**. 2010. URL http://www.rackspace.com/knowledge_center/sites/default/files/whitepaper_pdf/MovingyourInfrastructuretotheCloud-HowtoMaximizeBenefitsandAvoidPitfalls.pdf. Acessado em 24/03/2014.

LIMA, J. D. R. **Monitorando interface ipmi**, 2012. URL http://www.vivaolinux.com.br/artigo/Monitorando-interface-IPMI. Acessado em 17/11/2014.

LIU, F., TONG, J., MAO, J., BOHN, R., MESSINA, J., BADGER, L. e LEAF, D. *NIST Cloud Computing Reference Architecture: Recommendations of the National Institute of Standards and Technology (Special Publication 500-292)*. CreateSpace Independent Publishing Platform, USA, 2012. ISBN 1478168021, 9781478168027.

MELL, P. M. e GRANCE, T. **Sp 800-145. the nist definition of cloud computing**. Technical report, Gaithersburg, MD, United States, 2011.

METKAR, G., AGRAWAL, S. e SINGH, S. **A live migration of virtual machine based on the dynamic threshold at cloud data centres**. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(10):401–405, 2013. ISSN 2277 128X. URL http://www.ijarcsse.com.

METSCH, T., EDMONDS, A., NYRÉN, R. e PAPASPYROU, A. **Open cloud computing interface–core**. Em *Open Grid Forum, OCCI-WG, Specification Document. Available at: http://ogf.org/documents/GFD.183.pdf*, 2011.

MOORE, J. **Cloud brokers: Simplify complexity**, 2013. URL http://fcw.com/articles/2013/01/07/cloud-broker.aspx. Acessado em 14/09/2014.

MOR, S., ALVES, M., LIMA, J., MAILLARD, N. e NAVAUX, P. O. A. **Eficiência energética em computação de alto desempenho: Uma abordagem em arquitetura e programação para green computing**. Em *XXXVII Seminário Integrado de Software e Hardware-SEMISH*, pág. 346–360, 2010.

NORCOTT, W. D. **Iozone filesystem benchmark**, 2006. URL `http://www.iozone.org/docs/IOzone_msword_98.pdf`. Acessado em 17/05/2014.

NUSSBAUM, L., ANHALT, F., MORNARD, O. e GELAS, J.-P. **Linux-based virtualization for HPC clusters**. Em *Proceedings of the Linux Symposium*, julho 2009. URL `http://kernel.org/doc/ols/2009/ols2009-pages-221-234.pdf`.

OLIVEIRA, R. S.; CARISSIMI, A. S. T. S. S. *Sistemas Operacionais*. Instituto de Informática da UFRGS: Sagra Luzzatto, Porto Alegre, SC, 3 edition, 2004.

PETITET, A., WHALEY, C., DONGARRA, J., CLEARY, A. e LUSZCZEK, P. **A portable implementation of the high-performance linpack, benchmark for distributed-memory computers**, 2012. URL `http://www.netlib.org/benchmark/hpl/`. Acessado em 22/8/2014.

POVOA, L. V., JUNIOR, P. W. B., MONTEIRO, C. E., MUELLER, D., MARCONDES, C. A. e SENGER, H. **Predição de consumo energético com base na utilização de recursos computacionais**. 2013.

QUANG-HUNG, N., LE, D.-K., THOAI, N. e SON, N. T. **Heuristics for energy-aware vm allocation in hpc clouds**. Em *in Proceeding of The 1st International Conference on Future Data and Security Engineering 2014.*, 2014.

QUANG-HUNG, N., THOAI, N. e SON, N. T. **Epobf: Energy efficient allocation of virtual machines in high performance computing cloud**. Em *in Proceeding of International Conference on Advanced Computing and Applications (ACOMP 2013), 2013, pp. 173-182*, 2013.

RAMBHADJAN, M. e SCHUTIJSER, A. **Surfnet cloud computing solutions**. *Universiteit van Amsterdam, System and Network Engineering*, 2010.

ROUSE, M. **Definition - high performance computing (hpc)**, 2007a. URL `http://searchenterpriselinux.techtarget.com/definition/high-performance-computing`. Acessado em 09/05/2014.

ROUSE, M. **Definition - intelligent platform managment (ipmi)**, 2007b. URL `http://searchwindowsserver.techtarget.com/definition/Intelligent-Platform-Management-Interface`. Acessado em 09/05/2014.

SIMONS, J. E. e BUELL, J. **Virtualizing high performance computing**. *SIGOPS Oper. Syst. Rev.*, 44(4):136–145, dezembro 2010. ISSN 0163-5980. URL `http://doi.acm.org/10.1145/1899928.1899946`.

SINGH, J. e SINGH, S. **Optimized energy efficient resource management in cloud data center**. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(7):436–437, 2013. ISSN 2277 128X. URL http://www.ijarcsse.com.

SINHA, R., PUROHIT, N. e DIWANJI, H. **Power aware live migration for data centers in cloud using dynamic threshold**. *International Journal of Computer Technology and Applications*, 2(6), 2011.

STEINDER, M., WHALLEY, I., HANSON, J. E. e KEPHART, J. O. **Coordinated management of power usage and runtime performance**. Em *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, págs. 387–394. IEEE, 2008.

TANENBAUM, A. S. *Redes de Computadores*. Elsevier, Rio de Janeiro, trad. 4 ed. edition, 2003.

TANENBAUM, A. S. e STEEN, M. V. *Distributed Systems: Principles and Paradigms (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006. ISBN 0132392275.

TOP500.ORG. **Top 500 supercomputers site**, 2013. URL http://www.top500.org/. Acessado em 22/8/2014.

URGAONKAR, R., KOZAT, U. C., IGARASHI, K. e NEELY, M. J. **Dynamic resource allocation and power management in virtualized data centers**. Em *Network Operations and Management Symposium (NOMS), 2010 IEEE*, págs. 479–486. IEEE, 2010.

VAQUERO, L. M., RODERO-MERINO, L., CACERES, J. e LINDNER, M. **A break in the clouds: Towards a cloud definition**. *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55, dezembro 2008. ISSN 0146-4833. URL http://doi.acm.org/10.1145/1496091.1496100.

WALTERS, J. P., YOUNGE, A. J., KANG, D.-I., YAO, K. T., KANG, M., CRAGO, S. P. e FOX, G. **Gpu passthrough performance: A comparison of kvm, xen, vmware esxi, and lxc for cuda and opencl applications**. Em *IEEE CLOUD'14*, págs. 636–643, 2014.

WERNER, J. **Uma abordagem para alocação de máquinas virtuais em ambientes de computação em nuvem verde**. Dissertação de Mestrado, Universidade Federal do Rio Grande do Sul - UFRGS, 2011.

WESTPHALL, C. B. e VILLARREAL, S. R. **Princípios e tendências em green cloud computing.** *Revista Eletrônica de Sistemas de Informação*, 12(1), 2013.

YE, K., JIANG, X., CHEN, S., HUANG, D. e WANG, B. **Analyzing and modeling the performance in xen-based virtual cluster environment**. Em *12th IEEE International Conference on High Performance Computing and Communications, HPCC 2010, 1-3 September 2010, Melbourne, Australia*, págs. 273–280, 2010. URL http://dx.doi.org/10.1109/HPCC.2010.79.

YOUNGE, A., HENSCHEL, R., BROWN, J., VON LASZEWSKI, G., QIU, J. e FOX, G. **Analysis of virtualization technologies for high performance computing environments**. Em *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, págs. 9–16, July 2011.