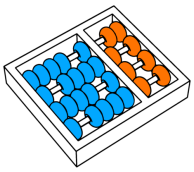
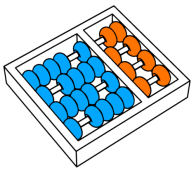


Daniel Guimarães do Lago

"Escalonamento Ciente de Energia em Nuvens usando Controle de Refrigeração Ativa e DVFS"

CAMPINAS
2012



Universidade Estadual de Campinas
Instituto de Computação

Daniel Guimarães do Lago

"Escalonamento Ciente de Energia em Nuvens usando Controle de Refrigeração Ativa e DVFS"

Orientador(a): **Prof. Dr. Edmundo Roberto Mauro Madeira**

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Computação da Universidade Estadual de Campinas para obtenção do título de Mestre em Ciência da Computação.

Este exemplar corresponde à versão final da Dissertação defendida por Daniel Guimarães do Lago, sob orientação de Prof. Dr. Edmundo Roberto Mauro Madeira.

Assinatura do Orientador(a)

CAMPINAS
2012

FICHA CATALOGRÁFICA ELABORADA POR
MARIA FABIANA BEZERRA MULLER - CRB8/6162
BIBLIOTECA DO INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E
COMPUTAÇÃO CIENTÍFICA - UNICAMP

Lago, Daniel Guimarães do, 1985-
L137e Escalonamento ciente de energia em nuvens usando controle de refrigeração ativa e DVFS / Daniel Guimarães do Lago. – Campinas, SP : [s.n.], 2012.

Orientador: Edmundo Roberto Mauro Madeira.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Computação em nuvem. 2. Energia - Consumo. 3. Sistemas distribuídos. I. Madeira, Edmundo Roberto Mauro,1958-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

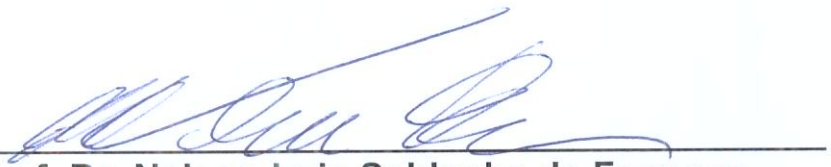
Informações para Biblioteca Digital

Título em inglês: Power-aware scheduling on clouds using active cooling control and DVFS
Palavras-chave em inglês:
Cloud computing
Energy consumption
Distributed systems
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Edmundo Roberto Mauro Madeira [Orientador]
Fábio Luciano Verdi
Nelson Luis Saldanha da Fonseca
Data de defesa: 29-11-2012
Programa de Pós-Graduação: Ciência da Computação

TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 29 de Novembro de 2012, pela Banca examinadora composta pelos Professores Doutores:

Prof. Dr. Fábio Luciano Verdi
DComp / UFSCar



Prof. Dr. Nelson Luis Saldanha da Fonseca
IC / UNICAMP

Prof. Dr. Edmundo Roberto Mauro Madeira
IC / UNICAMP

Instituto de Computação
Universidade Estadual de Campinas

Escalonamento Ciente de Energia em Nuvens usando Controle de Refrigeração Ativa e DVFS

Daniel Guimarães do Lago[1]

29 de Novembro de 2012

Banca Examinadora:

- Prof. Dr. Edmundo Roberto Mauro Madeira (*Orientador*)
- Prof. Dr. Fábio Luciano Verdi
  Departamento de Computação - UFSCar
- Prof. Dr. Nelson Luis Saldanha da Fonseca
  Instituto de Computação - UNICAMP
- Dr. Christian Esteve Rothenberg
  Centro de Pesquisa e Desenvolvimento em Telecomunicações - CPqD (*Suplente*)
- Dra. Islene Calciolari Garcia
  Instituto de Computação - UNICAMP (*Suplente*)

[1]Suporte financeiro de: Bolsa CAPES e Bolsa FAPESP (processo 2010/14592-9)

Abstract

This work presents algorithms that operate in the scheduling process of virtual machines in cloud computing, based on concepts of Green Computing. These algorithms focus on energy consumption minimization, and also provide quality of service based on virtual load execution priority. To achieve energy consumption minimization, some features are used, such as underutilized server shutdown, migration of server loads that operate below a certain threshold, DVFS, and Fan Control. The choice of which server will receive the virtual loads is obtained through energy efficiency maximization, given by the ratio MIPS to power consumption per server. Furthermore, the algorithms of this work also act on the submission of loads with priorities, allowing high priority loads to run in lower times. Results from simulations showed that the developed algorithms, when confronted against other studied algorithms, can improve energy consumption in clouds composed of heterogeneous *data centers*, tying with the best algorithms in the case of homogeneous *data centers*, also providing quality service.

Resumo

Neste trabalho são apresentados algoritmos que atuam no processo de escalonamento de máquinas virtuais para computação em nuvem, fundamentados nos conceitos de *Computação Verde*. Estes algoritmos focam na minimização da energia consumida, além de proverem qualidade de serviço baseada em prioridade de execução de cargas virtuais. Para atingir o objetivo da minimização de consumo de energia são usados alguns recursos, tais como: desligamento de servidores subutilizados, migração de cargas de servidores que operam abaixo de um determinado limiar, *DVFS* e *Fan Control*. A escolha de qual servidor receberá as cargas virtuais é obtida através da maximização da eficiência em energia, dada pela razão entre MIPS e a potência consumida em cada servidor. Além disso, os algoritmos deste trabalho atuam na submissão de cargas com prioridades, permitindo com que cargas de maior prioridade sejam executadas em tempos menores. Resultados obtidos através de simulações mostraram que os algoritmos desenvolvidos, quando confrontados a outros algoritmos pesquisados, podem melhorar o consumo de energia em nuvens compostas por *data centers* heterogêneos, empatando com os melhores algoritmos no caso de *data centers* homogêneos, além de prover qualidade de serviço.

Agradecimentos

Agradeço à minha família, em especial às mulheres que são a razão da minha existência, mamãe Vânia, pelo amor incondicional; vovó Teresinha, pelo exemplo de bondade; e vovó Lolita, parâmetro absoluto de honestidade. Ao meu avô Tomé (*in memorian*), pela educação, princípios e valores que me ensinou; relembrando também suas saudosas palavras quando lhe perguntavam o que fazia: "promotor por profissão, professor por vocação". Aos meus irmãos Davi, Lucas, André, pelos exemplos de tranquilidade, visão e obstinação. À minha irmã Luísa, sempre compreensiva, que me trouxe serenidade em dias difíceis. Ao tio Mané, que tanto me incentivou nesses últimos anos.

Ao meu orientador, professor Edmundo, sempre prudente e equilibrado em suas ponderações, cumprindo de forma exímia sua função.

Ao Luiz Fernando Bittencourt, sempre atencioso, cujos olhos clínicos foram imprescindíveis para a conclusão desse trabalho. À Unicamp, que demonstrou merecer a fama que possui como excelência em computação. Aos meus amigos do LRC, em especial Thiago Genez, Gustavo Alkmim e Rafael Lopes. Aos meus amigos de república: Júlio por reensinar o valor da disciplina; e Danilo por reforçar a importância do foco e autosuficiência.

Aos professores Luiz Henrique e Marluce, da UFLA, cujos ensinamentos me permitiram chegar até aqui. Ao professor Wilbur Trajano da UFRGS que, por várias madrugadas, me auxiliou acerca do comportamento físico de computadores.

À parcela honesta dos contribuintes brasileiros, que permite o financiamento científico e educacional nesse país; e à parcela honesta das autoridades que faz uso decente do resultado do suor da população.

À FAPESP (processo 2010/14592-9) e à CAPES, pelo apoio financeiro que possibilitou o desenvolvimento desse trabalho.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Uma das mais notáveis tecnologias emergentes da Internet nos dias de hoje é a Computação em Nuvem. Este conceito, iniciado em 1999 por Fredrik Malmer (WebOS) [1], refere-se a um modelo [2] que permite acesso em rede ubíquo, conveniente e sob demanda para uma gama de recursos computacionais configuráveis (ex: redes, servidores, armazenamento, aplicações e serviços), que podem ser rapidamente aprovisionados e liberados com mínimo esforço administrativo ou interação do provedor de serviços. O modelo de nuvem é composto de cinco características essenciais — auto-serviço sob demanda, acesso banda larga à rede, aprovisionamento de recursos, elasticidade rápida e serviço pago pelo uso —, três níveis de modelos de serviços — Software como serviço (SaaS), Plataforma como serviço (PaaS) e Infraestrutura como serviço (IaaS) — e quatro níveis de modelo de implantação — Nuvem privada, nuvem comunitária, nuvem pública e nuvem híbrida. Mais do que isso, atualmente é possível ver muitos trabalhos, e até mesmo conferências, onde é proposto "tudo" como um serviço (XaaS) [3].

Muitos especialistas declararam 2010 como sendo o "Ano da Nuvem" [4]. Este fato foi motivado pelo lançamento de diversos dispositivos de baixo poder de processamento como o *iPad* e crescimento das vendas de *netbooks*, *tablets* e afins. Esta previsão se mostrou verdadeira; prova disso é o surgimento de plataformas para Computação em Nuvem de grandes corporações como o *Microsoft Azure* [5] e *Google App Engine* [6], além da expansão de outras, como o *Amazon EC2* [7]. De fato, nestes dois últimos anos houve um crescimento tão acentuado nessa área que, hoje, é possível identificar a nuvem como sendo uma mega-tendência [8].

Foi constatado [9] que hoje, nos EUA, o consumo de energia dos servidores é da mesma magnitude do consumo de energia de todas as televisões desse país. Além disso, em recente pesquisa realizada pela Microsoft [10] foi verificado que se as empresas dos EUA fizessem uso amplo da computação em nuvem economizariam quase R$ 900.000,00/ano. Esta economia provêm da redução de gastos operacionais, de energia elétrica ociosa, de

resfriamento de *data centers*, de aquisição de hardware, do gerenciamento de *desktops* e servidores, e da implantação centralizada de aplicações. Além disso, com a computação em nuvem existem outras vantagens, citando como exemplo colaboração, escalabilidade, economia de energia, elasticidade, armazenamento *online*, disponibilidade e serviços. Tais vantagens são abordadas com maior profundidade na Seção 2.1.

Apesar de todos estes benefícios, algumas organizações em prol do meio ambiente questionam a computação em nuvem. O Greenpeace [11] estima que em 2020 o número de computadores pessoais quadruplicará, ultrapassando 4 bilhões de dispositivos, com a emissão de gás carbônico de *laptops* ultrapassando a de computadores de mesa (representando 22% das emissões). O número de dispositivos móveis (exceto *laptops*) duplicará, ultrapassando a barreira dos 5 bilhões de dispositivos mas, apesar deste crescimento, suas emissões crescerão em apenas 4%.

Mesmo com a crise econômica mundial, tem ocorrido expansão significativa da computação baseada em nuvem, além da grande atenção ao crescimento do desenvolvimento de projetos de *data centers* eficientes em energia e expansão de tamanho e de escala dos *data centers* construídos pelas grandes empresas. Algumas questões chave que nascem destes fatos são: quão grande será o consumo de eletricidade e emissões indiretas de gases? Onde a nuvem será construída e quais serão as fontes de energia para alimentá-la? Qual será o impacto do *data center* com relação à demanda de combustíveis fósseis? De que maneira a eficiência e as melhorias de projetos poderão reduzir o aumento do consumo de energia e das taxas de emissão de poluentes? O que consumimos e o que é estimado para alimentar a computação em nuvem?

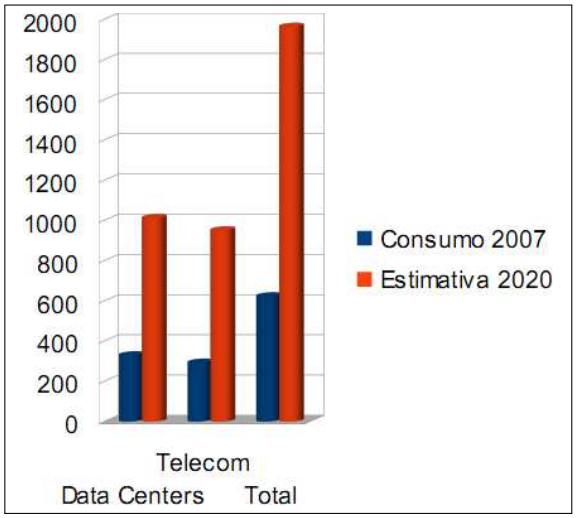


Figura 1.1: Consumo e Estimativa de Consumo de Energia na Computação em Nuvem (Métrica: Bilhões de kWh)

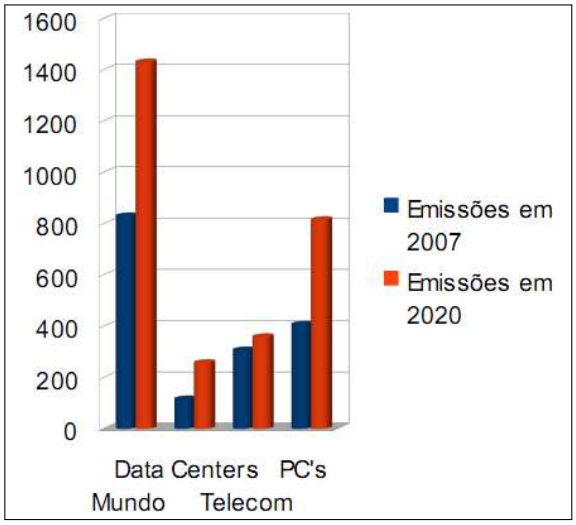


Figura 1.2: Emissões de MtCO2e (emissões de 2007 e emissões projetadas para 2020)

Tang et al. afirmam que o custo de energia para manter *data centers* que operam em uma nuvem é vertiginosamente alto [12]. De fato, o Greenpeace verificou que em 2007 já foram consumidos 623 bilhões de kWh e estima-se que em 2020 este valor atinja a casa dos 1,96 trilhões de kWh (Figura 1.1). Lefèvre [13] estimou que em 2009 todos os computadores do mundo usaram aproximadamente 90.000 MW de potência, o que representa 10% da energia global consumida. Além disso o impacto ambiental é preocupante: em 2007 foram liberadas na atmosfera 830 toneladas de dióxido de carbono apenas para alimentar a computação em nuvem, e há previsão de piora para 2020. Com o crescimento desta área estima-se que a computação em nuvem atingirá uma preocupante emissão de 1430 toneladas de dióxido de carbono por ano, conforme pode ser observado na Figura 1.2.

A utilização de aplicativos dinâmicos cientes de energia pode tratar a subutilização de servidores bem como os crescentes custos de energia em um *data center* [14]. Neste trabalho são propostos e avaliados algoritmos para escalonamento de tarefas em aglomerados computacionais que operam em nuvem, com princípios aderentes à Computação Verde, tendo como foco principal minimizar o consumo e custos de energia em *data centers*, assim como a poluição gerada por estes. Para atingir este objetivo são utilizados conceitos de gerenciamento distribuído de energia, dimensionamento dinâmico de tensão e frequência, migração de cargas virtuais, e desligamento de servidores subutilizados. Também é aplicado controle de refrigeração para minimizar o consumo de energia, originalmente desenvolvido para redução de ruído.

Em uma nuvem é desejável que determinadas cargas de trabalho sejam processadas mais rapidamente do que outras de menor importância, ou seja, essas cargas requerem uma prioridade maior do que as demais. Lidar com prioridades traz vantagens para

o processamento seletivo de cargas, como a redução do tempo de execução, mas pode introduzir problemas, como atrasos no processamento das cargas de baixa prioridade. É foco desse trabalho lidar com prioridades, de forma a minimizar os tempos de execução de cargas com prioridades elevadas, se possível sem comprometer cargas com menores prioridades e evitando elevar o consumo de energia da nuvem.

Em outras palavras pode-se dizer que esse trabalho busca uma solução para o seguinte problema: dado um conjunto composto por H servidores, M máquinas virtuais e C cargas de trabalho, tendo cada carga C_i uma prioridade P_i associada, efetuar o escalonamento de M em H e de C em M, minimizando a energia consumida pelas tarefas executadas e os tempos de execução das cargas com maior prioridade do conjunto P, de modo que a minimização do tempo das cargas com maior prioridade não impacte em um aumento substancial do consumo de energia.

Para o processo de escalonamento de máquinas virtuais, os algoritmos propostos tentam alocar as máquinas virtuais com a melhor eficiência em energia, dada pela razão entre MIPS e potência consumida, o que é especialmente útil em *data centers* heterogêneos, onde temos conjuntos de máquinas com configurações distintas de hardware. Os algoritmos apresentados nesse trabalho também possuem características para lidar com qualidade de serviço no processo de escalonamento, permitindo reduzir o tempo de processamento de cargas com maiores prioridades submetendo-as para máquinas virtuais mais aptas, sem comprometer significativamente o consumo de energia.

Para validar os algoritmos propostos, a simulação foi escolhida como método. O simulador *CloudSim* [15] foi utilizado, sendo este desenvolvido pelo laboratório CLOUDS da Universidade de Melbourne, especializado no assunto desde 2004. Os algoritmos propostos foram implementados e executados no simulador, com os resultados apresentados nesta dissertação.

As contribuições deste trabalho são o escalonamento com foco na minimização do consumo de energia de *data centers* que operam em nuvem, usando migração de cargas virtuais; dimensionamento dinâmico de tensão e frequência; e aplicação do mecanismo *Fan Control* (originalmente usado com o objetivo de redução de ruído); além do trabalho com prioridades, permitindo cargas específicas, de alta prioridade, serem processadas mais rapidamente. As principais diferenças dos algoritmos propostos neste trabalho, comparados aos semelhantes encontrados na literatura, são os fatos de considerar em seus cálculos a economia que pode ser conseguida pelo mecanismo de *Fan Control* e, especialmente em caso de *data centers* heterogêneos, a forma de tratar migrações de máquinas virtuais.

Esta dissertação está organizada da seguinte maneira: na Seção 2 são apresentados conceitos básicos empregados neste trabalho; na Seção 3 são mostrados alguns trabalhos relacionados com o que está sendo proposto; na Seção 4 são expostos os algoritmos propostos nesta dissertação; na Seção 5 são analisados os resultados conseguidos pelos

algoritmos propostos; e na Seção 6 são mostradas as conclusões que foram obtidas com a realização deste trabalho, além de alguns trabalhos futuros.

Capítulo 2

Conceitos Básicos

Nesse capítulo são apresentados conceitos básicos necessários para o entendimento deste trabalho.

2.1 Computação em Nuvem

Knorr et al. [16] definem Computação em Nuvem como sendo "um novo modelo de suplemento, consumo e entrega para serviços em Tecnologia da Informação baseado na internet, que tipicamente envolve uma disposição sobre a internet de recursos dinamicamente escaláveis e frequentemente virtualizados". Outro ponto de vista é provido por Vaquero et al. [17], que definem nuvens como sendo grandes repositórios de recursos virtualizados (hardware, plataformas de desenvolvimento e/ou serviços), facilmente acessíveis. Estes recursos podem ser reconfigurados dinamicamente de modo a se ajustar de acordo com as cargas, otimizando a utilização destes mesmos recursos. Este repositório de recursos é tipicamente explorado utilizando um modelo do tipo pagamento-por-uso, onde os fornecedores de infraestrutura oferecem garantias no formato de SLAs (*service level agreements*).

A Computação em Nuvem tem o potencial de transformar grande parte da indústria de Tecnologia da Informação, tornando softwares ainda mais atrativos como um serviço e modelando a forma como os dispositivos de hardware são projetados e adquiridos. Desenvolvedores com ideias inovadoras para novos serviços de Internet não mais necessitam de gastar seu potencial financeiro adquirindo hardware para implantar seus serviços e nem um administrador para gerenciá-lo.

Além disso, o custo de utilização de computação em nuvem é menor e o resultado é bem mais rápido, o que a torna escalável. Ambrust et al. [18] vão além e afirmam que a elasticidade de recursos em uma nuvem não tem precedentes na história da tecnologia da informação. Elasticidade se trata de uma das principais características na computação em nuvem, podendo ser entendida como a capacidade do ambiente computacional em nuvem

de aumentar ou diminuir os recursos demandados e aprovisionados para cada usuário, aumentando ou diminuindo a capacidade disponibilizada. Quando o usuário necessita de maior capacidade da nuvem, ele solicita essa maior capacidade, e quando tal capacidade não é mais necessária ela é liberada. Essa alocação dinâmica de recursos permite a economia de escala.

De fato, os ganhos com a computação em nuvem são tão acentuados que a Casa Branca apresentou no meio de 2011 uma estratégia para mover grande parte dos seus data centers para a nuvem, permitindo, além da elasticidade e demais vantagens da computação em nuvem, uma economia da ordem de 60 bilhões de dólares, o que representa 75% de todos os gastos dos Estados Unidos com manutenção de data centers [19].

Do ponto de vista físico, as nuvens são compostas por um ou mais *data centers* e estes, por sua vez, tipicamente compostos por milhares de máquinas em rede capazes de realizar trabalhos submetidos por usuários. As máquinas que compõem os *data centers* são ligadas entre si. Quando *data centers* geograficamente distintos compõem uma nuvem, a estes é dado o nome de *data centers* federados.

Normalmente um servidor efetua o escalonamento desses trabalhos e os servidores que os recebem efetuam o processamento desejado. Zhang et al. [20] afirmam que é um desafio o gerenciamento de energia em nuvem, assunto que pretende ser abordado pelo presente trabalho, onde se propõe atuar no escalonamento de tarefas em nuvens visando os objetivos da Computação Verde.

Do ponto de vista do usuário, as nuvens podem ser acessadas como instâncias de máquinas virtuais. Por exemplo, na plataforma de computação em nuvem Amazon EC2, os usuários podem usar computadores virtuais para suas próprias aplicações, permitindo a execução de um serviço na rede no qual o usuário poderá iniciar uma imagem de uma máquina virtual, a qual é dado o nome de instância.

O escalonamento nas nuvens também pode ser visto sob dois pontos principais: do usuário e do provedor da nuvem. Do ponto de vista do usuário um escalonador pode ter objetivos como reduzir tanto o tempo de execução quanto o custo para o usuário [21]. Do outro lado, do ponto de vista do provedor, um escalonador de tarefas deve melhorar a utilização de recursos enquanto reduz os custos de manutenção e consumo de energia [14]. Nesse trabalho abordamos o escalonamento do ponto de vista do provedor.

Em uma nuvem cada máquina física pode executar uma ou mais instâncias de máquinas virtuais e também é possível efetuar a migração de máquinas virtuais entre os servidores [22]. Uma das ferramentas para efetuar esta migração é o vMotion [23]. Na Seção 2.3 detalharemos como esse recurso funciona.

2.2 Computação Verde

A preocupação com o meio ambiente tem aumentado em todas as áreas do conhecimento humano. Em computação, não é diferente. Murugesan [24] define Computação Verde como sendo "o estudo e prática de projetar, construir, utilizar e dispôr de computadores, servidores e sistemas associados — como monitores, impressoras, dispositivos de armazenamento, sistemas de redes e comunicação — de forma eficiente e efetiva com impacto mínimo ao meio ambiente".

Michael Dell [25] afirma que sempre acreditou que Tecnologia da Informação fosse o motor de uma economia eficiente, mas ela também pode se direcionar para ser mais *green*, e Wang [26] sugere que a ligação entre Computação Verde e o baixo consumo energético é imediata. Diversas iniciativas têm sido realizadas para tornar a computação menos poluidora no sentido de consumir menos energia e poluir menos o ambiente. Nos Estados Unidos, a Organização para Desenvolvimento e Cooperação Econômica afirma que existem mais de 90 iniciativas do governo americano para *Green ICT's* [27]. Um consórcio internacional denominado *Green Grid* [28] foi criado, composto por centenas de empresas, tais como: AMD, APC, Dell, HP, IBM, Intel, Microsoft e VMware, dedicado em desenvolver a eficiência energética em ecossistemas de *data centers*.

Fabricantes vêm focando também a Computação Verde. A AMD, por exemplo, recentemente lançou sua primeira série de processadores para servidores que operam em nuvem, com consumo médio de energia [29] (ACP) de 32 *watts*.

Para atingir os objetivos da Computação Verde as principais abordagens estão relacionadas com longevidade de produtos, reciclagem de materiais, *telecommuting* (teleconferência, telepresença, VoIP); além dos itens contemplados neste trabalho, sendo eles eficiência de algoritmos, alocação de recursos, virtualização, servidores de terminais e administração de energia.

A virtualização permite atingir os objetivos da computação verde através de consolidação e uso de dispositivos apropriados. Pode-se exemplificar consolidação com o seguinte cenário: no passado era necessário que cada sistema computacional tivesse sua própria estrutura de armazenamento para funcionar. A virtualização em armazenamento permite que sistemas acessem um subsistema de armazenamento que esteja em algum lugar na rede. Isso também quer dizer que cópias de dados armazenados em cada disco de cada sistema agora também podem ser armazenados no mesmo subsistema de armazenamento compartilhado. Está claro que esta abordagem reduz o número de dispositivos de armazenamentos necessários, a quantidade de energia requerida, o calor gerado, e como efeito colateral também reduz os custos administrativos tais como *backups*, manutenção de armazenamentos de arquivos e afins.

Com relação ao uso de dispositivos apropriados pode-se usar o seguinte cenário: note

que em uma organização há softwares e arquivos que são acessados com frequências distintas. Pode-se, por exemplo, armazenar e executar programas que são usados com maior frequência em virtualizações executadas nas máquinas com maior desempenho e velocidade, o que normalmente está relacionado com maior consumo de energia também. Pode-se também armazenar softwares e arquivos menos utilizados em virtualizações de máquinas mais econômicas em consumo de energia. Além disso, pode-se armazenar as informações e software usados muito raramente em virtualizações que são ligadas apenas quando há a necessidade de acesso a tais dados.

Uma das preocupações levantadas durante o desenvolvimento deste trabalho foi com relação à redução da vida útil das máquinas, decorrente do possível desgaste prematuro dos componentes eletrônicos, por causa do ciclo de ligamento e desligamento. Em [30] é visto que os componentes de hardware de computadores que mais apresentam defeitos, depois de teclados, são os discos rígidos. Pinheiro et al. [31] afirmam que constantes ciclos de energia, resultantes de constantes ligamentos e desligamentos, não afetam substancialmente servidores nos primeiros dois anos. No entanto, para drives com 3 anos ou mais, tais ciclos podem aumentar as taxas de defeito nestes dispositivos em 2%, o que deve ser levado em conta de acordo com os objetivos em questão.

2.3 Migração de Máquinas Virtuais

A tecnologia de máquinas virtuais emergiu recentemente como um componente básico para *data centers* e aglomerados, principalmente devido às suas capacidades de isolamento de cargas de trabalho, consolidação e migração [32]. De modo geral esse recurso permite servir múltiplos usuários de uma forma segura, eficiente e flexível. Como resultado essas infraestruturas virtualizadas são consideradas componentes-chave para conduzir o emergente paradigma da Computação em Nuvem [33].

A migração de cargas virtuais procura aperfeiçoar a gerência, desempenho e tolerância a falhas de sistemas. Mais especificamente, as razões que justificam a migração de máquinas virtuais em um sistema de produção incluem: a necessidade de fazer balanceamento de carga, que pode ser conseguida através da migração de máquinas virtuais de servidores sobrecarregados/superaquecidos; e a necessidade de seletivamente desligar servidores para manutenção após migrar suas cargas de trabalho para outros servidores [34]. Essa migração pode ser feita sem gerar indisponibilidade de serviço e não é perceptível para o usuário, podendo ser realizada, por exemplo, para retirar a carga de máquinas subutilizadas para desligá-las.

Nos algoritmos apresentados neste trabalho o conceito de migração de máquinas virtuais é usado com o objetivo de migrar cargas de servidores subutilizados e, em seguida, desligá-los, desse modo a maximizar a economia de energia. Mais especificamente, é veri-

ficado de tempos em tempos a qual percentual da capacidade máxima de processamento os servidores estão operando. Em seguida tenta-se efetuar a migração das cargas daqueles que estão operando abaixo de um determinado limiar e, caso seja possível migrar todas as suas máquinas virtuais, desliga-os. Do ponto de vista técnico estes desligamentos podem ser feitos com os servidores enviando uma sinalização ACPI para a placa-mãe [35], e podem ser religados via um pacote *Wake on LAN* proveniente do escalonador [36].

2.4 Dimensionamento Dinâmico de Tensão e Frequência

Dimensionamento Dinâmico de Tensão (DVS - *Dynamic Voltage Scaling*) [37] é uma técnica de gerência de energia em uma arquitetura computacional onde a tensão de determinado componente pode ser alterada de acordo com parâmetros específicos. No âmbito computacional, o aumento de tensão propicia maior estabilidade de um hardware, ao custo de reduzir sua vida útil, por exemplo, com eletromigração ou injeção de transportador quente [HCI]); e a redução de tensão permite reduzir o consumo de energia elétrica do dispositivo de hardware.

Em circuitos digitais baseados no transistor MOSFET, é possível efetuar a alternância de estados de tensões de alimentação entre "alta" e "baixa". Fabricantes de processadores se beneficiam dos recursos de DVS para minimizar o consumo de energia de seus dispositivos e, também, reduzem a frequência de tais equipamentos, permitindo-os permanecer estáveis. Esta técnica de dimensionar dinamicamente a tensão e a frequência recebe o nome de *DVFS* (*Dynamic Frequency and Voltage Scaling*).

As tecnologias *Intel SpeedStep* [38] e *AMD Coll'n'Quiet* [39] ajustam, automaticamente, em nível de hardware, a frequência e tensão de operação dos seus processadores de acordo com suas cargas de trabalho. Se há altas cargas de trabalho, o processador aumenta os níveis de tensão e frequência, caso contrário diminui estes níveis. Os estados possíveis de frequência e tensão são chamados de *P-States*. Embora os processadores venham com essas tecnologias, o suporte a elas muitas vezes vem desativado na placa mãe, requerendo atenção do administrador do sistema para que as ative.

Para calcular o impacto em energia quando o *DVFS* está ativado, é necessário verificar a diferença entre a energia consumida pelo processador, operando em baixa frequência e tensão, e a energia consumida pelo mesmo processador trabalhando em capacidade máxima. A potência consumida pelo núcleo do processador pode ser obtida a partir da fórmula $P = cfV^2$, onde P é a potência consumida pelo processador em *watts*; c é a capacitância dos transistores internos comutada por ciclo de *clock* do processador e medida em *farads* (determinado pela litografia, constante independentemente da frequência ou tensão); f é a frequência de trabalho do processador em *hertz* e V é a tensão de alimentação da CPU em *volts*. A energia consumida pelo processador pode ser calculada pela

potência na qual ele trabalha multiplicada pelo tempo de operação. A consequência disso é que a energia consumida cresce linearmente com a frequência de trabalho do processador e quadraticamente com a tensão.

Para demonstrar a influência das tecnologias supramencionadas no consumo de energia (e consequentemente de energia dissipada), é tomado como base um processador Intel Pentium M de 1,6 *GHz*, com a tecnologia *SpeedStep* ativada. Neste processador as frequências se ajustam de 600 *MHz* até 1,6 *GHz*, em passos de 200 *MHz*. A tensão utilizada na frequência máxima é de 1,484 *volts* e na frequência mínima é 0,956 *volts*. Pode-se verificar que o consumo de energia, no ajuste da frequência máxima para a frequência mínima, é reduzido em aproximadamente 84% e, portanto, bastante significativo. Os cálculos são apresentados a seguir:

$$1 - \frac{600 \times 0,956^2}{1600 \times 1,484^2} \approx 1 - \frac{548,362}{3523,61} \approx 84\%$$

Nos algoritmos propostos neste trabalho o conceito de *DVFS* é sempre aplicado, com o objetivo de possuir um consumo de energia inteligente, usando a potência do processador adaptada às cargas que necessita processar.

2.5 Fan Control

Fan Control é um dos nomes dados para representar a tecnologia que monitora a temperatura dos computadores e, mediante condições térmicas, regula a intensidade dos dissipadores ativos. Diversas fabricantes de placas-mãe possuem vários modos de operação dessa tecnologia, cada um com seus próprios nomes, por exemplo: AOpen: SilentTEK [40]; ASUS: Q-Fan [41]; MSI: Core Center [42]; Abit: μGuru [43]; Gigabyte: EasyTune 6 [44]; Intel S478: Active Monitor / Desktop Control Centre [45]; Intel S775: Desktop Utilities [46] e Dell Inspiron/Latitude/Precision: I8kfanGUI [47].

Algumas dessas tecnologias controlam a intensidade de trabalho do dissipador ativo através do sistema operacional ou diretamente pela BIOS, sendo que todas apresentam comportamentos semelhantes: o administrador regula duas temperaturas de controle, uma para "baixa rotação" e outra para "alta rotação". Quando o sistema está operando em temperatura inferior àquela de "baixa rotação" seus dissipadores ativos são desligados. Quando a temperatura está entre "baixa rotação" e "alta rotação", o dissipador é polarizado com baixa tensão (normalmente 5–7 *volts*) e trabalha, em geral, abaixo da metade de sua capacidade. Quando o sistema atinge a temperatura de "alta rotação" os dissipadores são polarizados com alta tensão (normalmente 12 *volts*) e trabalham em plena capacidade. Embora tais tecnologias estejam presentes na maior parte das placas-mãe, normalmente vêm desativadas por padrão.

Apesar do foco principal do *Fan Control* ser a redução de ruído dos periféricos de resfriamento do hardware, pode-se observar um efeito secundário: tomando como base a fórmula de potência $P = \frac{V^2}{R}$, onde P é a potência elétrica medida em *watts*, V é a diferença de potencial elétrico medida em *volts* e R é a resistência elétrica do circuito medida em *ohms*, percebe-se que a potência de um equipamento eletrônico está relacionada com sua tensão e resistência.

Os dissipadores ativos comerciais hoje (populares "ventoinhas") consomem em média de 2 a 8 *watts* quando alimentados por uma tensão de 12 *volts*. Repare que ventoinhas possuem uma resistência associada e que, quando expostas às tecnologias de *Fan Control*, mesmo com a redução da tensão, a resistência não varia. Tomando um dissipador ativo de qualquer potência (denotada por "x" *watts*), inicialmente alimentado por 12 *volts*, ao reduzir sua tensão para 5 *volts*, seu consumo de energia será reduzido em aproximadamente 83%, conforme mostrado abaixo:

$$R = \frac{12^2}{x} = \frac{5^2}{x - \Delta x} \Rightarrow 144x - 144\Delta x = 25x \Rightarrow \Delta x = \frac{(144 - 25)x}{144} = \frac{119}{144}x \approx 0,83x$$

Com as operações apresentadas é visto que utilizando tecnologias de *Fan Control* é possível reduzir de forma significativa o consumo de energia e a consequente produção de calor.

Existem diversas iniciativas que atuam na infraestrutura do *data center* para resfriá-lo ([48] [49] [50] entre muitos outros). No entanto, na abordagem usada neste trabalho, a atuação é na própria geração de temperatura pelos servidores, não atacando, portanto, mecanismos de infraestrutura, como aparelhos de ar-condicionado ou exaustores.

Neste trabalho é usado o mecanismo de *Fan Control* juntamente com *DVFS* para otimizar o consumo de energia.

Capítulo 3

Trabalhos Relacionados

Neste capítulo são apresentados trabalhos que apresentam alguns objetivos semelhantes ao deste.

Beloglazov et al. [51] propõem um sistema de administração de recursos eficiente em consumo de energia que reduz os custos operacionais e provê qualidade de serviço. Nesse trabalho, a economia de energia é obtida através da consolidação contínua de máquinas virtuais de acordo com a utilização de recursos, topologias de redes virtuais estabelecidas entre as máquinas virtuais e estados de temperatura dos nós de computação. Resultados obtidos por simulação apresentaram economia de energia, que chega a 83% comparada ao sistema sem gerência de energia, e também garantindo a qualidade de serviço.

A seguir são apresentados alguns trabalhos, relacionados com este, que lidam com economia de energia no contexto da computação em nuvem.

Berl et al. [52] estudam e revisam o uso de métodos e técnicas eficientes em energia no contexto da Computação em Nuvem, em especial no campo de hardware e de infraestrutura de redes. O trabalho mostra que a instalação de *plug-ins* específicos e centros de controle de energia para redes de larga escala de software e hardware pode atingir impacto significativo na nuvem, reduzindo a energia consumida pela execução do software e pelo hardware, com o aperfeiçoamento do balanceamento de carga e redução da energia de comunicação e do efeito estufa decorrente das emissões de CO_2.

Garg et al. [53] propõem um algoritmo quase ótimo de políticas de escalonamento que exploram a heterogeneidade entre múltiplos *data centers* de um provedor de nuvem. São considerados vários fatores relacionados com energia (tais como custo, taxa de emissão de carbono, carga e eficiência de consumo do processador), enquanto se alterna entre diferentes *data centers* de acordo com sua localização, projeto de arquitetura e sistema de administração. O autor afirma que as políticas de escalonamento propostas permitem atingir economias de até 25% em comparação com as políticas atuais.

Binder e Suri [54] propõem um algoritmo probabilístico de alocação e despacho de

tarefas que minimiza o número de servidores ativos requeridos, gerenciando o ambiente e conseguindo reduzir o consumo de energia, mas garantindo qualidade de serviço em SLAs.

Nathuji et al. [55] e Hu et al. [56] usam máquinas virtuais com o objetivo de reduzir o consumo de energia em ambientes virtualizados em sistemas empresariais.

Em [57] Yanggratoke et al. propõem uma resolução para o problema de alocação de recursos em grandes nuvens baseado em um protocolo que provê uma solução heurística. Como resultado conseguiram minimizar o consumo de energia através de consolidação de servidores quando o sistema está subutilizado e está prevista uma alocação razoável de recursos para o caso de sobrecarga.

Os próximos trabalhos citados, além de objetivarem a economia de energia, também lidam com a adição de algum tipo de qualidade de serviço em uma nuvem.

Duy et al. [58] projetam, implementam e avaliam um algoritmo de escalonamento integrando um preditor de redes neurais para otimizar o consumo de energia em servidores de uma nuvem. Este preditor é construído para prever a carga de trabalho futura baseada em um histórico de demanda. De acordo com a predição, o algoritmo desliga servidores não utilizados e os reinicia para minimizar o número de servidores em execução, minimizando desta forma também o uso da energia. Para avaliação foram efetuadas simulações com duas cargas de trabalho e verificado que este modelo é capaz de reduzir em até 46% o consumo de energia.

Srikantaiah et al. [59] estudam como obter a consolidação de eficiência em energia baseando-se no inter-relacionamento entre consumo de energia, utilização de recursos e desempenho das cargas de trabalho consolidadas. É mostrado que existe um ponto ótimo de operação entre estes parâmetros baseado no problema do empacotamento aplicado ao problema de consolidação.

Bertini et al. [60] buscam reduzir o impacto ambiental em clusters de servidores web. Eles modelam o problema selecionando quais servidores devem permanecer ligados e como devem ser seus desempenhos de processamento através de Programação Inteira Mista, combinando suas soluções com teoria de controle. Seus resultados aplicados em *data centers* pequenos, de até 10 máquinas, mostraram redução no consumo de energia de até 40%. Também aplicaram controle de QoS para tentar garantir eficiência em energia e uma boa experiência do usuário.

Xu et al. [61] propõem estratégias de escalonamento com múltiplas restrições de *QoS* para múltiplos *workflows* em computação em nuvem, aumentando a taxa de sucesso para tarefas de múltiplos *workflows* com diferentes requisitos de *QoS*. Nesse trabalho não há preocupação quanto à economia de energia.

Como se pôde ver nos trabalhos mencionados, muitas iniciativas foram tomadas baseadas em diversos aspectos para tornar os *data centers* que operam numa nuvem mais amigáveis ao meio ambiente. Diferente de todos eles, este trabalho visa atingir a redução

Proposta	Power Aware	Fan Control	Workflow	QoS
Beloglazov et al. [51]	X	*[1]		X
Berl et al. [52]	X			
Garg et al. [53]	X			
Binder e Suri [54]	X			
Nathuji et al. [55] e Hu et al. [56]	X			
Yanggratoke et al. [57]	X			
Duy et al. [58]	X			*[2]
Srikantaiah et al. [59]	X			X
Xu et al. [61]			X	X
Bertini et al. [60]	X			X
Lago	X	X		X

Tabela 3.1: Itens Contemplados Pelos Trabalhos Relacionados

do consumo de energia elétrica através da administração racional de cargas em máquinas físicas, desligando máquinas ociosas, aplicando dimensionamento dinâmico de tensão e frequência nas máquinas em atividade e utilizando eficientemente os mecanismos de controle de refrigeração ativa dos computadores. A estrutura dos *data centers* visada por esse trabalho independe dos tamanhos destes e é baseada em máquinas virtuais, cujas características de migrações são aproveitadas para retirar a carga de servidores subutilizados, melhorando assim a eficiência energética. Também trabalha-se o conceito de qualidade de serviço nos algoritmos propostos, permitindo que cargas de trabalho com maior prioridade sejam executadas em máquinas virtuais mais aptas, reduzindo o tempo de processamento necessário para tais cargas.

Na Tabela 3.1 é apresentada uma síntese das principais diferenças entre este trabalho e os demais citados.

Dasgupta et al. [14] colocam que a normalização de cargas de trabalho em sistemas heterogêneos, como as nuvens, é um desafio a ser abordado. Como será demonstrado nos próximos capítulos o principal ganho com a aplicação dos algoritmos mostrados nesse trabalho são obtidos em *data centers* heterogêneos.

[1] Não usa Fan Control, mas o aquecimento dos nós é verificado através de um monitor de máquinas virtuais, realizando a migração de nós superaquecidos.

[2] A qualidade de serviço aplicada é no sentido de não violar SLAs.

Capítulo 4

Algoritmos Propostos para Alocação de Máquinas Virtuais e Submissão de Cargas de Trabalho

Em uma nuvem, um dos servidores, denominado *broker*, deve determinar quantos e quais *data centers* fazem parte da nuvem e suas características, tais como: servidores, arquitetura, sistema operacional, entre outras. Além disso, o *broker* também deve lidar com eventos, por exemplo: requisições de recursos, criação de máquinas virtuais, retorno do processamento das cargas virtuais e finalização das cargas e máquinas virtuais.

Neste trabalho são propostos dois algoritmos executados pelo *broker*: um que age na alocação de servidores para a criação de máquinas virtuais e, outro, que atua na submissão de cargas para as máquinas virtuais criadas. No primeiro algoritmo o foco é a economia de energia, enquanto o segundo busca adicionar qualidade de serviço ao escalonamento, sem prejudicar o primeiro. Esses algoritmos foram projetados para lidar com o escalonamento em nuvens computacionais constituídas por *data centers* não-federados, e também é considerado que as tecnologias de *DVFS* e *Fan Control* estão ativadas para todos os servidores.

Uma nuvem composta por um *data center* não-federado significa que todos os servidores estão fisicamente em um mesmo *data center*, ou seja, não há que se trabalhar com a comunicação entre *data centers* dispostos em pontos geograficamente distintos. É apresentado na Figura 4.1 um modelo exemplificando este tipo de nuvem. $N1$ é a nuvem e $DC1$ é o *data center* não federado.

Já numa nuvem composta por *data centers* federados, ocorre a interligação entre dois ou mais *data centers* que compõem a nuvem, dispostos em pontos geograficamente distintos. Lidar com este tipo de configuração torna o processo de escalonamento muito mais custoso, uma vez que é necessário conectá-los com uma largura de banda muitas

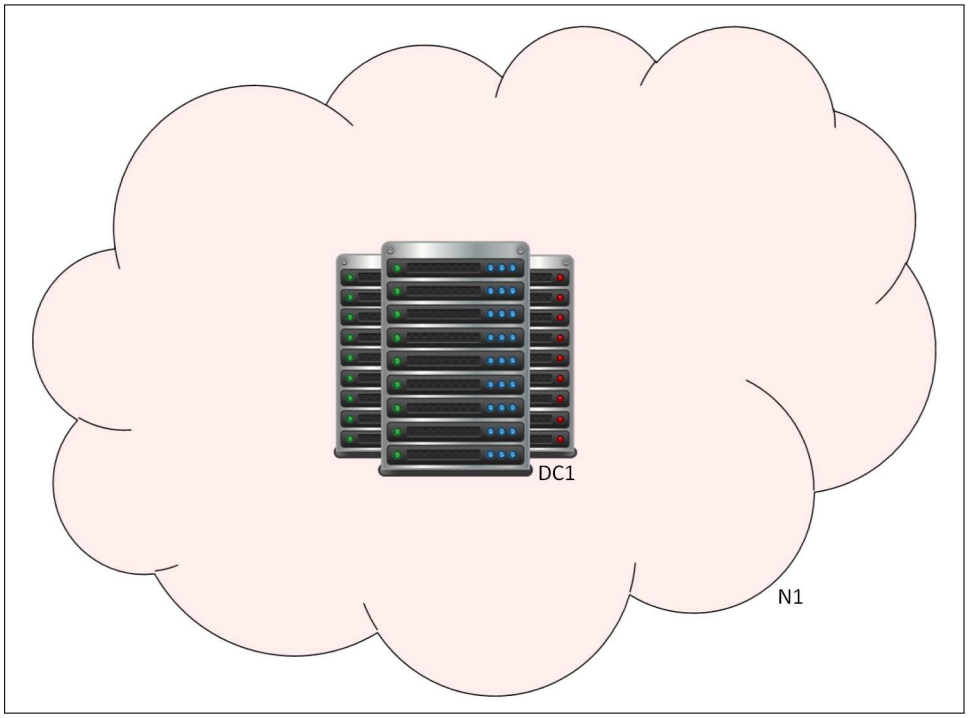


Figura 4.1: Exemplo de Nuvem Composta por Data Center Não Federado

vezes limitada, o que pode acarretar uma série de impedimentos, em especial a maior latência na conclusão da execução de cargas de trabalho por causa da comunicação. É exemplificado na Figura 4.2 uma nuvem $N2$, composta por dois *data centers* $DC1$ e $DC2$, interligados, portanto é uma nuvem composta por *data centers federados*.

O escopo dos algoritmos propostos inclui *data centers* homogêneos e *data centers* heterogêneos.

Em *data centers* homogêneos, todas as máquinas apresentam configurações iguais de hardware. É exemplificado na Figura 4.3 um exemplo de *data center* homogêneo. Este apresenta apenas seis servidores — $H1$, $H2$, $H3$, $H4$, $H5$ e $H6$ — para fins de simplicidade de entendimento, e um *broker* B que atua efetuando, entre outras atividades, a alocação de máquinas virtuais e submissão de cargas para as máquinas virtuais. Note que todos os servidores apresentam a mesma configuração, $c1$.

Em *data centers* heterogêneos, há conjuntos de máquinas que apresentam configurações distintas de hardware. É exemplificado na Figura 4.4 um exemplo de *data center* heterogêneo. Este apresenta apenas seis servidores — $H1$, $H2$, $H3$, $H4$, $H5$ e $H6$ — para fins de simplicidade de entendimento, e um *broker* B. Note que o conjunto de servidores $H1$, $H2$ e $H3$ apresenta configuração $c1$ mas, diferentemente do caso de *data centers homogêneos*, também há o conjunto $H4$ e $H5$ que apresenta configuração $c2$ e o servidor $H6$ que apresenta configuração $c3$.

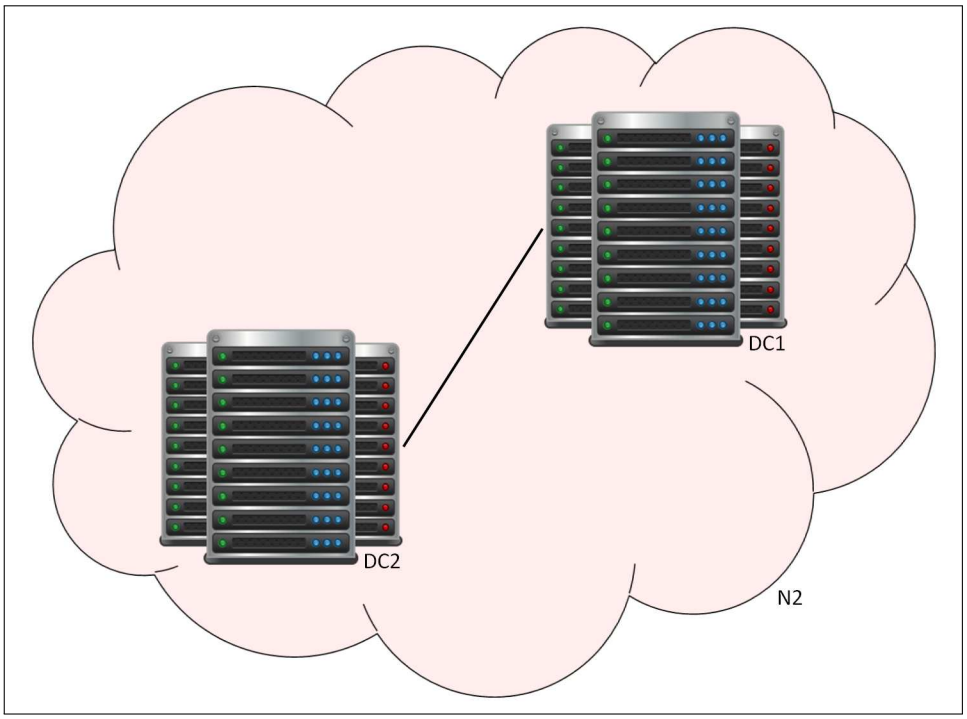


Figura 4.2: Exemplo de Nuvem Composta por Data Centers Federados

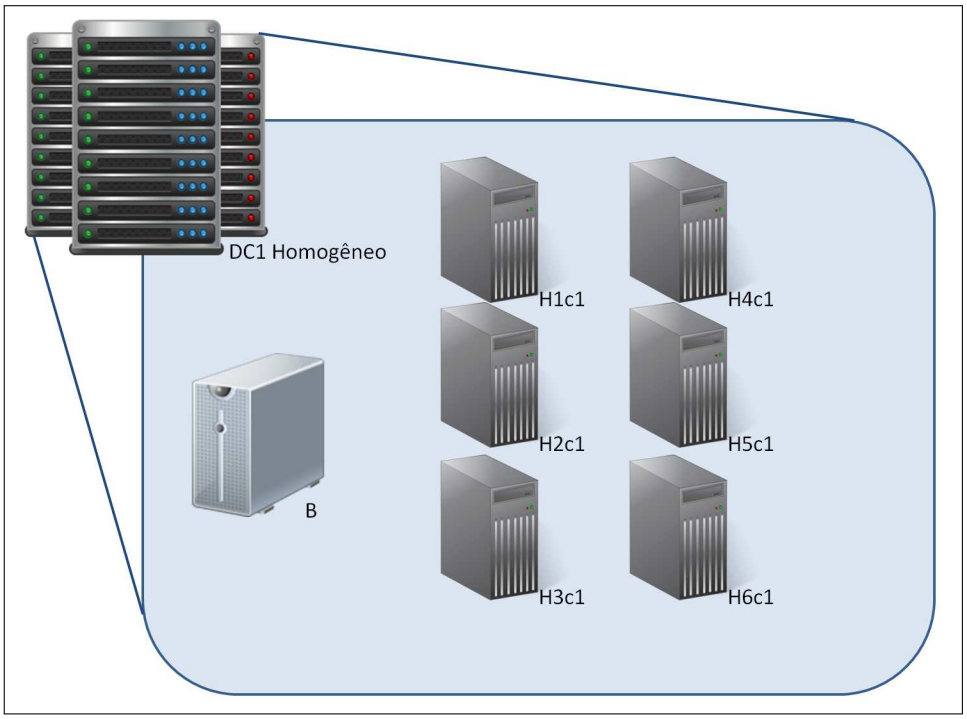


Figura 4.3: Exemplo de Data Center Homogêneo

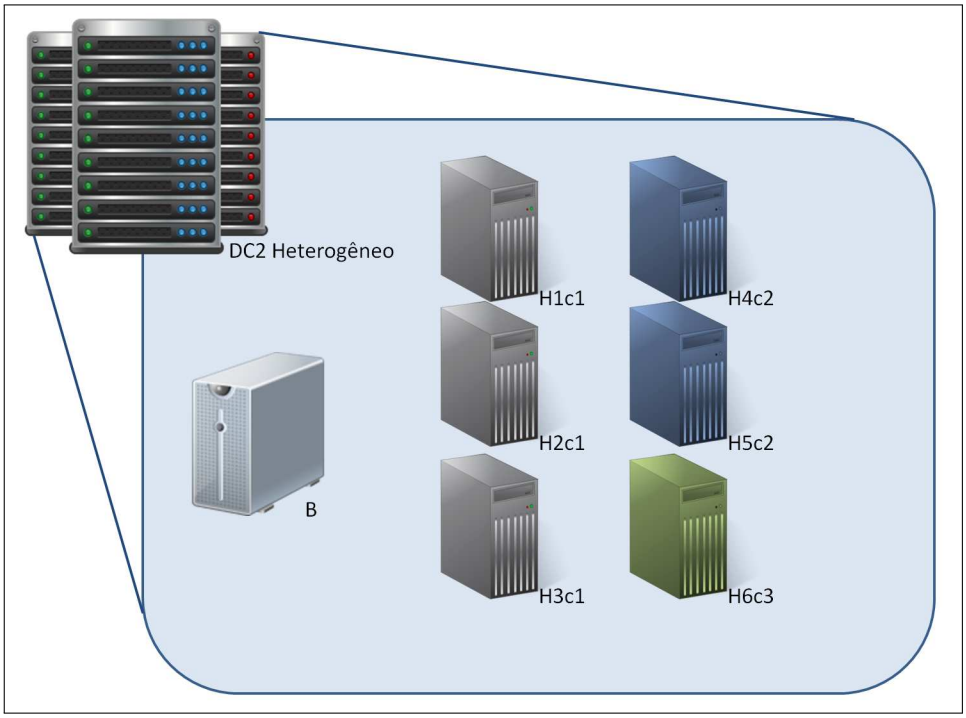


Figura 4.4: Exemplo de Data Center Heterogêneo

Para o escopo também foi definido que não é possível conhecer ou estimar os pesos das cargas de trabalho que serão alocadas nas máquinas virtuais do *data center*. Em decorrência disso os algoritmos foram projetados para se adaptarem dinamicamente às necessidades de processamento, otimizando o consumo de energia *on-the-fly*. Uma vez que todas as máquinas virtuais tenham sido criadas, é iniciado o processo de submissão das cargas.

Não faz parte do escopo dos algoritmos tratar de cenários de falha, onde o escalonador ou demais servidores falham, uma vez que pela complexidade acarretada tornaria o foco do trabalho demasiadamente extenso. Cogita-se em trabalhos futuros analisar e lidar com estes tipos de cenários.

Na Seção 4.1 é apresentado o algoritmo de alocação de máquinas virtuais, cujo objetivo é a minimização do consumo de energia no processo de escalonamento de máquinas virtuais em uma nuvem e, na Seção 4.2, é mostrado o algoritmo para a submissão de cargas de trabalho, cujo objetivo é prover qualidade de serviço através da minimização do tempo de execução das cargas com maiores prioridades.

4.1 Alocação de Máquinas Virtuais

O algoritmo que foi elaborado para a alocação de máquinas virtuais apresentado nesta seção recebeu o nome de *Lago Allocator* (LA) [62]. Este algoritmo é utilizado em conjunto com o algoritmo descrito na Seção 4.2, que atua na submissão de cargas de trabalho para as máquinas virtuais alocadas.

O objetivo deste algoritmo é a minimização de consumo de energia. A ideia fundamental deste algoritmo é a realização de uma série de comparações, necessariamente de baixa complexidade (devido à necessidade de escala do algoritmo), analisando em especial casos de empate e critérios de desempate, e tomando decisões baseadas na melhor forma de consumir a menor quantidade possível de energia.

Este algoritmo atua no momento da alocação de cada máquina virtual, se embasando em dois pilares para minimizar o consumo de energia: cálculo instantâneo do servidor mais apto para receber a máquina virtual em questão, através de comparações de eficiência em energia e de estimativas de consumo de potência após alocação, e minimização das migrações de máquinas virtuais entre servidores, através da verificação daqueles que provavelmente demorarão mais para serem desligados. Com o número de migrações reduzido, há uma tendência de redução também no tempo de processamento das cargas alocadas em tais máquinas, em consequência reduzindo o consumo de energia não só pela diminuição deste tempo de processamento, mas também do custo energético associado à migração das máquinas virtuais. Este custo pode ser calculado através da energia consumida para ligar o servidor destino da migração (caso esteja previamente desligado), somada com a energia consumida pelo servidor destino durante o tempo da migração, somada com o custo de desligamento do servidor de origem da migração, somado com o consumo de energia dos periféricos de rede e hardware durante a migração.

As principais diferenças deste algoritmo para os semelhantes encontrados na literatura é o fato de considerar em seus cálculos a economia que pode ser conseguida pelo mecanismo de *Fan Control* e, especialmente em caso de *data centers* heterogêneos, a forma como ele atua para minimizar a migração de máquinas virtuais. São descritas agora algumas funções usadas pelo algoritmo, seguido pelo algoritmo e na sequência uma explicação detalhada do seu funcionamento.

A função `getMIPS` retorna a capacidade máxima MIPS de uma máquina virtual ou de um servidor.

A função `getCurMIPSUse` retorna a utilização de MIPS atual de um determinado servidor. De modo geral essa função pode ser definida como:

$$\texttt{getCurMIPSUse(host)} = \texttt{getMIPS(host)} - \texttt{MIPS\_Base}_{host} - \sum_{i=1}^{n} \texttt{getMIPS(host\_vm}_i\texttt{)}$$

Onde $\texttt{MIPS\_Base}_{host}$ representa a quantidade de MIPS requerida pelo sistema. Caso o sistema não requeira MIPS, toda a capacidade de processamento desse servidor estará disponível para as máquinas virtuais, pois $\texttt{MIPS\_Base}_{host} = 0$.

A quantidade total de MIPS disponíveis de um servidor pode ser obtida através de `getMIPS(host)` − `getCurMIPSUse(host)`. Para o caso $\texttt{MIPS\_Base}_{host} = 0$, os MIPS disponíveis no servidor serão iguais ao MIPS total do servidor subtraído do somatório de MIPS consumidos pelas máquinas virtuais que estão alocadas; ou seja, $\texttt{MIPS}_{Disponiveis} =$ `getMIPS(host)` $-\sum_{i=1}^{n}$ `getCurMIPSUse(i)`, onde n é o número de máquinas virtuais. Estes exemplos são para os casos mais simples, nos quais cada máquina virtual consome apenas uma entidade de processamento e cada servidor possua uma entidade de processamento. Define-se entidade de processamento como toda a entidade presente em um servidor capaz de realizar processamento útil, sendo tipicamente o processador. Para os casos de máquinas virtuais que usam múltiplos núcleos, ou servidores multi-core, adequações devem ser realizadas para acordar com o cenário abordado.

A função `getMaximumPower` retorna a potência máxima de um servidor (em *watts*), composta por sua potência base (usada pela placa-mãe, discos, RAM, etc.), somada com a potência do processador em uso máximo (com *DVFS* desativado), somado com a potência do dissipador ativo em uso máximo (com *Fan Control* desativado). A função `getPower` retorna o uso atual de energia de um servidor, que é resultado de sua potência base somada com a potência atual do processador (com *DVFS* ativado), somado com a potência atual do dissipador ativo (com *Fan Control* ativado). Uma vez que essa função depende da plataforma, ela deve ser implementada de acordo.

A função `getPowerAfterAllocation` retorna a potência estimada por um servidor após a alocação de uma máquina virtual. `getUtilizationOfCpu` retorna o uso percentual atual do CPU e `getTotalMips` retorna o total de MIPS que uma CPU suporta. Em um modelo de potência linear, a função `getPowerAfterAllocation` pode ser definida a seguir:

$$paa = h_{sp} + (h_{max\_pow} - h_{sp}) \times \frac{MIPS_c + MIPS_{vm}}{MIPS_{max}}$$

Onde *paa* é o consumo de potência após a alocação da máquina virtual, *hsp* é a potência estática do servidor, h_{max_pow} é o consumo máximo de potência do servidor, $MIPS_c$ é o uso atual de MIPS pelo servidor, $MIPS_{vm}$ é o consumo atual de MIPS pelas máquinas virtuais alocadas e $MIPS_{max}$ é o total de MIPS do servidor.

O algoritmo *Lago Allocator* está apresentado no Algoritmo 1.

Para cada máquina virtual a ser alocada no *data center*, são verificados, entre todos os servidores, quais são candidatos para receber a nova máquina virtual. A primeira condição para um servidor poder receber uma máquina virtual submetida é possuir MIPS disponíveis suficientes para suportar a máquina virtual a ser alocada (linha 5). Na Figura 4.5 é exemplificado este primeiro passo. Deseja-se alocar a máquina virtual $VM1$ no

Algoritmo 1 Lago Allocator: findHostForVm(vm)

```
1: bestEfficiency ← ∞
2: bestHost ← null
3: for all hosts in data center do
4:   utilization ← getCurMIPSUse(host) + getMIPS(vm)
5:   if utilization < getMIPS(host) then
6:     efficiency ← getMIPS(host) / getMaximumPower(host)
7:     if efficiency > bestEfficiency then
8:       bestEfficiency ← efficiency
9:       bestHost ← host
10:    else if efficiency = bestEfficiency then
11:      pw_vm_at_host ← getPower(bestHost)+getPowerAfterAllocation(host, vm)
12:      pw_vm_at_bestHost ← getPower(host) + getPowerAfterAllocation(bestHost, vm)
13:      if pw_vm_at_host < pw_vm_at_bestHost then
14:        bestHost ← host
15:      else if pw_vm_at_host = pw_vm_at_bestHost then
16:        if getUtilizationOfCpu(host) > getUtilizationOfCpu(bestHost) then
17:          bestHost ← host
18:        else if getUtilizationOfCpu(host) = getUtilizationOfCpu(bestHost) then
19:          if getTotalMips(host) > getTotalMips(allocatedHost) then
20:            bestHost ← host
21:          end if
22:        end if
23:      end if
24:    end if
25:  end if
26: end for
27: return bestHost
```

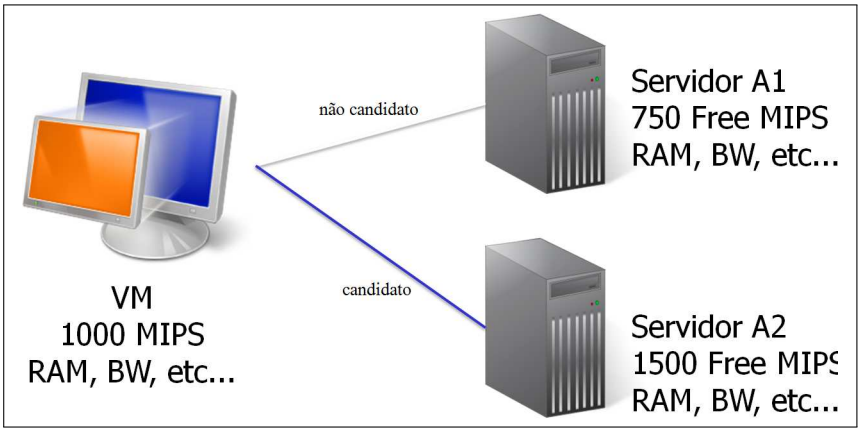


Figura 4.5: Seleção de Servidores Candidatos

data center da nuvem. Esta máquina virtual requer 1000 MIPS, alguma quantidade de RAM e de largura de banda que, para simplicidade de entendimento, serão considerados desprezíveis. Supõe-se existir no *data center* que compõe esta nuvem os servidores $A1$ e $A2$. Independentemente das configurações, o servidor $A1$ possui disponível 750 MIPS — pode ser que ele possua sua capacidade total de 750 MIPS e não alocou nenhuma máquina virtual, ou que já tenha alocado previamente algumas máquinas virtuais e neste instante esteja com essa capacidade disponível. Esta capacidade é insuficiente para alocar a $VM1$, portanto este servidor não será candidato. Já o outro servidor mencionado, $A2$, com 1500 MIPS disponíveis, pode receber $VM1$, portanto é selecionado como um servidor candidato.

O algoritmo então seleciona o melhor dentre os servidores candidatos, adotando como parâmetro a eficiência em energia (linha 7). Eficiência em energia, neste contexto, se refere à razão entre as milhões de instruções por segundo (MIPS) que um servidor pode processar e seu consumo de energia ($EE = \frac{MIPS}{P}$, onde $MIPS$ são as milhões de instruções por segundo e P é a potência, medida em *watts*). Considere que se deseja alocar uma máquina virtual pelo *broker* e considerando os servidores candidatos $B1$, $B2$ e $B3$ do cenário apresentado na Figura 4.6. Calcula-se então a eficiência energética de cada um. A eficiência em energia de $B1$ é dada por $\frac{1000}{150} = 6,67$, de $B2$ é $\frac{1500}{200} = 7,5$ e de $B3$ é $\frac{2000}{350} = 5,7$. Por ser $B2$ o que possui a maior eficiência em energia, ele é escolhido para a alocação da máquina virtual.

Um empate pode ocorrer, especialmente em *data centers* homogêneos. Um critério de desempate é feito da seguinte maneira: tendo em vista que os servidores contam com *DVFS* ativado, assim como *Fan Control*, provavelmente existirá alguns servidores consumindo menos energia do que outros. Então os servidores são confrontados entre si e o que consumir menor potência com a alocação da máquina virtual será escolhido

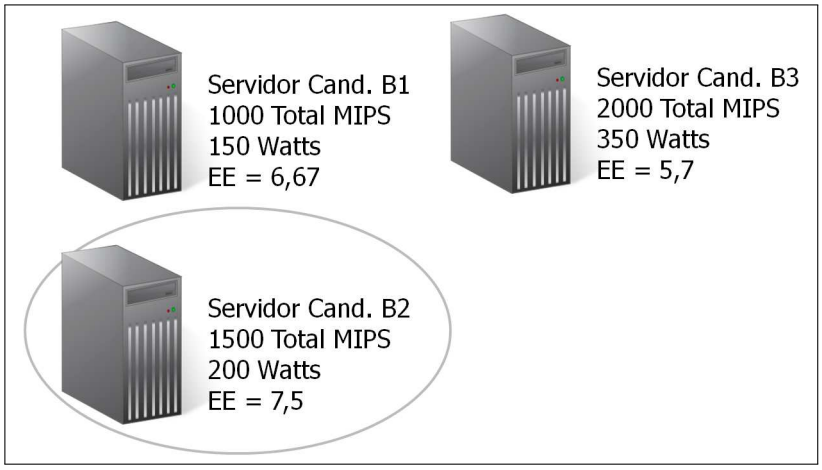


Figura 4.6: Cálculo de Eficiência em Energia

(linha 13). No cenário apresentado na Figura 4.7, são dados três servidores, $C1$, $C2$ e $C3$, dentre os quais se deseja alocar uma máquina virtual $VM2$, candidatos a $VM2$ e com igual eficiência em energia. É calculado que após a alocação de $VM2$ o servidor $C1$ consumirá 125 *watts* após a alocação, $C2$ por sua vez 129 *watts* e $C3$ 128 *watts*. $C1$, por ser o servidor que consumirá menor potência após a alocação de $VM2$, é escolhido.

Para exemplificar de forma simplificada como uma estimativa de potência consumida após a alocação de uma máquina virtual pode ser feita, apresentamos o seguinte caso hipotético: considere um processador cuja potência consumida em operação máxima é de 130 *watts*, e que este opere com DVFS ativado. Suponha que este processador tenha capacidade máxima de 3 *GHz*. Observando a tabela de *P-States* deste processador verificamos a existência de 4 estados: *P0*, onde este processador trabalha a 3 *GHz* e consome 130 *watts*; no estado *P1* estes valores são 2,5 *GHz* e 100 *watts*; no estado *P2* têm-se 2 *GHz* e 75 *watts*; e o estado *P3* (o mais econômico) apresenta um consumo de 55 *watts* para uma frequência de 1,5 *GHz*. Isso quer dizer que enquanto a capacidade de processamento requisitada estiver abaixo de 1,5 *GHz*, este processador permanecerá no estado *P3*, consumindo 55 *watts* e, quando ultrapassar este limiar, e estiver operando abaixo de 2 *GHz*, o consumo será de 75 *watts*, e assim sucessivamente. Sabe-se que o servidor no qual este processador está instalado consome, em carga máxima, 250 *watts*, e que é desprezível a variação do consumo de seus componentes, exceto o processador. Ou seja, para calcular o consumo de potência deste servidor sem o processador subtraímos da potência máxima de operação do sistema, de 250 *watts*, a potência máxima do processador, de 130 *watts*, e verificamos que sem o processador este sistema consome $250 - 130 = 120$ *watts* para se manter ligado. Ou seja, inicialmente este sistema, com o processador operando em capacidade mínima, consome $120 + 55 = 175$ *watts*. Considere que tal processador realiza uma operação por ciclo (implicando que este processador possui 3.000 MIPS). Ao alocarmos

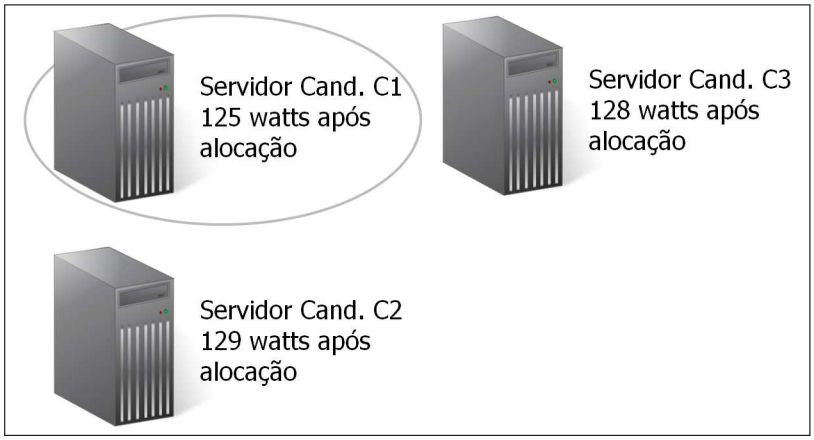


Figura 4.7: Cálculo de Consumo após Alocação de Máquina Virtual

uma máquina virtual de 800 MIPS no sistema, este passará a consumir 800 MIPS. Como não foi ultrapassado o limiar de *P3*, o consumo do sistema permanece o mesmo. Ao alocar mais uma máquina virtual de igual capacidade, a utilização do processador passará a ser 1600 MIPS, alterando seu estado de *P3* para *P2*, implicando em um consumo de potência do sistema de $120 + 75 = 195$ *watts*. Se alocarmos uma terceira máquina virtual de 800 MIPS neste sistema, ele passará a necessitar de 2400 MIPS, alterando seu estado para *P1* e consumindo $120 + 100 = 220$ *watts*.

Ainda que o critério de desempate apresentado seja aplicado, poderá ocorrer um novo empate, por exemplo, se servidores de iguais consumos de potência ainda não tiverem recebido nenhuma máquina virtual. Outro critério de desempate então é realizado: o servidor com a maior utilização de CPU é escolhido. Essa escolha é baseada no fato que se uma CPU estiver processando maior carga, o servidor provavelmente demorará mais tempo para atingir seu limiar para migração de máquinas virtuais (linha 16). É mostrado na Figura 4.8 um cenário onde se deseja alocar uma máquina virtual $VM3$, existindo três servidores, $D1$, $D2$ e $D3$, candidatos a $VM3$, todos apresentando igual eficiência em energia e consumo após a alocação de $VM3$. Note, no entanto, que eles apresentam carga de CPU distintas. $D3$ é então escolhido para a alocação $VM3$, uma vez que é aquele que está com maior carga de trabalho, e por isso é provavelmente aquele que levará maior tempo para atingir seu limiar para migração das máquinas virtuais por ele processadas.

Se ainda ocorrer um empate o servidor com maior poder de processamento será escolhido, pois ele é potencialmente mais hábil para receber maior número de cargas (linha 19). Na circunstância de outro empate o primeiro servidor disponível será escolhido. Um cenário é apresentado na Figura 4.9, composto pelos servidores $E1$, $E2$ e $E3$, candidatos a alocação da máquina virtual $VM4$ e empatados em todos os critérios citados anteriormente. Neste caso o servidor candidato $E2$ é o escolhido para a alocação de $VM4$, uma

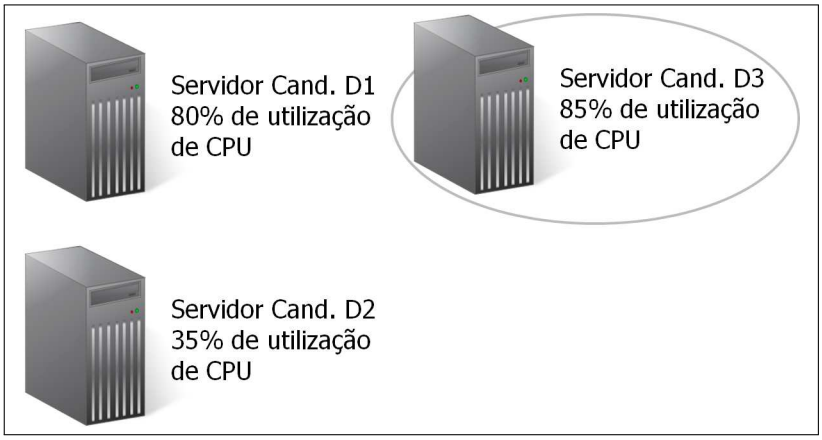


Figura 4.8: Cálculo de Carga de Processamento

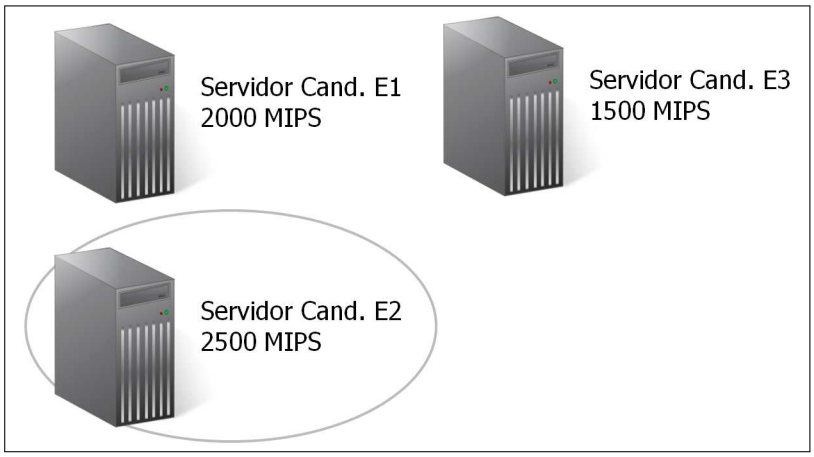


Figura 4.9: Escolha por Maior MIPS

vez que é o que possui maior capacidade de processamento e, potencialmente, aquele que suportará maior número de máquinas virtuais/cargas de trabalho.

Com todas as máquinas virtuais criadas, ocorre a submissão das cargas para processamento. É apresentado na Seção 4.2 um algoritmo para esse processo.

Deste ponto em diante o algoritmo passa a checar quais servidores estão com uso abaixo de um limiar de processamento definido para migrar as cargas virtuais desses servidores, seguido do seu desligamento. Do ponto de vista prático essa verificação é realizada em intervalos definidos e fixos de tempo. Essas informações podem ser fornecidas diretamente pelos servidores, uma vez que eles têm as especificações de suas capacidades máximas e das máquinas virtuais alocadas. Outra alternativa para efetuar essa verificação é armazenar na máquina que efetua o escalonamento um banco de dados contendo as capacidades de processamento, RAM, etc. dos servidores, sendo a verificação realizada conforme rea-

lização da submissão e finalização das máquinas virtuais, sem a necessidade de consultar os servidores. A escolha dos servidores destinos para os quais as máquinas virtuais dos servidores subutilizados deverão ser migradas é realizada pelo próprio Algoritmo 1.

O valor ótimo para o limiar de migração pode ser difícil de calcular. Há pesquisas acadêmicas que procuram o melhor caminho para calcular esse limiar. Mukherjee e Sahoo [63], por exemplo, propõem o uso de um sistema de colônia de abelhas para calcular bons limiares para migração das cargas virtuais. Como trabalho futuro é considerada a adição de técnicas como essa para implementar limiares adaptativos para este algoritmo.

O fato de realizar cálculos de eficiência em energia e de potência consumida, após a alocação de cada máquina virtual, implica que o algoritmo avalia se compensa concentrar maior número de máquinas virtuais/cargas de trabalho em somente um servidor, ou se compensa distribuir as máquinas virtuais entre vários servidores.

Caso apenas este algoritmo seja utilizado no processo de escalonamento em nuvem, haverá a minimização do consumo de energia, mas sem nenhum tipo de qualidade de serviço. É proposto, a seguir, um algoritmo que permite a adição de qualidade de serviço com base em prioridade de execução de cargas virtuais, o que é desejável neste contexto.

4.2 Submissão de Cargas de Trabalho

Nesta seção é explicado o algoritmo proposto que age na submissão de cargas de trabalho para as máquinas virtuais previamente criadas [64]. O objetivo deste algoritmo é a adição de qualidade de serviço ao escalonamento em nuvens. A qualidade de serviço proposta é baseada na adição de prioridades às cargas que serão processadas pelas máquinas virtuais. Tais prioridades são definidas pelos usuários da nuvem, e podem assumir vários níveis, por exemplo: alta, média e baixa. O objetivo da qualidade de serviço proposta é a minimização do tempo de execução das cargas que possuírem maior prioridade.

A ideia por trás deste algoritmo é que ele deve ser simples (devido à necessidade de escala), deve ser benéfico no sentido de tornar mais rápida a execução de cargas com alta prioridade e, principalmente, deve impactar minimamente o consumo de energia, já que normalmente a adição de qualidade de serviço em um processo de escalonamento está associada ao aumento no consumo de energia. De modo geral, este algoritmo verifica, em todo o universo de cargas para submissão, quais cargas possuem maior prioridade, submetendo-as às máquinas virtuais, repetindo este procedimento enquanto houver cargas para ser submetidas.

Este algoritmo atua nas cargas de trabalho que não possuem vínculo com alguma máquina virtual específica, ou seja, ele pode determinar em qual máquina virtual cada carga de trabalho será alocada. Faz parte do objetivo que a adição da qualidade de serviço traga vantagens, impactando o mínimo possível no consumo de energia, no sentido de não

aumentá-lo. É descrito a seguir algumas funções usadas pelo algoritmo de submissão de cargas.

A função `sort_decreasing_by_free_MIPS(vmList)` ordena um vetor `vmList` contendo objetos que representam máquinas virtuais em ordem decrescente de MIPS disponíveis; `getPriority(c)` retorna a prioridade de uma carga de trabalho c. A função `send(c, vm)` encaminha uma carga de trabalho c para ser processada em uma máquina virtual vm, ou seja, o *broker* envia através da rede a carga c para a máquina virtual previamente instanciada vm que, após o recebimento, inicia o processamento. A função `resort_decreasing_by_free_MIPS(vmList, i)` reordena a máquina virtual no índice i de uma lista de máquinas virtuais `vmList` em ordem decrescente de MIPS disponíveis. As constantes `MAX_PRIORITY` e `MIN_PRIORITY` são valores inteiros que representam as prioridades máxima e mínima possíveis para uma carga virtual qualquer.

É apresentada no Algoritmo 2 uma versão geral desse algoritmo. No Algoritmo 3 é apresentada uma versão otimizada para o caso em que o número de máquinas virtuais é igual ao de cargas de trabalho. Assim como no *CloudSim* [15], uma *cloudlet* é definida como uma tarefa submetida às máquinas virtuais do *data center*.

Algoritmo 2 Lago Allocator: submitCloudlets(vm)

```
1: Vm[] vmList {Lista contendo as máquinas virtuais que podem ser alocadas para cloudlets}
2: Cloudlet[] cloudletList {Lista de cloudlets que deverão ser alocadas nas máquinas virtuais}
3:
4: sort_decreasing_by_free_MIPS(vmList)
5: for current_priority := MAX_PRIORITY → MIN_PRIORITY do
6:   for all cloudletsincloudletList do
7:     if getPriority(cloudlet) = current_priority then
8:       send(cloudlet, vmList[0])
9:       resort_decreasing_by_free_MIPS(vmList, 0)
10:    end if
11:  end for
12: end for
```

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

Se em um determinado instante não existirem máquinas virtuais aptas para receberem novas cargas de trabalho, estas serão postergadas. Note que o Algoritmo 2 funciona, mas ele pode ser otimizado.

Note também que a função `resort_decreasing_by_free_MIPS()` encontra-se aninhada numa varredura de todas as *cloudlets*, implicando que o algoritmo tenha complexidade $O(p \times n \times n \log n) = O(p \times n^2 \times \log n)$, onde p é o número de prioridades e n é o número de *cloudlets* e supondo um algoritmo de ordenação de complexidade $O(n \times \log n)$. Observando nuvens nas quais cada carga de trabalho é processada por uma máquina virtual distinta, ou seja, o número de cargas é igual ao de máquinas virtuais, pode-se remover a custosa função `resort_decreasing_by_free_MIPS()`, conforme descrito no Algoritmo 3.

Algoritmo 3 Lago Allocator: submitCloudletsEnhanced(vm)

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

Haverá a postergação da submissão de cargas caso não existam máquinas virtuais disponíveis para o processamento destas durante as iterações. Com todas as cargas submetidas, o *broker* passará a checar os servidores e desligar aqueles em que todas as máquinas virtuais tenham terminado de processar suas cargas, além de verificar quais servidores estão sendo utilizados abaixo de um limiar predefinido para migrar as máquinas virtuais destes e desligá-los.

Para exemplificar o funcionamento deste algoritmo, são apresentadas na Figura 4.10 quatro máquinas virtuais, $VM1$, $VM2$, $VM3$ e $VM4$, devidamente instanciadas pelo Algoritmo 1. A $VM1$ apresenta capacidade de processamento de 750 MIPS, além de RAM e largura de banda que serão considerados desprezíveis, para simplicidade de entendimento. De igual maneira $VM2$ apresenta 1500 MIPS, $VM3$ possui 1000 MIPS e $VM4$ possui

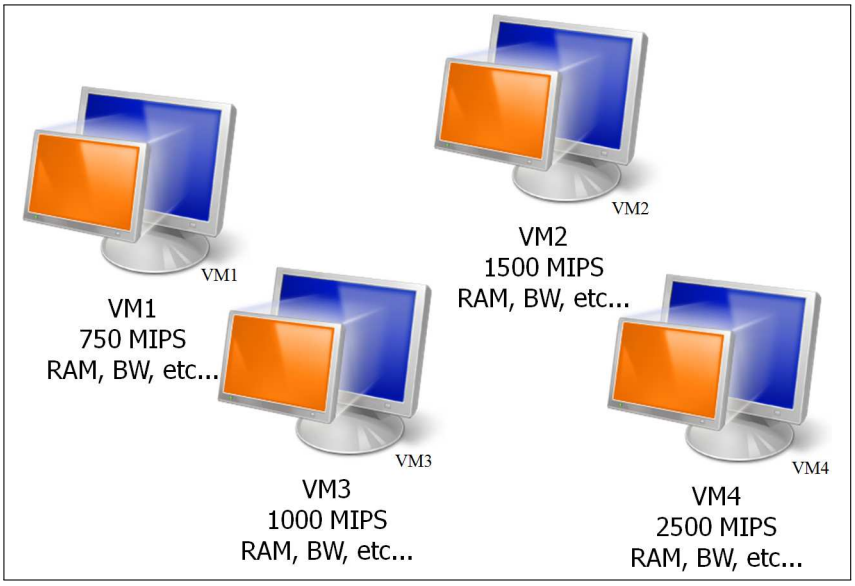


Figura 4.10: Máquinas Virtuais Instanciadas

2500 MIPS.

Suponha que em um dado momento deseja-se submeter quatro cargas de trabalho, $C1$, $C2$, $C3$ e $C4$, para as máquinas virtuais instanciadas, conforme mostrado na Figura 4.11. Suponha, ainda, a existência de três níveis de prioridade: *alta*, *média* e *baixa*. A carga $C1$ apresenta 100.000 MI (milhões de instruções) para serem processadas, requer prioridade alta, e apresenta requisitos de RAM e largura de banda desprezíveis, para simplicidade de entendimento. De igual forma $C2$ contém 100.000 milhões de instruções para serem processadas, sendo esta carga de prioridade média. $C3$ possui 200.000 milhões de instruções e necessita de ser processada rapidamente, por isso requer prioridade alta. Também existe $C4$, com 100.000 MI para processar e não exige urgência, por isto sua prioridade foi escolhida como baixa.

Desenvolvendo o cenário apresentado, o algoritmo escolhe inicialmente as cargas que possuem maior prioridade, conforme indicado na Figura 4.12. Neste caso as cargas escolhidas para serem submetidas são $C1$ e $C3$, por possuírem prioridade "alta". Note que como $C1$ está em primeiro na lista ela deverá ser submetida antes de $C3$.

Conforme visto na Figura 4.13 o algoritmo inicia a submissão das cargas. A primeira carga submetida então é $C1$. Inicialmente o algoritmo efetua a ordenação decrescente por MIPS das máquinas virtuais, obtendo, portanto, a ordem $VM4$, $VM2$, $VM3$, $VM1$. A carga $C1$ então é submetida à primeira máquina virtual da lista, ou seja, $VM4$ recebe $C1$ e passa a processá-la.

Tendo a carga $C1$ sido submetida, a próxima dentre as cargas com igual prioridade (alta) é enviada, no caso, $C3$. Como a máquina virtual $VM4$ já foi utilizada para o

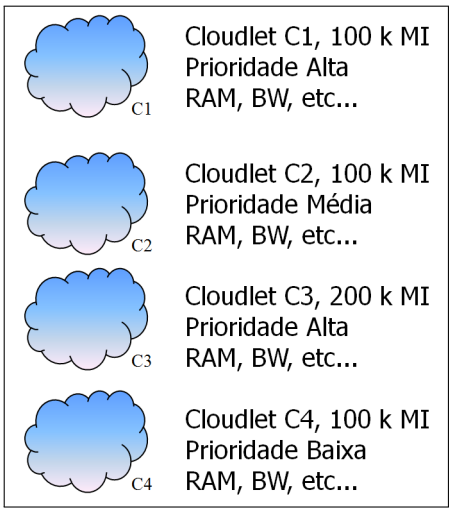


Figura 4.11: Cloudlets para Submissão

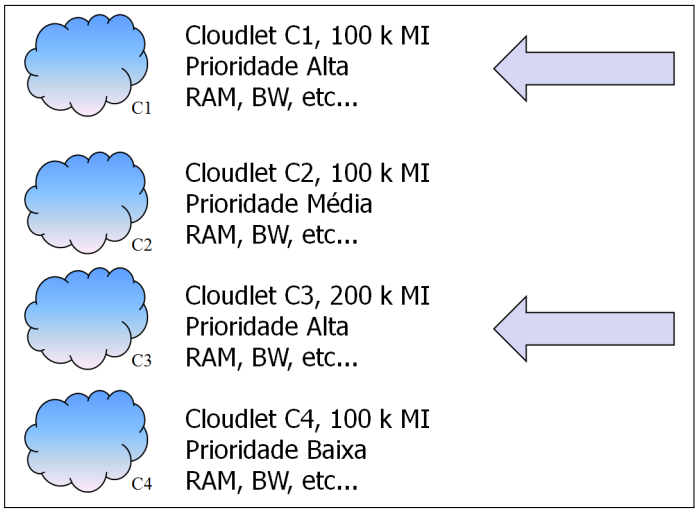


Figura 4.12: Cloudlets com Prioridade Alta para Submissão

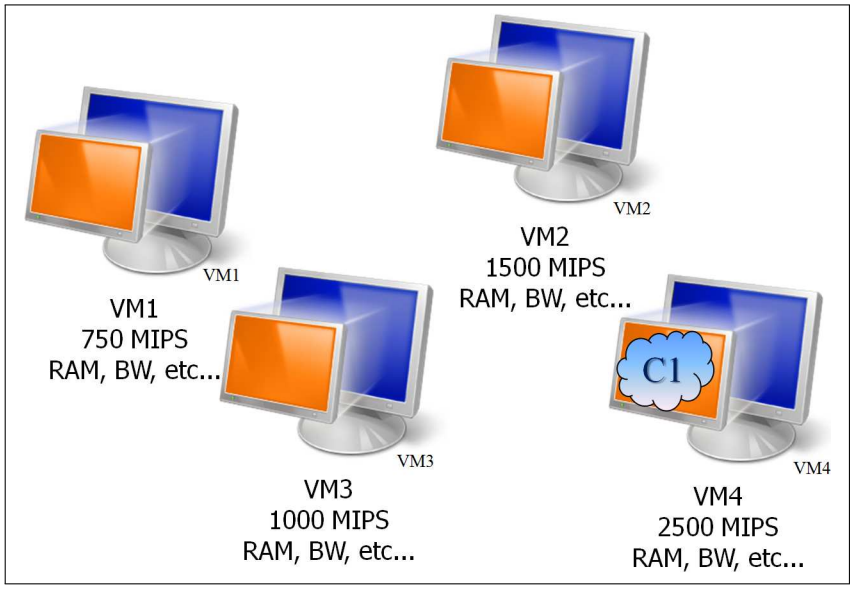


Figura 4.13: Cloudlet C1 Submetida

processamento de $C1$, a próxima é escolhida, no caso, $VM2$. Então, como visto na Figura 4.14, $C3$ é submetido para $VM2$. Note que $C3$ possui 200.000 MI, enquanto $C1$ possui apenas 100.000 MI. Note, ainda, que $VM4$ apresenta 2.500 MIPS, enquanto $VM2$ apresenta somente 1.500 MIPS. É perceptível que $VM4$ processará muito mais rapidamente $C1$ do que $VM2$ processará $C3$. Uma otimização neste caso seria possível, invertendo as submissões, ou seja, enviando $C1$ para $VM2$ e $C3$ para $VM4$. No entanto, como o algoritmo proposto deve tratar casos onde os tamanhos das cargas de trabalho são desconhecidos e/ou inestimáveis, tal otimização é impossível de ser realizada. Uma possível ideia para tentar contornar tal impossibilidade é dar ao usuário o poder para tentar estimar o "peso" das cargas que serão processadas, de modo a submeter antes as cargas (que apresentarem mesma prioridade) de maior peso, aproveitando melhor as máquinas virtuais mais potentes, reduzindo o tempo global de processamento da soma de todas as cargas. Ainda que tais estimativas sejam grosseiras, o tempo de processamento ainda assim tende a ser menor.

Uma vez que todas as cargas com prioridade alta foram devidamente submetidas, o algoritmo entra em nova iteração, reduzindo a prioridade das cargas pelas quais será feita a varredura. Na Figura 4.15 é apresentado o desenvolvimento do algoritmo do cenário anterior. Neste caso as cargas com prioridade "normal" são selecionadas, ou seja, a carga $C2$ entra na lista para ser submetida.

Repetindo os passos mostrados anteriormente, a máquina virtual com maior capacidade de processamento entre as disponíveis na lista é selecionada, ou seja, como $VM2$ e $VM4$ já foram utilizadas, é selecionada $VM3$ para receber $C2$. A submissão então é

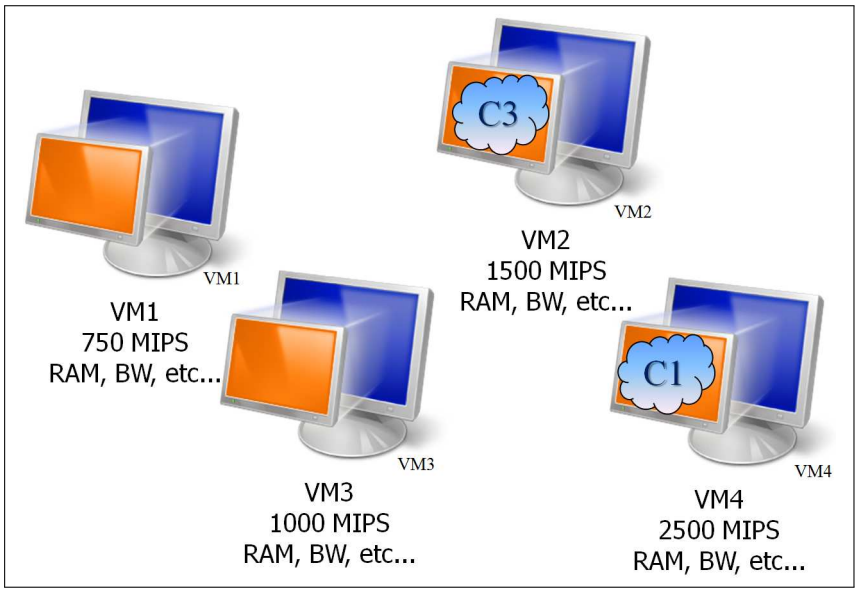


Figura 4.14: Cloudlet C3 Submetida

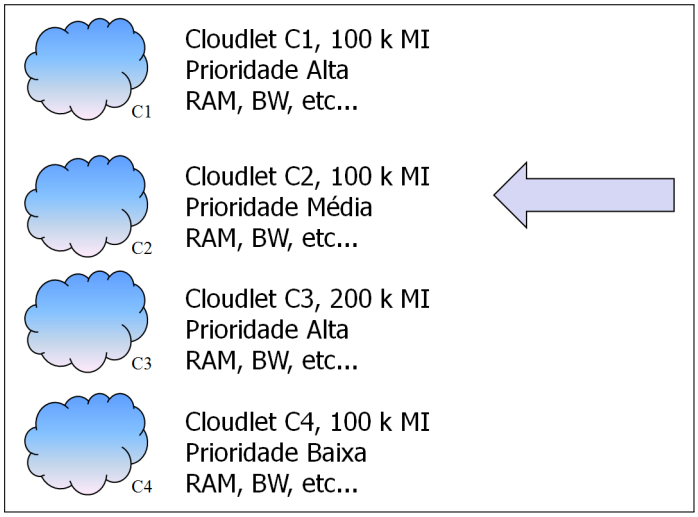


Figura 4.15: Cloudlets com Prioridade Média para Submissão

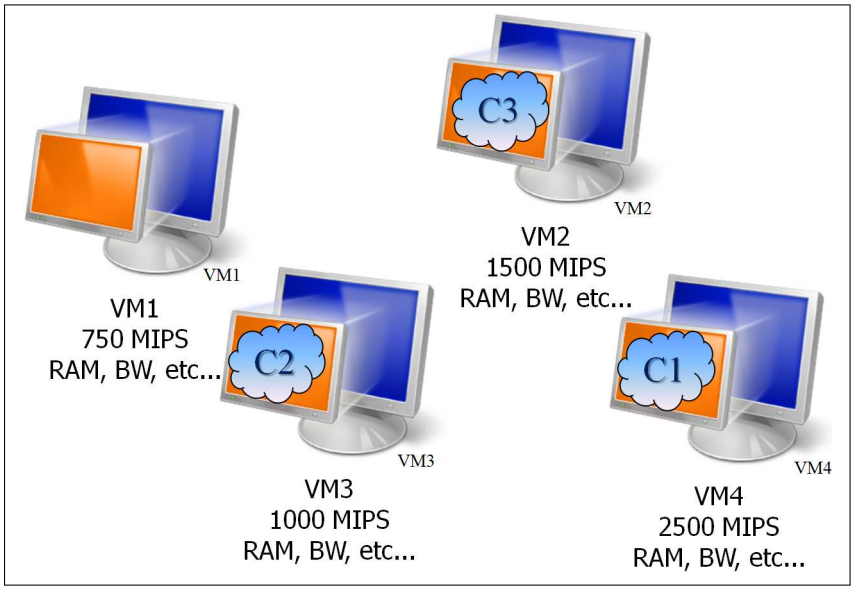


Figura 4.16: Cloudlet C2 Submetida

efetuada, conforme mostrado na Figura 4.16.

Com o fim da lista das cargas de prioridade média finalizada, pois todas foram submetidas, o algoritmo entra em nova iteração. Na Figura 4.17, que apresenta a continuação da execução do algoritmo, percebe-se que agora ele passa a varrer as cargas com prioridade "baixa", neste caso, $C4$ é escolhida para submissão.

$C4$ é, finalmente, submetida, conforme visto na Figura 4.18. Como não há mais cargas para processar em espera, o algoritmo então passa a aguardar a chegada de novas cargas para submissão.

Futuras referências, neste trabalho, para o LA executado em conjunto com este algoritmo serão chamadas de *Lago Allocator* com Qualidade de Serviço ($LA - Q$).

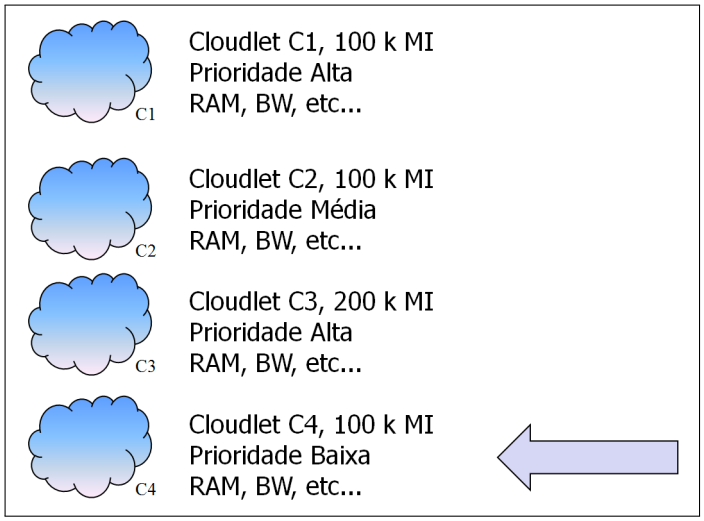


Figura 4.17: Cloudlets com Prioridade Baixa para Submissão

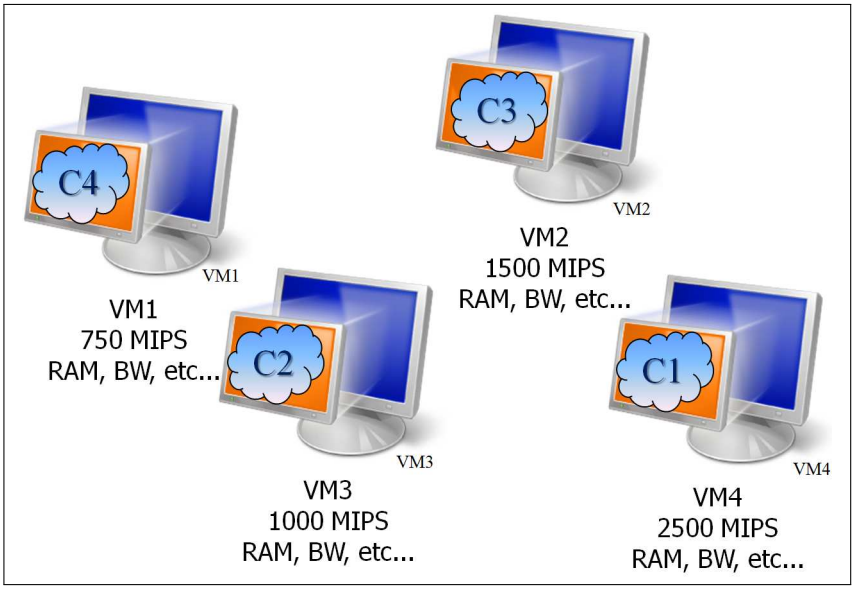


Figura 4.18: Cloudlet C4 Submetida

Capítulo 5

Simulações e Análise de Resultados

A simulação foi escolhida como método de validação dos algoritmos propostos. A implantação real dos algoritmos, embora desejável, é extremamente difícil, dada a infraestrutura necessária para sua avaliação. Por outro lado se um simulador cientificamente reconhecido for utilizado, é garantida confiabilidade aos resultados apresentados. Neste capítulo é apresentado como foi realizada a validação dos algoritmos através da simulação.

5.1 CloudSim

Para simulações eficientes, que analisam diversos cenários, é fundamental a escolha de um simulador robusto. O *CloudSim* [15] é um simulador relevante para computação em nuvem, reconhecido academicamente com citações em centenas de trabalhos publicados. Ele permite elaborar simulações de ambientes de computação em nuvem e avaliar algoritmos de aprovisionamento de recursos, além de efetuar análises de duração de simulações e consumo de energia. Por esses motivos foi constatado que o *CloudSim* é capaz de suprir as necessidades desse trabalho e, portanto, foi escolhido como simulador.

O CloudSim foi construído baseado no *GridSim* [65], outro simulador reconhecido academicamente com citações em milhares de trabalhos publicados, tendo sido desenvolvido em 2002 pelo laboratório CLOUDS da Universidade de Melbourne. Observando isto, e que o *GridSim* já é desenvolvido há 10 anos (desde 2002), pode-se dizer que se trata de um simulador maduro o suficiente para ser considerado confiável para os objetivos deste trabalho.

Para tornar as simulações possíveis, duas extensões, além dos algoritmos, foram desenvolvidas para o simulador. Uma extensão criada foi o modelo de potência que suporte *Fan Control*. Esta estende a classe de modelo de potência linear, fazendo as devidas adequações para suportar *Fan Control*. A forma como este mecanismo atuará nas simulações é explicada detalhadamente na Subseção 5.4.1. Outra extensão que foi necessária de-

senvolver foi a de suporte de qualidade de serviço para *cloudlets*. Nesta, foi elaborada uma classe que estende a classe de *cloudlets* e outra que estende a classe de *broker*. Na primeira foi adicionado o suporte a prioridades, permitindo definir e obter as prioridades das *cloudlets* criadas para a simulação. Na segunda, foi adicionado no *broker* o suporte para reconhecimento dos níveis de prioridade das *cloudlets* e a submissão das cargas de forma a satisfazer a qualidade especificada pelo cenário de simulação.

5.2 Regras de Simulações

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

Para realizar comparações apropriadas entre os algoritmos, algumas definições são apresentadas:

- $A \rightarrow$ representa um algoritmo de escalonamento para computação em nuvem;
- $C \rightarrow$ representa um conjunto de configurações usadas por A. Pode ser: desligamento de servidores subutilizados, migração de cargas virtuais, *DVFS*, *Fan Control* e qualidade de serviço;
- $P \rightarrow$ representa um conjunto de parâmetros de simulação. Composto de: número de servidores, número de máquinas virtuais e milhões de instruções que cada *cloudlet* executa;
- $EC(A, C, P) \rightarrow$ é o consumo médio de energia do algoritmo A com um conjunto de configurações C e parâmetros P;
- $ECI(A, C, P) \rightarrow$ é o intervalo de confiança em 95% de $EC(A, C, P)$;
- $TC(A, C, P) \rightarrow$ é o tempo médio para realização de todas as tarefas (*makespan*) do algoritmo A com o conjunto de configurações C e parâmetros P;
- $TCI(A, C, P) \rightarrow$ é o intervalo de confiança em 95% de $TC(A, C, P)$.

Um escalonamento realizado por um algoritmo A_1, com configurações C_1 e parâmetros de simulação P, é considerado melhor em consumo de energia do que o escalonamento realizado por um algoritmo A_2, com configurações C_2 e parâmetros P, sendo $A_1 \neq A_2$, se a desigualdade mostrada em 5.1 for verdadeira.

$$EC(A_1, C_1, P) + ECI(A_1, C_1, P) < EC(A_2, C_2, P) - ECI(A_2, C_2, P) \qquad (5.1)$$

O escalonamento de um algoritmo A_1 é considerado intoleravelmente mais lento do que A_2 para os parâmetros de simulação P se, para um mesmo conjunto de configurações C, a desigualdade 5.2 for satisfeita.

$$TC(A_1, C, P) + TCI(A_1, C, P) > 2 \times (TC(A_2, C, P) + TCI(A_2, C, P)) \quad (5.2)$$

5.3 Algoritmos e Configurações para Comparação

Para validar os algoritmos propostos, os resultados de suas simulações foram comparados aos resultados de outros algoritmos clássicos da literatura — *Round Robin* e *Best Resource Selection* — e com o algoritmo com foco em gerenciamento de energia encontrado no *CloudSim* — que será denotado como *Minimum Power Diff*. Abaixo há uma descrição simplificada de cada algoritmo que faz parte das simulações:

- *Round Robin* → Cada máquina virtual é alocada em um servidor diferente, realizando ciclos quando todos os servidores tiverem alocado igual número de máquinas virtuais. Servidores que não podem alocar mais máquinas virtuais são saltados. Caso não haja mais servidores capazes de receber instanciações de máquinas virtuais, ocorre a postergação destas instanciações.

- *Best Resource Selection* → O servidor que possuir o processador com a maior razão $\frac{MIPS\_em\_uso}{MIPS\_total}$, sendo este processador capaz de suportar a máquina virtual submetida pelo escalonador, é alocado. Isso garante a minimização das migrações de máquinas virtuais e tende a produzir resultados mais rapidamente.

- *Minimum Power Diff* [15] → Para cada máquina virtual submetida ao escalonador, é escolhido, entre todos os servidores candidatos, aquele que ao receber a máquina virtual terá a menor diferença $PCAposA - PCAntesA$, onde $PCAposA$ é a potência consumida pelo servidor após a alocação da máquina virtual e $PCAntesA$ é a potência consumida pelo servidor antes da alocação.

- *Lago Allocator* → Nome do algoritmo alocador de servidores para máquinas virtuais proposto nesse trabalho.

Os algoritmos apresentados foram implementados com diferentes combinações dos seguintes recursos:

- *Power Aware* → Habilidade de desligar servidores não utilizados.

- *DVFS* 70% → Dimensionamento Dinâmico de Tensão e Frequência. Um modelo linear foi usado para os *P-States*, de modo que servidores não utilizados consomem 70% de sua potência, e servidores operando em capacidade máxima consumirão 100% de sua potência.

- *Threshold* 80% → Migração de cargas virtuais, tentando manter as entidades de processamento dos servidores trabalhando com pelo menos 80% de sua capacidade máxima. Este valor foi escolhido por ser muito próximo ao limiar do VMWare Distributed Power Management [66], de 81%. Para as simulações foi adotado que os algoritmos efetuarão a coleta de informações de uso de processamento dos servidores a cada 5 segundos.

- *Fan Control* → Controle de refrigeração ativa, usando um modelo de consumo de potência linear proporcional ao *DVFS*.

- *QoS* → Qualidade de Serviço no processo de submissão de cargas, efetuado pelo algoritmo apresentado na Seção 4.2. Inicialmente este algoritmo foi projetado para trabalhar juntamente com o *Lago Allocator*, mas por trabalhar de forma independente e para tornar a simulação mais completa, este algoritmo foi testado com todos os algoritmos para alocação de máquinas virtuais.

O valor de 70% estipulado para o DVFS quando o processador estiver operando em baixa frequência foi calculado da seguinte forma: na especificação técnica da tecnologia *Intel Enhanced Speed Step* [67] são apresentados processadores que quando submetidos às baixas frequências apresentam um consumo de aproximadamente 25% do consumo máximo. No modelo tratado para as simulações tomamos que a média da potência consumida pelos servidores é de aproximadamente 250 *watts*. Observando as especificações técnicas dos processadores de servidores de última geração no mercado ([68], [69] e [70]), um valor mediano aproximado do consumo é de 115 *watts*. Deste modo, temos que a potência consumida pelo servidor, em carga máxima, no modelo, será de $250 - 115 = 135 watts$ e, portanto, a potência consumida pelo servidor, em carga mínima, será de aproximadamente $135 + \frac{25}{100} \times 115 \approx 164 watts$. Portanto, nesta aproximação o consumo do servidor será de aproximadamente 66% e, inserindo uma margem de segurança, ficou estipulado que este parâmetro será de 70%. Valores próximos também podem ser inferidos de trabalhos na área (por exemplo [71] e [72]). O valor de 70% é corroborado pelos estudos de Fan et al. [73] e Kusic et al. [74].

5.4 Cenários

Foram desenvolvidas duas baterias de simulações. A primeira contempla cenários genéricos, com servidores de menor poder de processamento e números reduzidos de máquinas virtuais que estes suportam (limitadas principalmente pela capacidade de processamento dos servidores). A segunda bateria de simulações contempla cenários com servidores de alto desempenho e última geração, cada servidor suportando um número elevado de máquinas virtuais.

5.4.1 Cenários Genéricos

Nestes cenários buscamos desenvolver uma prova de conceito para os algoritmos desenvolvidos. Na bateria de simulações que executamos para este cenário consideramos servidores com processadores, genéricos, de menor poder de processamento, e consequentemente um número menor de máquinas virtuais que tais servidores suportam. A seguir especificamos como esta bateria de simulações foi realizada.

Configurações do Data Center

Um *data center* é um conjunto físico de servidores, interconectados por uma rede, disponíveis para receberem máquinas virtuais e, consequentemente, cargas de trabalho (*cloudlets*).

As simulações para avaliar os algoritmos propostos foram conduzidas em *data centers* pequenos, médios, grandes e gigantes. Os dada centers pequenos são compostos por 10 servidores, 20 máquinas virtuais e 20 *cloudlets*; os médios possuem 100 servidores, 200 máquinas virtuais e 200 *cloudlets*; os *data centers* grandes apresentam 1.000 servidores, 2.000 máquinas virtuais e 2.000 *cloudlets*; os gigantes têm 10.000 servidores, 20.000 máquinas virtuais e 20.000 *cloudlets*. Essa igualdade entre o número de *cloudlets* e máquinas virtuais foi escolhida com o objetivo de manter uma máquina virtual dedicada para cada *cloudlet*, maximizando o desempenho do processamento desta. Pode-se, também, colocar mais de uma *cloudlet* para ser processada em uma mesma máquina virtual, mas isto pode deteriorar a capacidade de processamento, uma vez que tal máquina virtual terá que processar múltiplas cargas, além de ser uma operação dependente de sistema operacional, impondo novas variáveis que não fazem parte do escopo deste trabalho. É proposto, como trabalho futuro, analisar o comportamento de máquinas virtuais que processem múltiplas cargas de trabalho. Estas definições de tamanhos pequeno, médio e grande, de um *data center* em função do número de servidores, foram baseadas em [75]. A definição de *data center* "gigante" foi dado ao último caso, devido ao fato do total de servidores deste caso superar o número de servidores das anteriores.

Além da divisão por tamanho, os *data centers* também foram separados em dois tipos: homogêneos, nos quais todos os servidores possuem configurações de hardware idênticas; e heterogêneos, onde há agrupamentos de servidores com configurações distintas. Por exemplo, se um *data center* for chamado de pequeno e homogêneo ele possuirá 10 servidores (com 20 máquinas virtuais e 20 *cloudlets*), todos possuindo iguais configurações de hardware. No caso de um *data center* heterogêneo, ele possuirá os servidores com configurações diferentes de hardware, distribuídos em agrupamentos; por exemplo, em 10 servidores divididos em 3 agrupamentos poderão ser encontrados um agrupamento de 4 servidores com configuração c_1, um agrupamento de 3 servidores com configuração c_2 e um agrupamento de 3 servidores com configuração c_3.

Configurações dos Servidores

O foco dos algoritmos propostos é a minimização do consumo de energia. No entanto, deve-se evitar a escassez de recursos de hardware, de modo que os resultados das simulações não sejam comprometidos por problemas que não são alvo do algoritmo. Por outro lado, servidores com configurações irrealistas não devem ser usados. Realizadas estas considerações, é definido que as configurações comuns para servidores de *data centers* homogêneos e heterogêneos são:

- Entidades de Processamento → Cada servidor tem uma entidade de processamento (o processador);
- Espaço em Disco → Cada servidor possui 1 TB de espaço em disco;
- RAM → Cada servidor tem 24 GB de RAM;
- Largura de Banda → Assume-se que todos os servidores estão em um *data center* não federado, portanto faz sentido que todos estejam conectados entre si através de *ethernet gigabit*;
- Arquitetura dos servidores do *Data Center* → É assumido que todos os servidores usam arquitetura baseada em x86;
- Sistema Operacional → O sistema operacional escolhido para os servidores do *data center* foi GNU/Linux;
- Monitor de Máquina Virtual → O monitor de máquina virtual escolhido para os servidores do *data center* foi o Xen;
- Entidades de Processamento das *Cloudlets* → Cada carga de trabalho consome uma entidade de processamento;

- Tamanho de Arquivo das *Cloudlets* → Cada *cloudlet* submetida ao *data center*, antes do processamento, possui 300 bytes (padrão dos modelos do *CloudSim*);
- Tamanho da Saída das *Cloudlets* → Cada *cloudlet* possui 300 bytes de dados após o processamento (padrão dos modelos do *CloudSim*);
- MIPS das Máquinas Virtuais → As máquinas virtuais do *data center* possuem capacidades de processamento de 500, 750 e 1000 MIPS, em uma distribuição *round-robin*. Por exemplo, em uma simulação com 20 máquinas virtuais são geradas 7 máquinas virtuais com 500 MIPS, 7 com 750 MIPS e 6 com 1000 MIPS;
- RAM das Máquinas Virtuais → Cada máquina virtual possuirá 128 MB de RAM. Este valor foi escolhido por ser o requerimento mínimo de sistemas operacionais populares em servidores ([76] e [77]), embora a alteração deste valor não impacte significamente nos resultados da simulação uma vez que há bastante memória disponível nos servidores para suportar as máquinas virtuais, não sendo um fator de gargalo.
- Largura de Banda das Máquinas Virtuais → Cada máquina virtual possuirá 2,5 Mbps de largura de banda;
- Tamanho da Imagem das Máquinas Virtuais → Cada máquina virtual possui 2,5 GB de tamanho de imagem;
- Monitor de Máquina Virtual → O monitor de máquina virtual, para as máquinas virtuais, é o Xen;
- Consumo Máximo de Potência dos *Coolers* dos Servidores → As simulações foram baseadas no *cooler stock* do Intel i5, que trabalha com 0,6 *ampères* e tensão máxima de 12 *volts*. Em outras palavras o consumo máximo será de 7,2 *watts* de potência;
- Consumo Mínimo de Potência dos *Coolers* dos Servidores → Apesar dos mecanismos de refrigeração ativa permitirem o completo desligamento do *cooler* em boas circunstâncias, para manter uma margem de segurança e garantir a validade das simulações nenhum dissipador ativo é desligado. Por isso foi adotado que cada *cooler* em baixa rotação, usa tensão de 5 *volts* e 0,6 *ampères*, operando a uma potência de 3 *watts*;
- Percentual de Potência de Servidor Desocupado → Nas simulações com *DVFS* ativado, assume-se que os servidores que estiverem trabalhando no nível mínimo de processamento consomem 70% de sua potência máxima. Ou seja, em uma modelagem analítica foi considerado que a potência consumida pelos componentes do

servidor, tais como placa mãe, memórias voláteis e permanentes, e processador em operação mínima, será fixa em 70% da potência máxima consumida pelo servidor. É adotado um modelo linear proporcional à carga de trabalho para calcular a potência consumida, de modo que quando um servidor tiver em carga máxima ele consumirá 100% de sua potência. Por exemplo, quando um servidor de 250 *watts* está operando em 0% de sua capacidade ele consome 175 *watts*; quando operando a 50% de sua capacidade ele consome 212 *watts*; e quando operando na capacidade máxima ele consome 250 *watts*; portanto a função que rege o consumo de potência é $P = 175 + 75 \times getUtilizationOfCpu(CPU)$, onde P é a potência consumida por um servidor em *watts* e $getUtilizationOfCpu(h)$ retorna o uso percentual da capacidade de processamento do servidor h;

- Limiar para Migração de Máquinas Virtuais → Para as simulações com limiar para migração de máquinas virtuais ativado, tenta-se a migração das máquinas dos servidores que estiverem trabalhando abaixo de 80% de suas capacidades, seguido pelos seus desligamentos;

- Tamanho das *Cloudlets* → As *cloudlets* no *data center* possuem 100.000, 150.000, 200.000, 250.000 e 300.000 milhões de instruções em uma distribuição *round-robin*;

- Prioridades das *Cloudlets* → Cada *cloudlet* pode assumir uma dentre três prioridades: baixa, média e alta; sendo que a distribuição dessas prioridades também é round-robin com prioridades alta, média, média e baixa (mantendo-se 25% de cargas com baixa prioridade, 50% com prioridade média e 25% com alta prioridade).

O limiar para migração de máquinas virtuais, fixo em 80%, foi escolhido com base nos padrões do *CloudSim* (apresentado na Seção 5) e de centenas de trabalhos baseados no referido simulador [15]. Choi et al. [78] mostra a complexidade da escolha de limiares dinâmicos para tal migração, assunto que pretende-se abordar em trabalhos futuros. Vale ressaltar que a migração de máquinas virtuais tem um custo associado, proveniente do tempo necessário para manter os servidores, entre os quais ocorre a migração, ligados durante a migração, adicionado do custo de energia da rede. O custo para manter os servidores ligados inclui, mas não se limita, aos consumos de energia dos processadores, discos, RAM e periféricos em atividade. Neste trabalho tais valores foram considerados na validação dos algoritmos propostos.

A Tabela 5.1 exemplifica como será essa distribuição para um conjunto de *cloudlets*. CID representa o ID da *cloudlet* e MI milhões de instruções. As máquinas virtuais do *data center* possuem capacidade de processamento de 500, 750 e 1000 MIPS em uma distribuição *round-robin*. Por exemplo, em uma simulação na qual 20 máquinas virtuais são geradas, 7 máquinas virtuais serão criadas com 500 MIPS, 7 máquinas virtuais terão

CID	C_1	C_2	C_3	C_4	C_5	C_6
MI	100 k	150 k	200 k	250 k	300 k	100 k
Prioridade	Baixa	Média	Média	Alta	Baixa	Média

Tabela 5.1: Cenário 1: Exemplo de Distribuição de *Cloudlets* e Prioridades

750 MIPS e 6 máquinas virtuais ficarão com 1000 MIPS. Além disso é adotado que cada máquina virtual possui 128 MB de RAM, 2,5 Mbps de largura de banda; cada máquina virtual tem tamanho de imagem de 2,5 GB.

O aquecimento de processadores depende de diversos fatores, incluindo a temperatura, o dissipador, a arquitetura e a litografia. Tendo em vista que não existe um padrão de aquecimento entre os vários processadores existentes, é muito difícil estimar um modelo válido para todos eles. No entanto, é sabido que a temperatura de um processador é diretamente proporcional à potência dissipada, e cresce proporcionalmente à frequência e com o quadrado de sua tensão de operação. Por exemplo, quando um processador, de 3000 *MHz*, possuindo um Vcore de 1,4 *volts*, 130 *watts* de potência e *DVFS* ativado, reduz seu *clock* para 1200 *MHz* e o Vcore para 1,1 *volts*, passa a consumir 32,1 *watts*.

Baseado neste princípio e com o objetivo de manter a validação genérica, foi adotado algo que faz sentido: com o *DVFS* e *Fan Control* ativados, a potência de operação do dissipador ativo será proporcional à carga de trabalho do processador, adotando um modelo linear para isso. Desse modo, o *cooler* consumirá 7,2 *watts* em capacidade máxima (quando alimentado com 12 *volts*) e, na sua capacidade mínima, o consumo será de $5 \times \frac{7,2}{12} = 3$ *watts* (quando alimentado com 5 *volts*).

É mostrada na Figura 5.1 a proporcionalidade entre carga da entidade de processamento, consumo de energia de uma máquina com *DVFS* ativado e o consumo de energia do *cooler* com *Fan Control* ativado.

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

Configurações para Data Centers Homogêneos

Para as simulações em *data centers* homogêneos, as seguintes configurações são adotadas:

- MIPS dos Servidores → Cada servidor em um *data center* homogêneo possui uma entidade de processamento, ou seja, um processador contendo um núcleo, com 2.000 MIPS;
- Consumo de Potência dos Servidores → O consumo de potência para cada servidor em um *data center* homogêneo é de 250 *watts*.

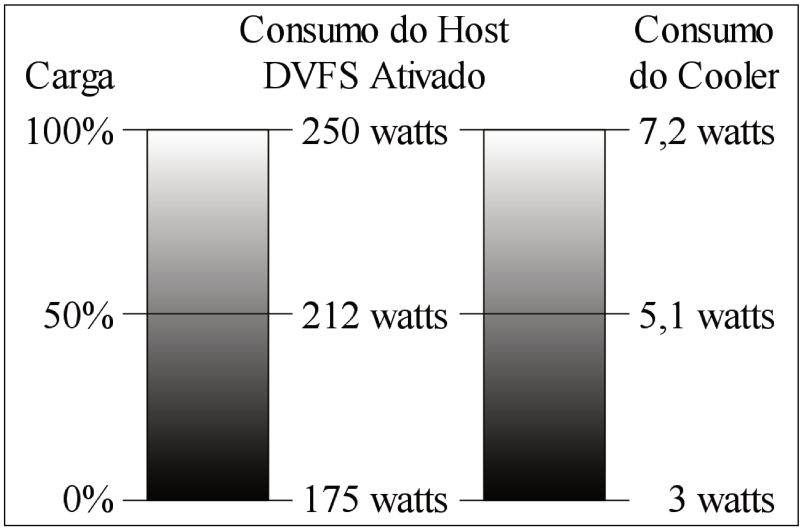


Figura 5.1: Cenário 1: Proporção entre Carga da Entidade de Processamento, Consumo de Energia de um Servidor com DVFS Ativado e Consumo de Energia de um Dissipador Ativo com Fan Control Ativado

Servidor	h_1	h_2	h_3	h_4	h_5	h_6	h_7
MIPS	1.000	1.500	2.000	2.500	3.000	1.000	1.500
Potência (*watts*)	200	250	300	350	200	250	300

Tabela 5.2: Cenário 1: Exemplo de Configuração Inicial de Servidores em um Data Center Heterogêneo

Configurações para Data Centers Heterogêneos

Para simulações em *data centers* heterogêneos, as seguintes configurações são adotadas:

- MIPS dos Servidores → Cada servidor em um *data center* heterogêneo possui uma entidade de processamento, ou seja, um processador contendo um núcleo, com as seguintes MIPS em uma distribuição *round-robin*: 1.000, 1.500, 2.000, 2.500, 3.000;
- Consumo de Potência dos Servidores → Cada servidor em um *data center* heterogêneo possui o seguinte consumo de potência em uma distribuição *round-robin*, 200, 250, 300 e 350 *watts*.

Está exemplificado na Tabela 5.2 como estão distribuídos os MIPS/potência dos servidores neste tipo de *data center*.

Resultados das Simulações

Trinta simulações foram realizadas para cada algoritmo, para cada um dos algoritmos e configurações possíveis mencionadas na Seção 5.3. Para descrever os algoritmos são usadas as seguintes siglas: *RR - Round Robin*; *BRS - Best Resource Selection*; *MPD - Minimum Power Diff*; *LA - Lago Allocator*; $LA-Q$ - *Lago Allocator* com Qualidade de Serviço (QoS) ativada.

Para descrever as configurações usadas pelos algoritmos em cada conjunto de simulações, são utilizadas as seguintes siglas: *N - Non Power Aware*; P = *Power Aware*; *D - DVFS*; *T - Threshold*; *F - Fan Control*. Por exemplo, o *Lago Allocator* com configurações *Power Aware*, *DVFS*, *Threshold* e *Fan Control* será referenciado como $LA-PDTF$. Para todos os valores de medição de energia apresentados nas próximas seções a métrica utilizada é $quilowatts \times hora$ e, para medidas de tempo, *segundos*. Para todos os gráficos apresentados nesta seção é adotado um intervalo de confiança de 95%.

O resultado desejável é a minimização do consumo de energia. Deve-se contemplar, também, a redução dos tempos de execução das cargas com alta prioridade, sem que esta acarrete um aumento substancial do consumo de energia. Além disso, os tempos de execução das cargas de baixas prioridades também não devem ser inaceitavelmente acentuados.

Vale ressaltar que o modelo usado nas simulações é o *bag-of-tasks*, ou seja, um conjunto de tarefas é enviado ao escalonador, em seguida ele executa as tarefas e o processo é finalizado. Em ambientes reais podem ser necessárias pequenas adaptações nas políticas de desligamento de máquinas. Por exemplo, se durante o processo de escalonamento de um conjunto de tarefas chega *outro* conjunto de tarefas, será necessário religar servidores para o processamento destas últimas. Uma política prática adotada por alguns sistemas de nuvem [66] é garantir que os servidores, após ligados, permaneçam um tempo mínimo sem desligar.

São apresentados nas Figuras 5.2 a 5.9 os resultados do consumo de energia dos algoritmos RR, BRS, MPD, LA e $LA-Q$, com configurações P, PD, PT, PDT e $PDTF$, resultantes das simulações.

De modo geral, os resultados das simulações não cientes de energia consumiram aproximadamente o dobro da energia do pior caso das simulações cientes de energia. Por isso os resultados para estas simulações não foram mostradas nos gráficos, mas estão apresentados nas Tabelas 5.3 e 5.4. Essas opções são inquestionavelmente a pior solução do ponto de vista do consumo de energia.

Observando os gráficos relacionados com os consumos de energia dos algoritmos, é possível perceber uma superioridade do LA e do $LA-Q$ sobre o MPD, especialmente em *data centers* heterogêneos. Também é perceptível que há praticamente um empate entre o LA e LA-Q. É apresentado na Tabela 5.5 um resumo comparativo entres os consumos de

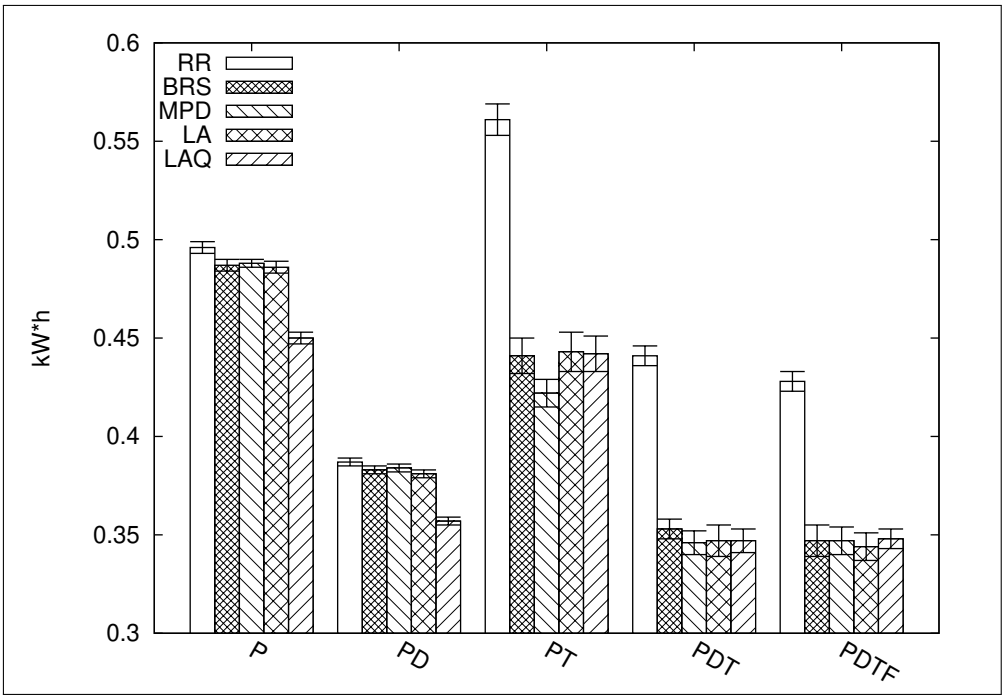


Figura 5.2: Cenário 1: Consumo de Energia em Data Center Pequeno Homogêneo

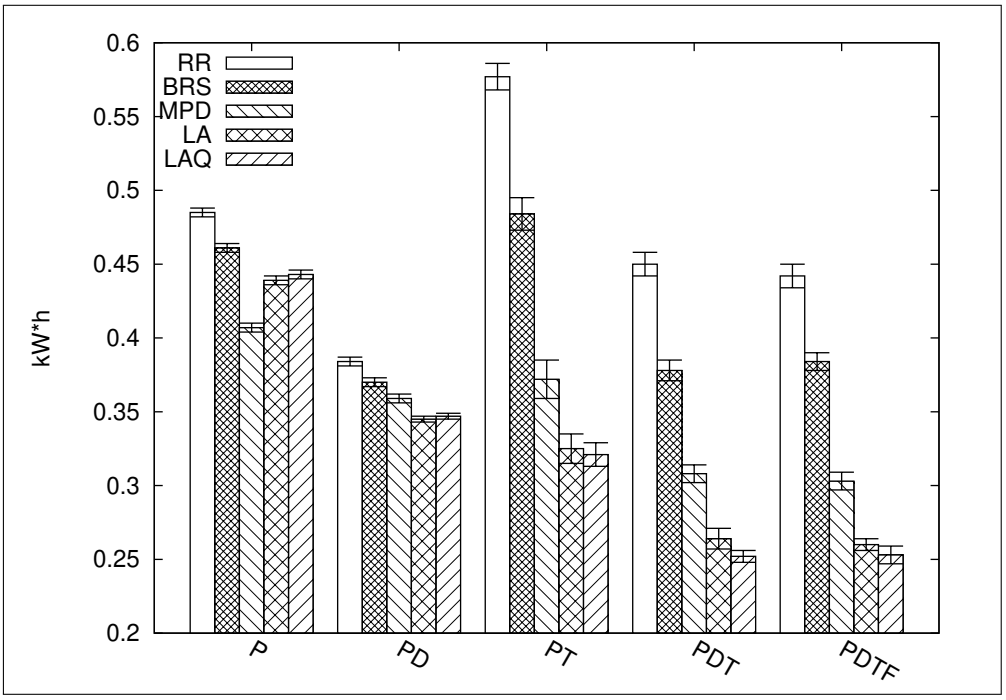


Figura 5.3: Cenário 1: Consumo de Energia em Data Center Pequeno Heterogêneo

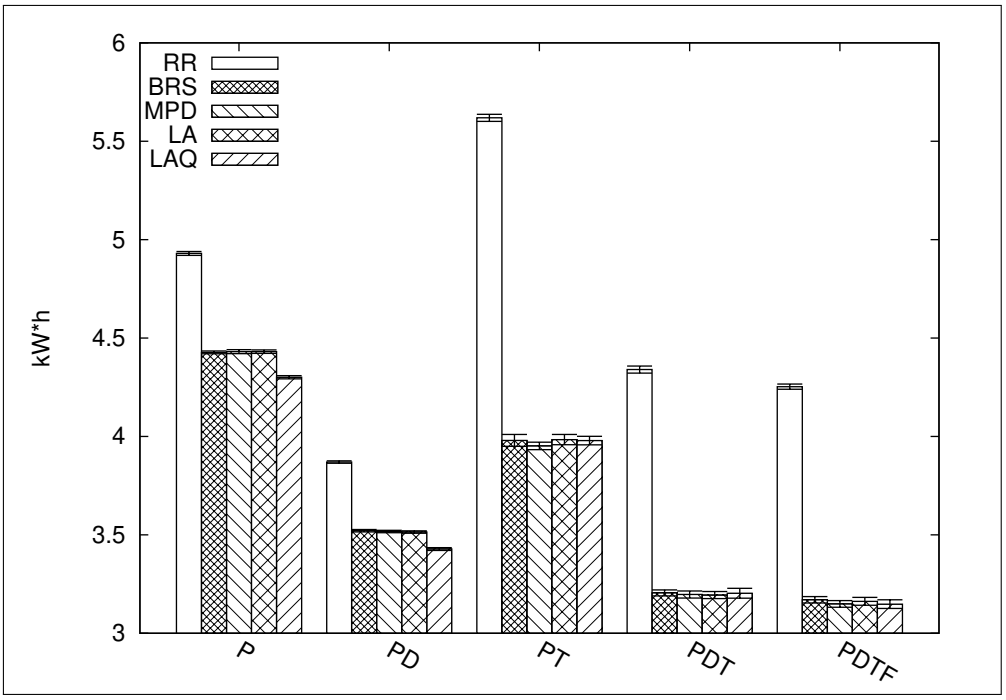


Figura 5.4: Cenário 1: Consumo de Energia em Data Center Médio Homogêneo

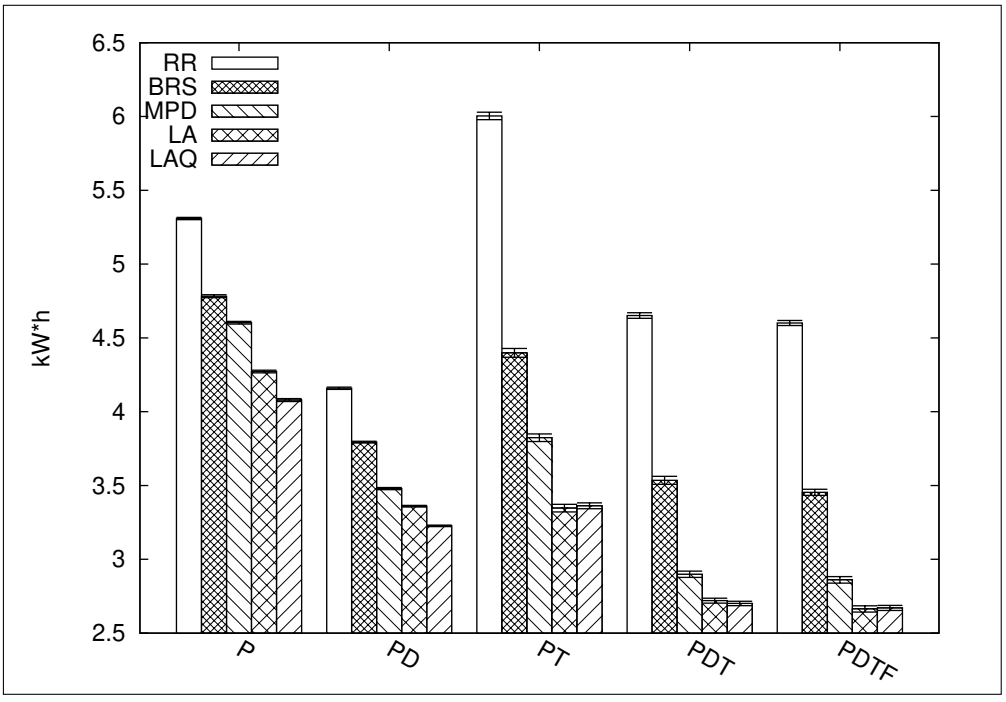


Figura 5.5: Cenário 1: Consumo de Energia em Data Center Médio Heterogêneo

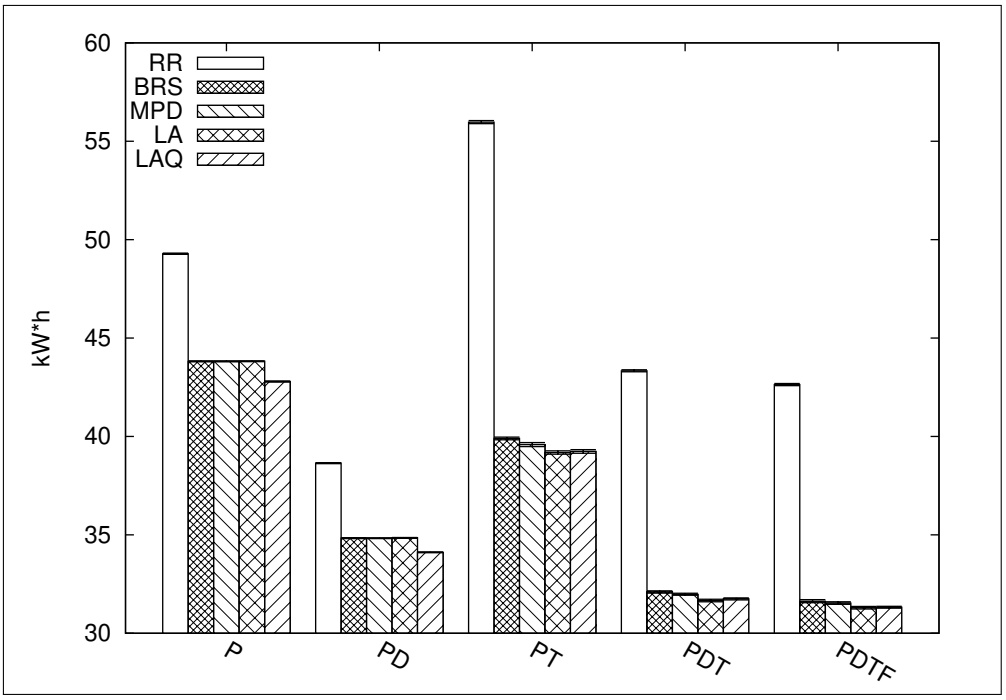


Figura 5.6: Cenário 1: Consumo de Energia em Data Center Grande Homogêneo

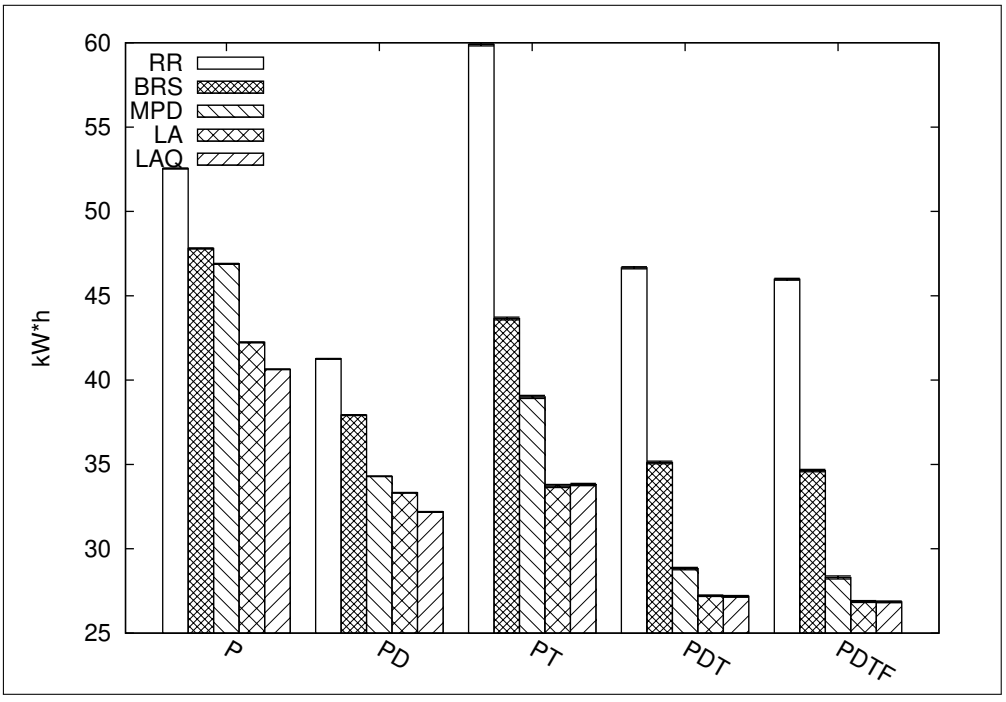


Figura 5.7: Cenário 1: Consumo de Energia em Data Center Grande Heterogêneo

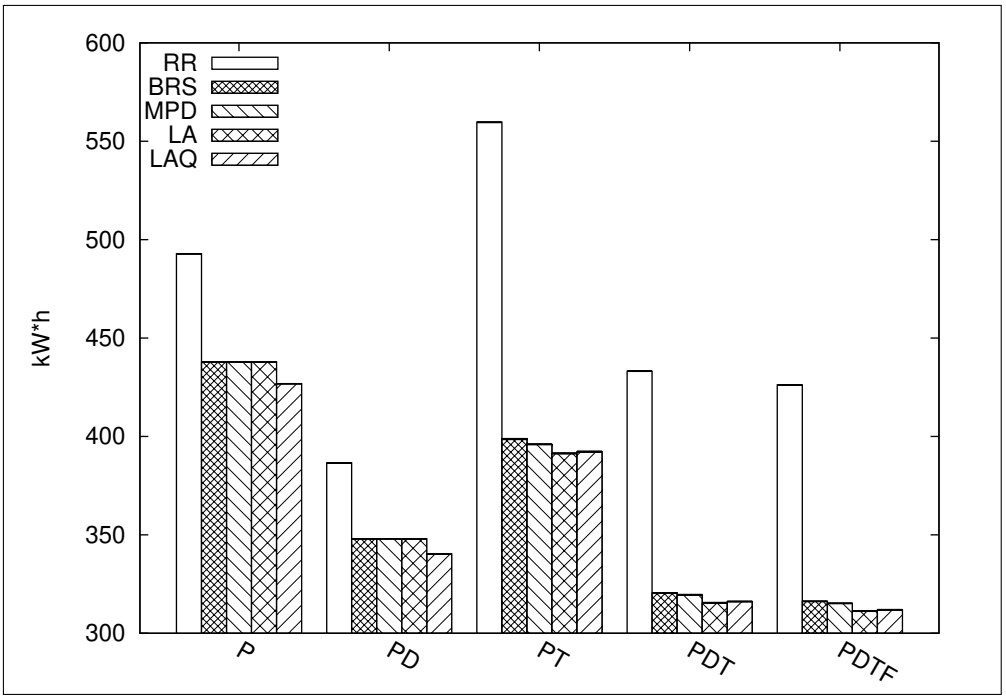


Figura 5.8: Cenário 1: Consumo de Energia em Data Center Gigante Homogêneo

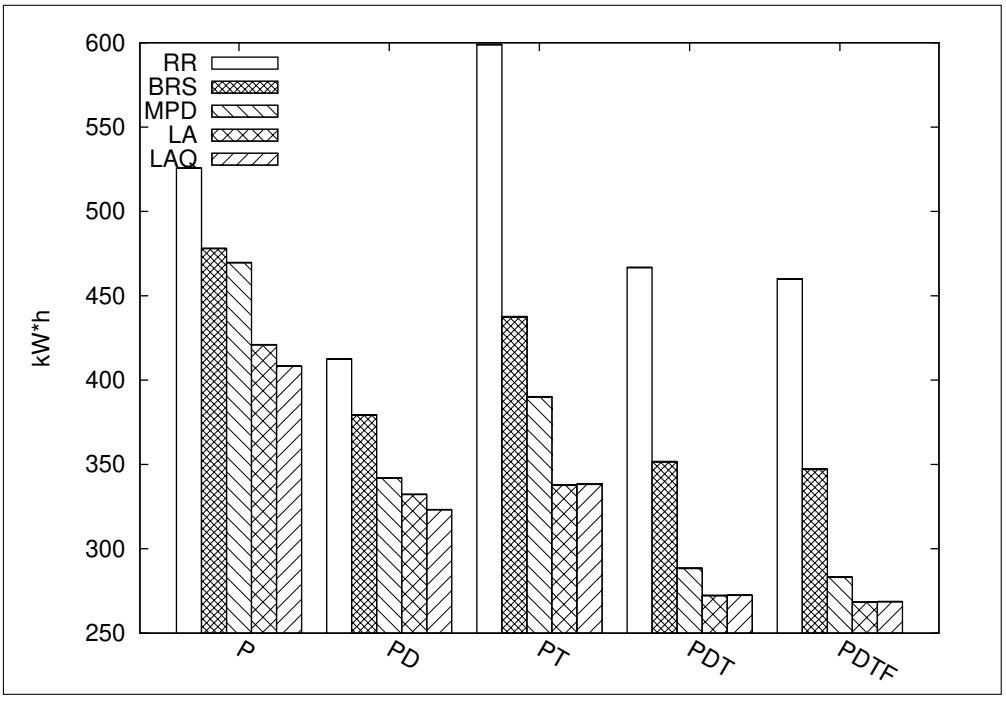


Figura 5.9: Cenário 1: Consumo de Energia em Data Center Gigante Heterogêneo

Cenário	RR	BRS
P. Homo	$0,837 \pm 0,015$	$0,834 \pm 0,010$
P. Hetero	$0,890 \pm 0,014$	$0,889 \pm 0,013$
M. Homo	$8,893 \pm 0,071$	$8,944 \pm 0,070$
M. Hetero	$9,770 \pm 0,073$	$9,732 \pm 0,069$
Gd. Homo	$91,864 \pm 0,509$	$92,038 \pm 0,516$
Gd. Hetero	$101,343 \pm 0,484$	$101,267 \pm 0,616$
Gg. Homo	$946,297 \pm 4,236$	$947,453 \pm 3,468$
Gg. Hetero	$1043,601 \pm 4,411$	$1034,561 \pm 3,407$

Tabela 5.3: Cenário 1: Consumo de Energia com Configuração Não Ciente de Energia para RR e BRS

Cenário	MPD	LA	LA-Q
P. Homo	$0,833 \pm 0,011$	$0,838 \pm 0,013$	$0,844 \pm 0,013$
P. Hetero	$0,891 \pm 0,014$	$0,891 \pm 0,012$	$0,881 \pm 0,014$
M. Homo	$8,897 \pm 0,060$	$8,881 \pm 0,069$	$8,939 \pm 0,063$
M. Hetero	$9,807 \pm 0,068$	$9,812 \pm 0,068$	$9,797 \pm 0,076$
Gd. Homo	$91,991 \pm 0,639$	$92,373 \pm 0,607$	$92,269 \pm 0,548$
Gd. Hetero	$101,458 \pm 0,661$	$101,471 \pm 0,596$	$101,382 \pm 0,647$
Gg. Homo	$948,611 \pm 4,636$	$946,875 \pm 3,944$	$942,593 \pm 2,903$
Gg. Hetero	$1040,800 \pm 4,506$	$1038,126 \pm 5,155$	$1038,890 \pm 3,911$

Tabela 5.4: Cenário 1: Consumo de Energia com Configuração Não Ciente de Energia para MPD, LA e LA-Q

Cenário	LA-PDTF x MPD-PDT	LA-PDTF x MPD-PDTF	LA-PDTFQ x MPD-PDT	LA-PDTFQ x MPD-PDTF	LA-PDTF x LA-PDTFQ
P. Homo	Empate	Empate	Empate	Empate	Empate
P. Hetero	LA é Melhor (14,4% a 23,0%)	LA é Melhor (12,3% a 21%)	LA-Q é Melhor (17% a 27,1%)	LA-Q é Melhor (14,8% a 25%)	Empate
M. Homo	Empate	Empate	LA-Q é Melhor (0,3% a 2,9%)	Empate	Empate
M. Hetero	LA é Melhor (7,2% a 10,5%)	LA é Melhor (5,8% a 9,1%)	LA-Q é Melhor (7,1% a 10%)	LA-Q é Melhor (5,6% a 8,6%)	Empate
Gd. Homo	LA é Melhor (1,8% a 2,6%)	LA é Melhor (0,3% a 1,2%)	LA-Q é Melhor (1,8% a 2,4%)	LA-Q é Melhor (0,3% a 1%)	Empate
Gd. Hetero	LA é Melhor (6,8% a 7,8%)	LA é Melhor (4,8% a 5,9%)	LA-Q é Melhor (6,9% a 7,8%)	LA-Q é Melhor (4,9% 5,9%)	Empate
Gg. Homo	LA é Melhor (2,6% a 2,8%)	LA é Melhor (1,2% a 1,4%)	LA-Q é Melhor (2,4% a 2,6%)	LA-Q é Melhor (1% a 1,2%)	LA é Melhor (0,1% a 0,3%)
Gg. Hetero	LA é Melhor (7,4% a 7,6%)	LA é Melhor (5,4% a 5,6%)	LA-Q é Melhor (7,2% a 7,5%)	LA-Q é Melhor (5,3% a 5,5%)	LA é Melhor (0% a 0,2%)

Tabela 5.5: Cenário 1: Resumo Comparativo ente MPD-PDT, MPD-PDTF, LA-PDTF e LA-PDTFQ

energia do $MPD-PDT$, $MPD-PDTF$, $LA-PDTF$ e $LA-PDTFQ$. Com base nestes resultados é constatado que apenas nos cenários de *data centers* pequenos e homogêneos, e médios e homogêneos, ocorre um empate entre $LA/LA-Q$ e MPD, independentemente da configuração do MPD. Para todos os demais cenários o $LA/LA-Q$ se mostra superior ao MPD.

Era de se esperar que o $LA-Q$, devido à qualidade de serviço que presta, possuísse consumo de energia significativamente maior do que o LA. No entanto, para as simulações realizadas, isso não ocorreu. Pode-se constatar na Tabela 5.5 que, de fato, o $LA-Q$ foi pior que o LA apenas no caso de *data centers* gigantes e, mesmo assim, por uma diferença muito pequena.

São apresentados nas Figuras 5.10 a 5.17 os tempos totais consumidos para a conclusão de todas as cargas de trabalho submetidas ao escalonador, para cada algoritmo e configuração. Estes tempos são chamados *makespan*.

Observando as figuras comparativas do *makespan* entre os algoritmos e configurações é possível constatar que, em nenhum caso, os tempos consumidos violam a regra de tempo máximo aceitável, definida na Seção 5.2.

De fato, mesmo com configurações distintas, o pior caso ocorre no caso de *data center* gigante e homogêneo, onde o caso mais lento do $LA/LA-Q$ ($LAPT$) é 45% mais lento do que melhor caso encontrado ($MPDPD$). Portanto, pode-se dizer que, para as simulações realizadas, o $LA-Q$, em qualquer configuração, nunca foi considerado inaceitavelmente

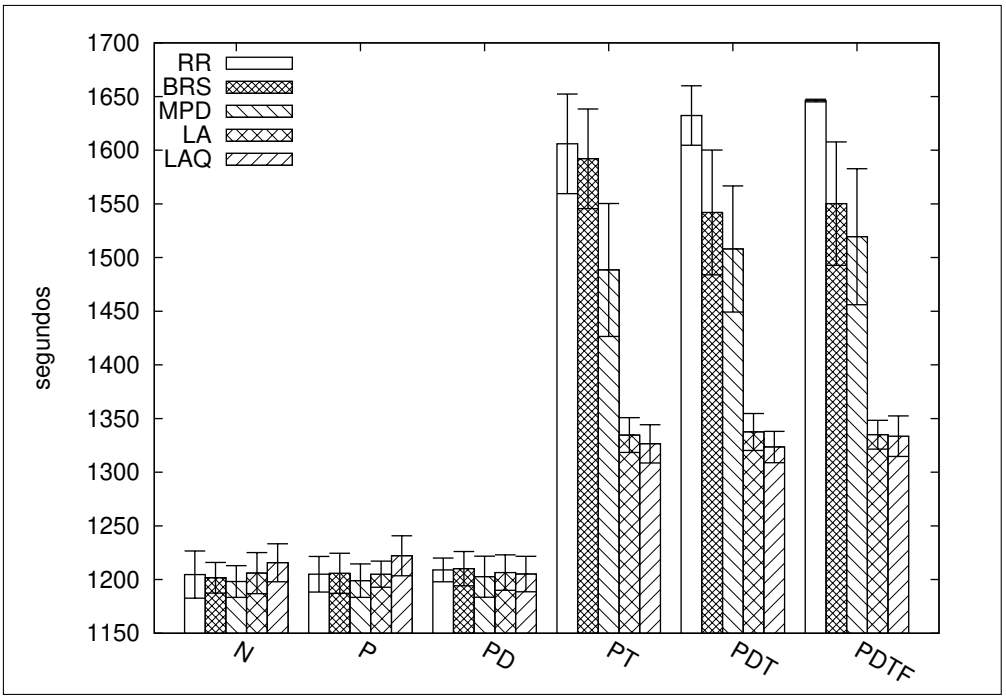


Figura 5.10: Cenário 1: Makespan em Data Center Pequeno Homogêneo

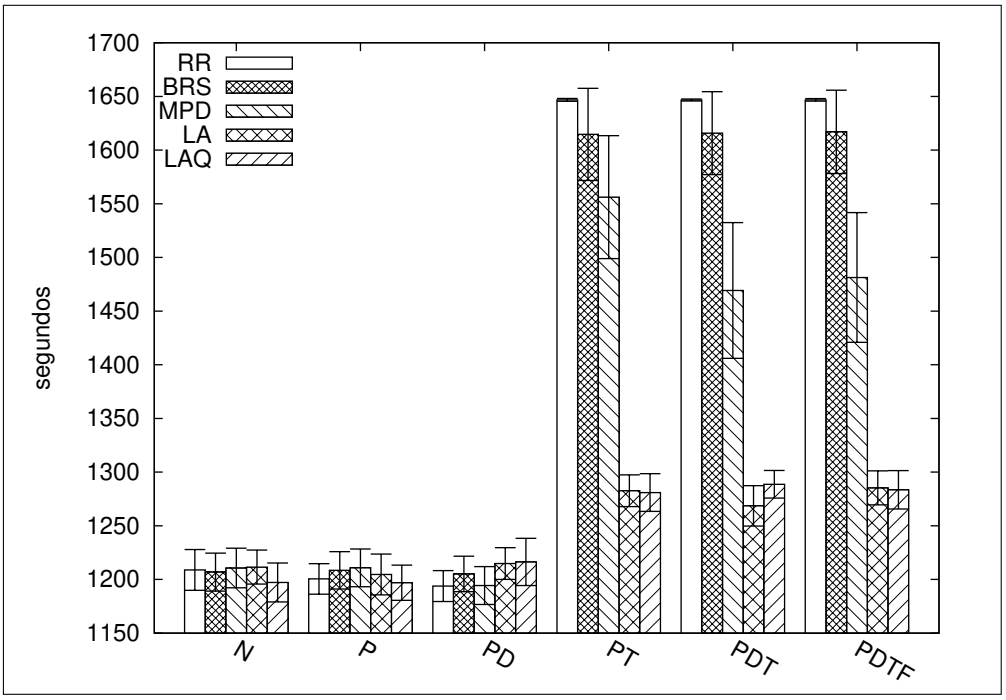


Figura 5.11: Cenário 1: Makespan em Data Center Pequeno Heterogêneo

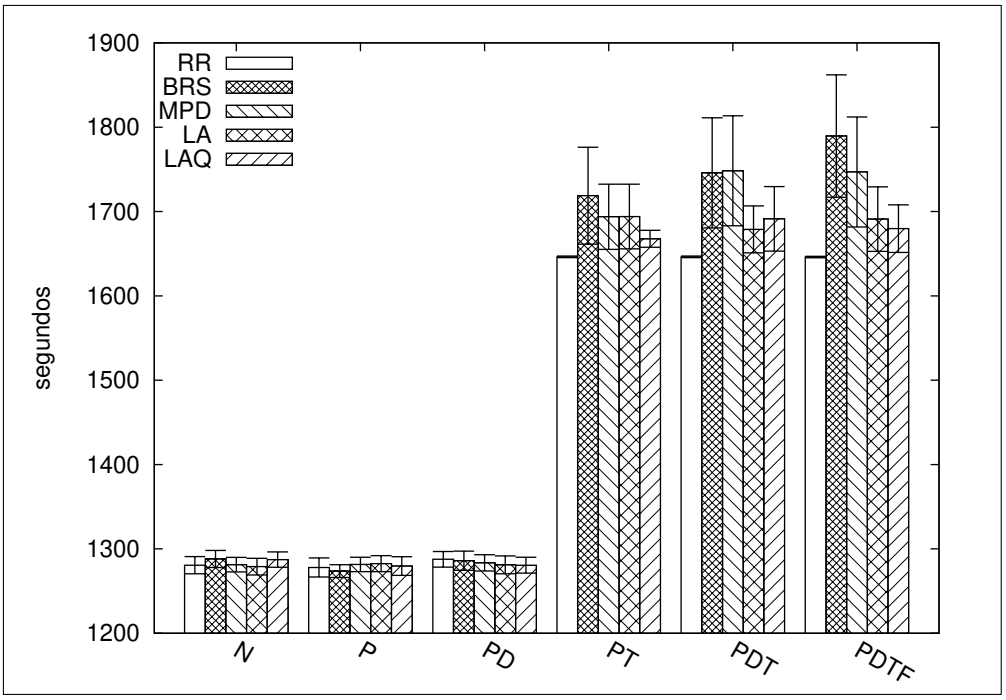


Figura 5.12: Cenário 1: Makespan em Data Center Médio Homogêneo

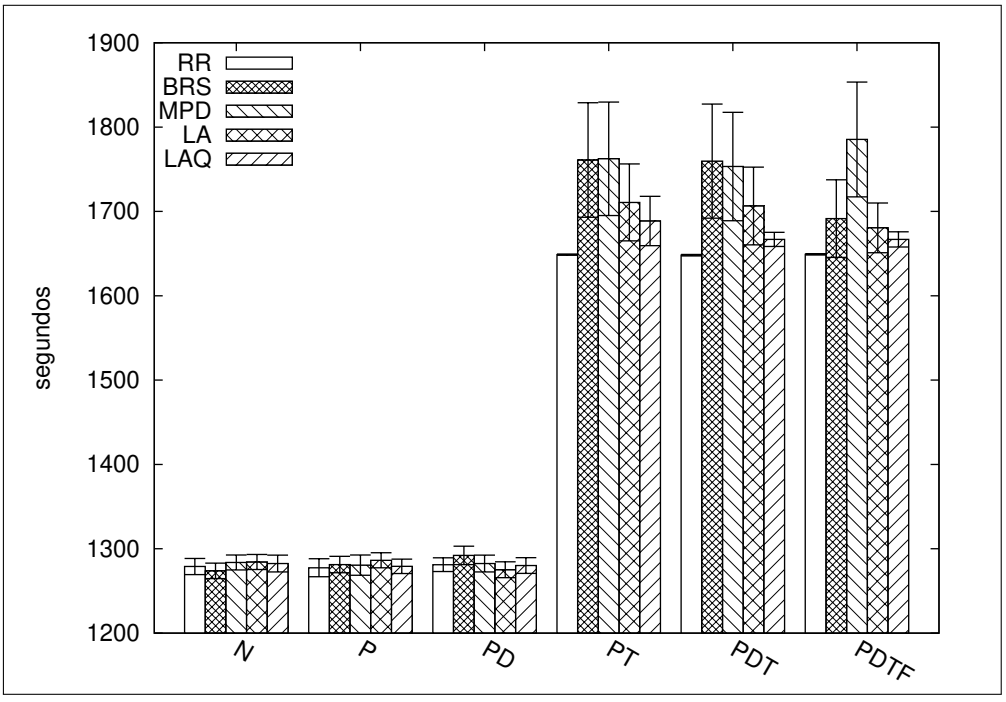


Figura 5.13: Cenário 1: Makespan em Data Center Médio Heterogêneo

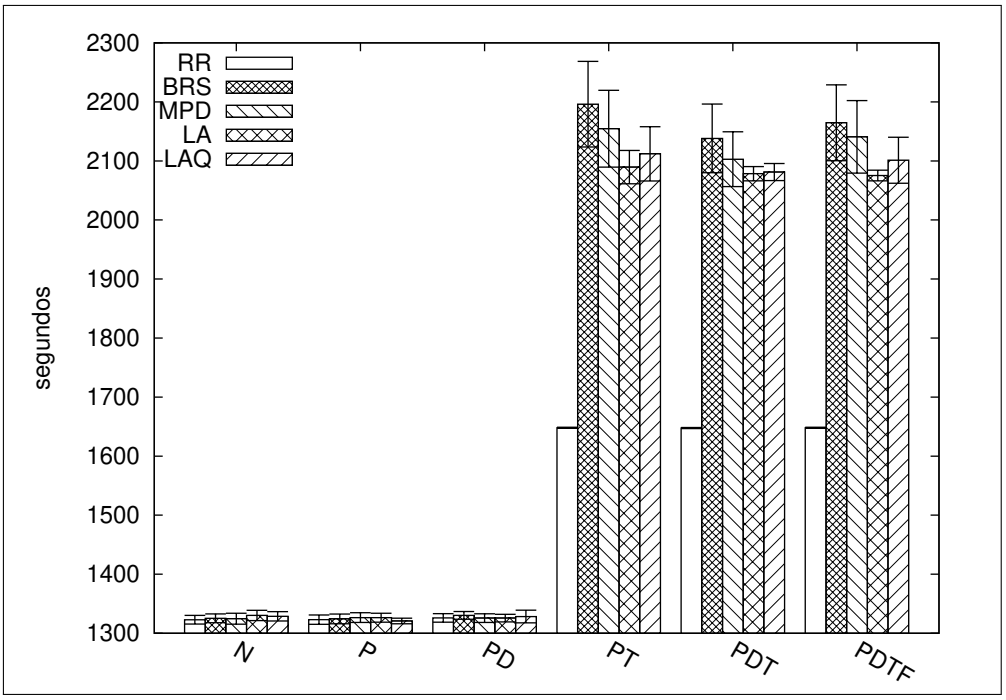


Figura 5.14: Cenário 1: Makespan em Data Center Grande Homogêneo

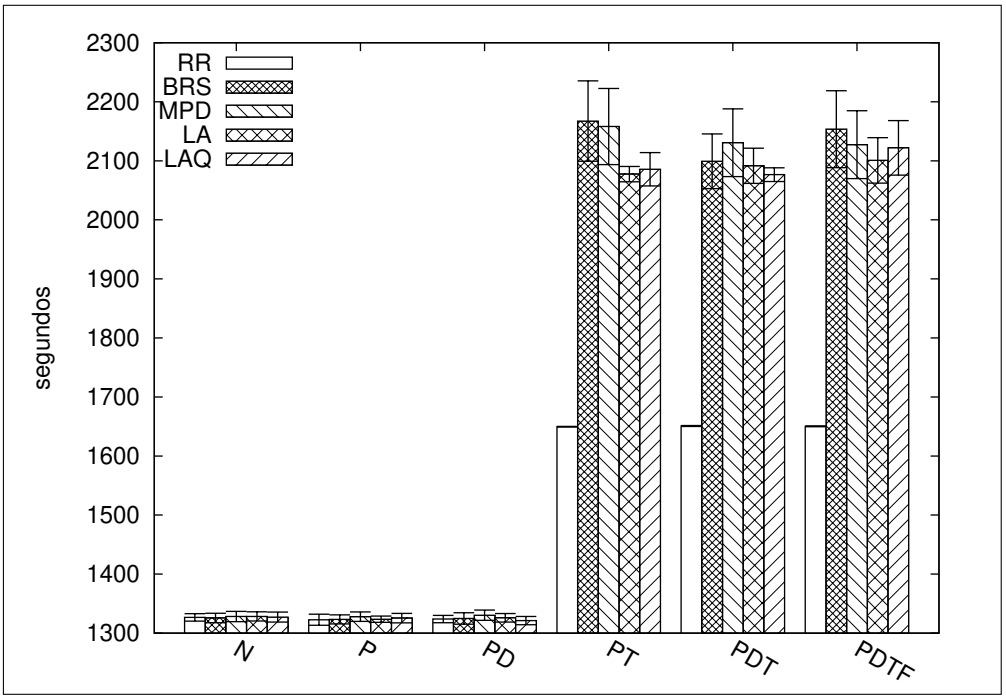


Figura 5.15: Cenário 1: Makespan em Data Center Grande Heterogêneo

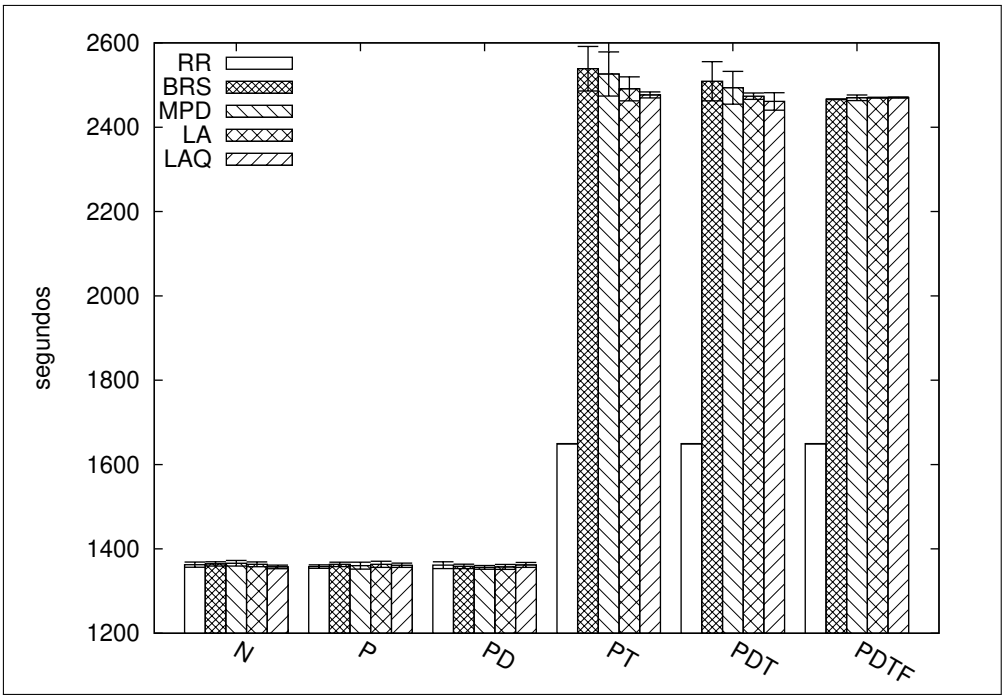


Figura 5.16: Cenário 1: Makespan em Data Center Gigante Homogêneo

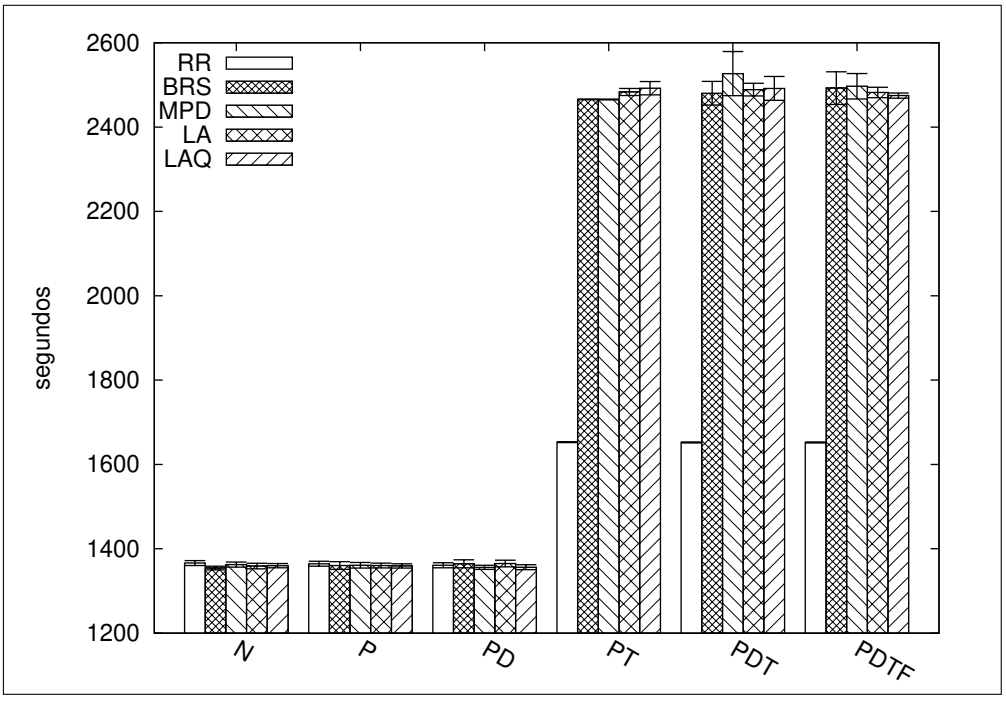


Figura 5.17: Cenário 1: Makespan em Data Center Gigante Heterogêneo

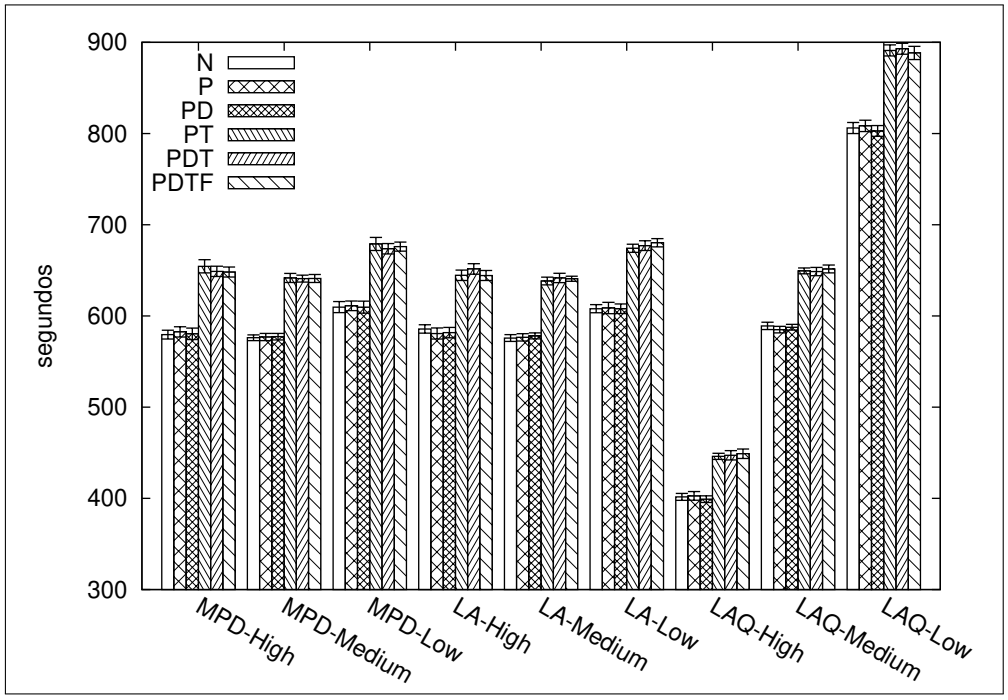


Figura 5.18: Cenário 1: Tempo Médio de Conclusão de Cloudlets em Data Center Pequeno Homogêneo

mais lento do que qualquer outro algoritmo.

São apresentados nas Figuras 5.18 a 5.25 os tempos médios de execução das *cloudlets*, agrupadas por prioridades, para os algoritmos MPD, LA e $LA-Q$. Os algoritmos RR e BRS não são apresentados nessa avaliação, uma vez que eles não tratam prioridades, possuindo comportamentos semelhantes ao MPD e LA.

Com base nos gráficos apresentados é perceptível uma clara redução do tempo de execução das *cloudlets* que possuem alta prioridade para os cenários tratados. Para estas *cloudlets* a redução do tempo de execução permaneceu entre 40% a 49%, para as *cloudlets* com prioridades médias houve uma ligeira melhoria de até 5% nos tempos de execução e, para as *cloudlets* com prioridades baixas, como era esperado, ocorreu um incremento no tempo de execução, em média entre 22% e 29%.

5.4.2 Cenários de Alta Performance

Nestes cenários objetivamos analisar o funcionamento dos algoritmos propostos em um cenário baseado em processadores atuais de alta performance. Com isso os servidores também suportam um número bem maior de máquinas virtuais. Os processadores dos servidores escolhidos para as simulações realizadas para este cenário são:

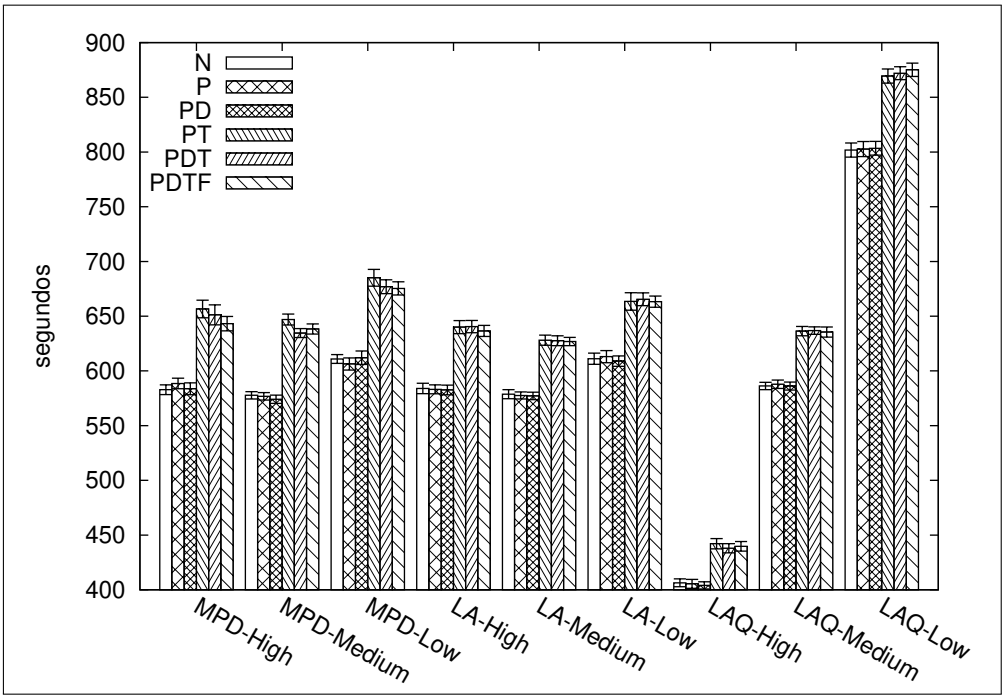


Figura 5.19: Cenário 1: Tempo Médio de Conclusão de Cloudlets em Data Center Pequeno Heterogêneo

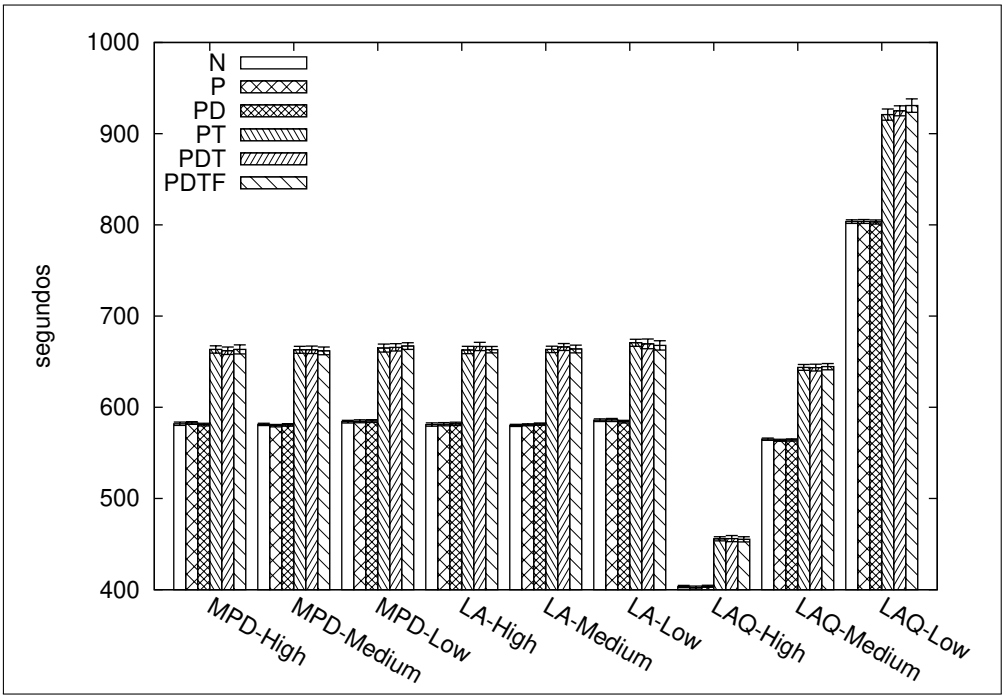


Figura 5.20: Cenário 1: Tempo Médio de Conclusão de Cloudlets em Data Center Médio Homogêneo

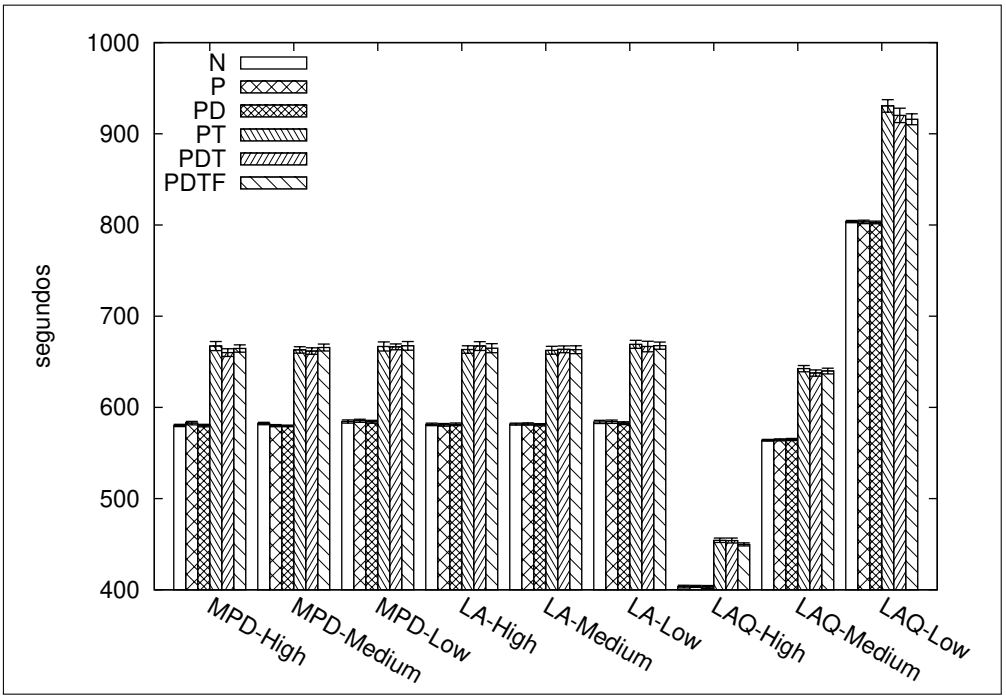


Figura 5.21: Cenário 1: Tempo Médio de Conclusão de Cloudlets em Data Center Médio Heterogêneo

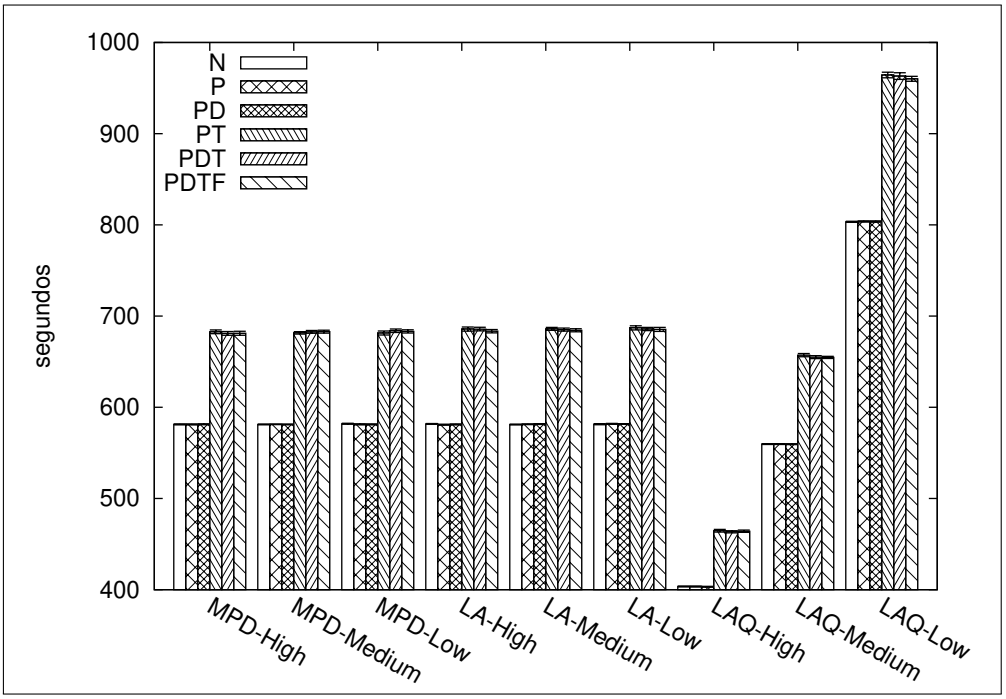


Figura 5.22: Cenário 1: Tempo Médio de Conclusão de Cloudlets em Data Center Grande Homogêneo

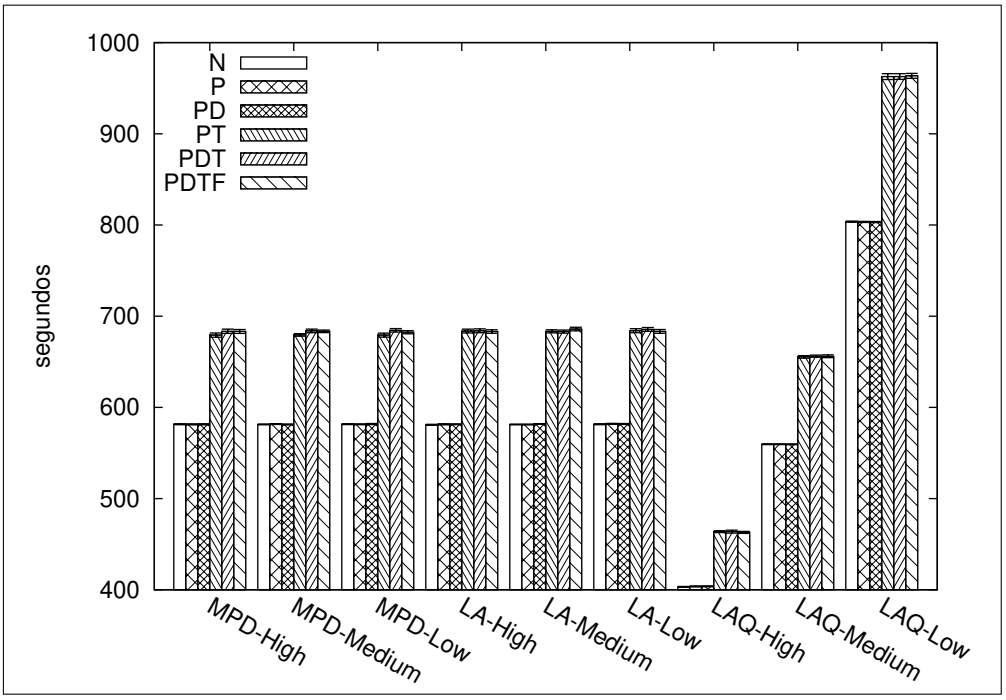


Figura 5.23: Cenário 1: Tempo Médio de Conclusão de Cloudlets em Data Center Grande Heterogêneo

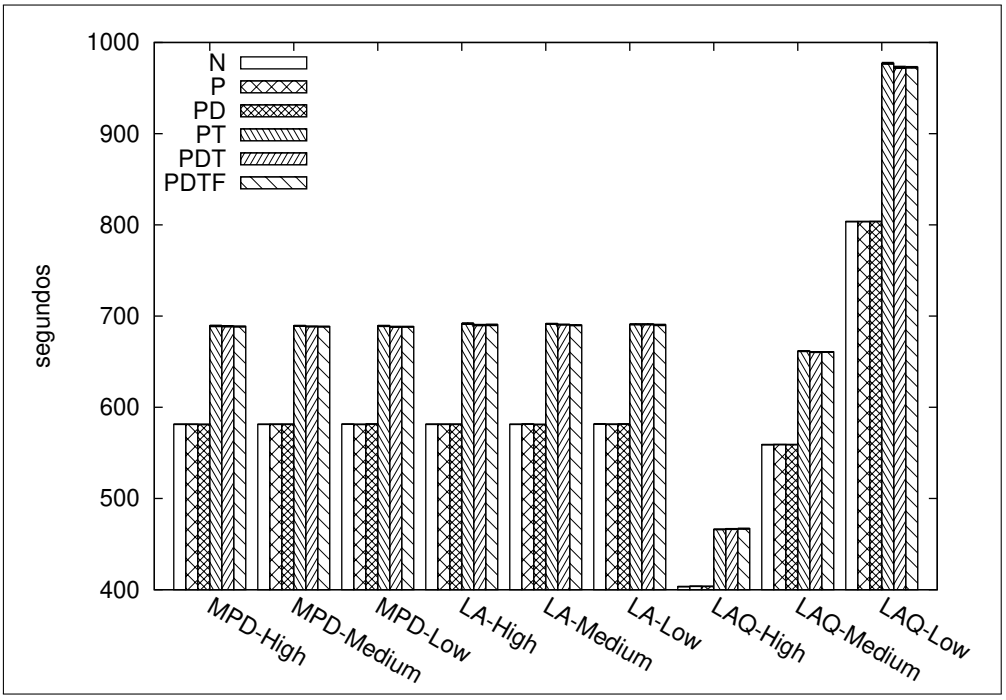


Figura 5.24: Cenário 1: Tempo Médio de Conclusão de Cloudlets em Data Center Gigante Homogêneo

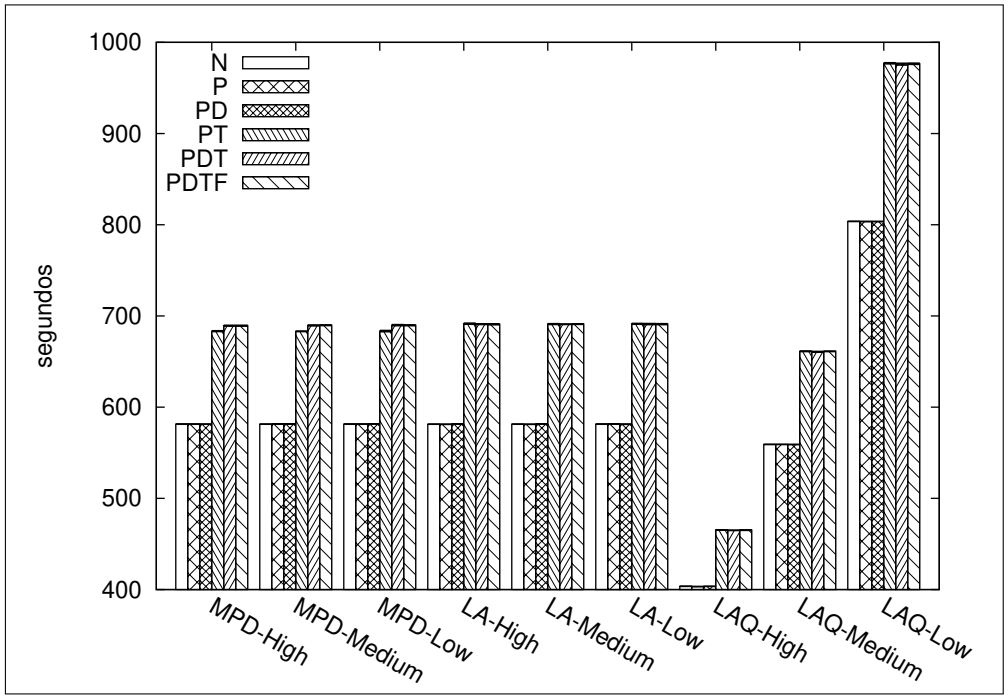


Figura 5.25: Cenário 1: Tempo Médio de Conclusão de Cloudlets em Data Center Gigante Heterogêneo

- Intel Xeon E5603, possuindo 4 núcleos de 1,6 GHz cada;
- Intel Xeon E7520, possuindo 4 núcleos de 1,86 GHz cada;
- Intel Xeon X5667, possuindo 4 núcleos de 3,06 GHz cada;
- AMD Opteron 6204, possuindo 4 núcleos de 3,3 GHz cada;
- AMD Opteron 6220, possuindo 8 núcleos de 3 GHz cada.

O critério para a seleção destes processadores foi tentar criar uma disparidade significativa e capacidade de processamento entre os servidores, para verificar em uma situação real como os algoritmos se comportam em situações de heterogeneidade grande, mas em *data centers* que usam processadores de última geração. A seguir especificamos como esta bateria de simulações foi desenvolvida.

Para efeitos de simulação consideramos que cada operação realizada pelas cargas de trabalho pode ser concretizada em um ciclo de *clock* dos processadores, ou seja, há uma razão de 1:1 entre frequência (MHz) e MIPS.

Configurações do Data Center

Para esta bateria de simulações também conduzimos os algoritmos propostos em *data centers* pequenos, médios, grandes e gigantes. Comparada à bateria anterior há uma diferenciação no número de máquinas virtuais e *cloudlets*. Nesta bateria, os *data centers* pequenos são compostos por 10 servidores, 60 máquinas virtuais e 60 *cloudlets*; os médios possuem 100 servidores, 600 máquinas virtuais e 600 *cloudlets*; os *data centers* grandes apresentam 1.000 servidores, 6.000 máquinas virtuais e 6.000 *cloudlets*; os gigantes têm 10.000 servidores, 60.000 máquinas virtuais e 60.000 *cloudlets*.

Assim como na bateria anterior, também separamos os *data centers* em dois tipos: homogêneos e heterogêneos.

Configurações dos Servidores

As configurações comuns para servidores de *data centers*, homogêneos e heterogêneos, modificadas nesta bateria de simulações, comparada à anterior, são:

- Entidades de Processamento → Cada servidor tem um processador, podendo ter vários núcleos (especificado na seção referente ao *data center* em questão);
- MIPS das Máquinas Virtuais → As máquinas virtuais do *data center* possuem capacidades de processamento de 750, 1500, 3000 MIPS, em uma distribuição *round-robin*. Por exemplo, em uma simulação com 20 máquinas virtuais são geradas 7 máquinas virtuais com 750 MIPS, 7 com 1500 MIPS e 6 com 3000 MIPS;
- Tamanho das *Cloudlets* → As *cloudlets* no *data center* possuem 200.000, 300.000, 400.000, 500.000, 600.000 milhões de instruções em uma distribuição *round-robin*.

A Tabela 5.6 exemplifica como é essa distribuição para um conjunto de *cloudlets*. CID representa o ID da *cloudlet* e MI milhões de instruções. As máquinas virtuais do *data center* possuem capacidade de processamento de 500, 750 e 1000 MIPS em uma distribuição *round-robin*. Por exemplo, em uma simulação na qual 20 máquinas virtuais são geradas, 7 máquinas virtuais serão criadas com 500 MIPS, 7 máquinas virtuais terão 750 MIPS e 6 máquinas virtuais ficarão com 1000 MIPS. Além disso é adotado que cada máquina virtual possui 128 MB de RAM, 2,5 Mbps de largura de banda; cada máquina virtual tem tamanho de imagem de 2,5 GB.

Configurações para Data Centers Homogêneos

Para as simulações em *data centers* homogêneos, as seguintes configurações são adotadas:

CID	C_1	C_2	C_3	C_4	C_5	C_6
MI	200 k	300 k	400 k	500 k	600 k	200 k
Prioridade	Baixa	Média	Média	Alta	Baixa	Média

Tabela 5.6: Cenário 2: Exemplo de Distribuição de *Cloudlets* e Prioridades

Servidor	h_1	h_2	h_3	h_4	h_5	h_6	h_7
MIPS*Cores	1600*4	1860*4	3066*4	3300*4	3000*8	1600*4	1860*4
Potência (*watts*)	200	250	300	350	200	250	300

Tabela 5.7: Cenário 2: Exemplo de Configuração Inicial de Servidores em um Data Center Heterogêneo

- MIPS dos Servidores → Cada servidor em um *data center* homogêneo possui um processador baseado no Intel Xeon X5667, ou seja, um processador com 4 núcleos operando a 3,06 GHz e, portanto, segundo considerações adotadas na simulação, 3.066 MIPS por núcleo;
- Consumo de Potência dos Servidores → O consumo de potência para cada servidor em um *data center* homogêneo é de 250 *watts*.

Configurações para Data Centers Heterogêneos

Para simulações em *data centers* heterogêneos, as seguintes configurações são adotadas:

- MIPS dos Servidores → Cada servidor em um *data center* heterogêneo possui um processador, em distribuição *round-robin*, baseado em um dos seguintes modelos: Intel Xeon E5603, Intel Xeon E7520, Intel Xeon X5667, AMD Opteron 6204, AMD Opteron 6220; possuindo, respectivamente, 4 núcleos com 1.600 MIPS cada, 4 núcleos com 1.860 MIPS cada, 4 núcleos com 3.066 MIPS cada, 4 núcleos com 3.300 MIPS cada, 8 núcleos com 3.000 MIPS cada.
- Consumo de Potência dos Servidores → Cada servidor em um *data center* heterogêneo possui o seguinte consumo de potência em uma distribuição *round-robin*, 200, 250, 300 e 350 *watts*.

Está exemplificado na Tabela 5.7 como estão distribuídos os MIPS/potência dos servidores neste tipo de *data center*.

Resultados das Simulações

Trinta simulações foram realizadas para cada algoritmo, para cada um dos algoritmos e configurações possíveis mencionadas na Seção 5.3. As siglas que utilizaremos para os

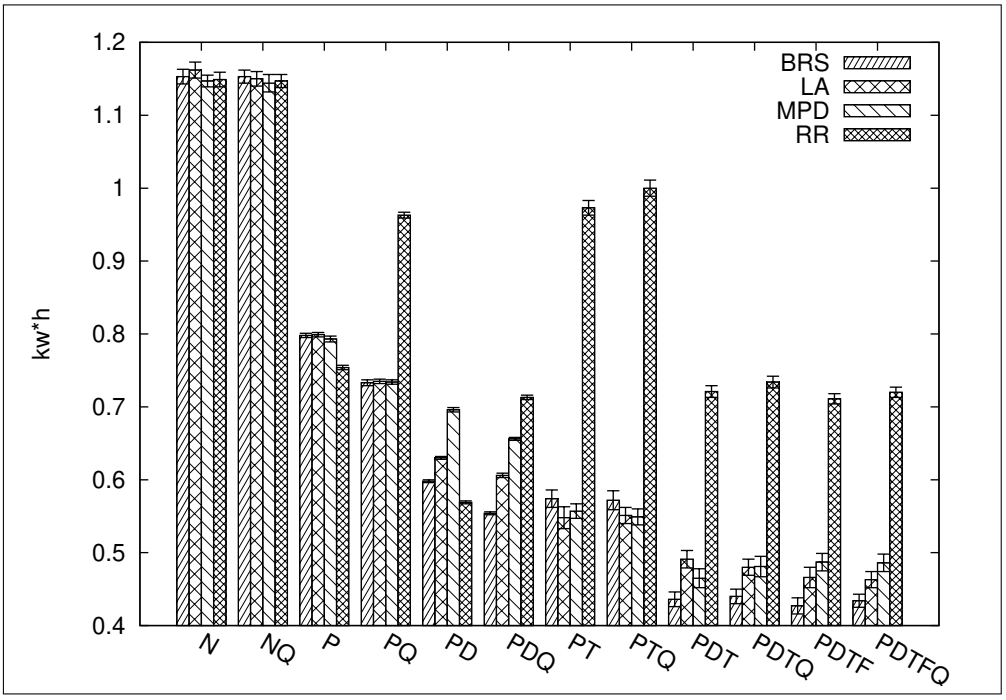


Figura 5.26: Cenário 2: Consumo de Energia em Data Center Pequeno Homogêneo

algoritmos e suas respectivas configurações são as mesmas apresentadas na Subseção 5.4.1.

Nesta bateria de simulações tentamos efetuar um maior número de comparação entre os resultados encontrados nos algoritmos, em especial no que diz respeito à qualidade de serviço. Conforme apresentado, o algoritmo que atua na submissão de cargas de trabalho, provendo qualidade de serviço, é independente do algoritmo de alocação de máquinas virtuais. Deste modo, nesta simulação efetuamos a implementação desta característica independentemente do algoritmo de escalonamento de máquinas virtuais. A sigla para esta configuração é definida como "Q", por exemplo, ao tratarmos do algoritmo BRS que possui como configurações $DVFS$ e provendo qualidade de serviço, este será referenciado como $BRS-DQ$. Para todos os valores de medição de energia apresentados nas próximas seções a métrica utilizada é $quilowatts \times hora$ e, para medidas de tempo, *segundos*. Para todos os gráficos apresentados nesta seção é adotado um intervalo de confiança de 95%.

Ressaltamos que o resultado desejável é a minimização do consumo de energia. Deve-se contemplar, também, a redução dos tempos de execução das cargas com alta prioridade, sem que esta acarrete um aumento substancial do consumo de energia. Além disso, os tempos de execução das cargas de baixas prioridades também não devem ser inaceitavelmente acentuados e que o modelo de distribuição de tarefas é o *bag-of-tasks*, onde um conjunto de tarefas é enviado ao escalonador, em seguida ele executa as tarefas e o processo é finalizado.

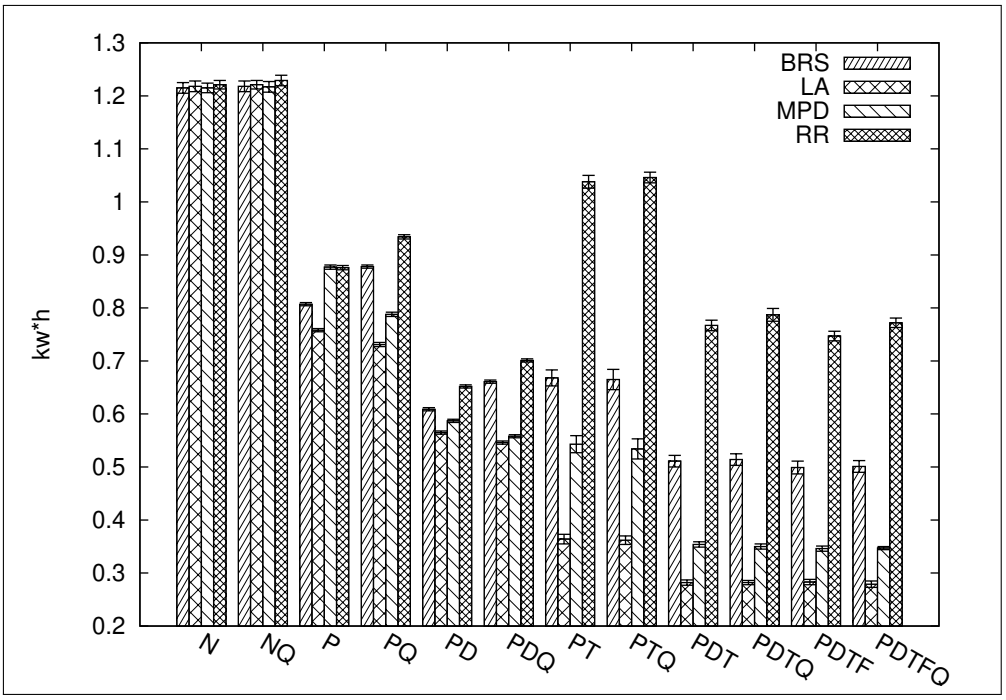


Figura 5.27: Cenário 2: Consumo de Energia em Data Center Pequeno Heterogêneo

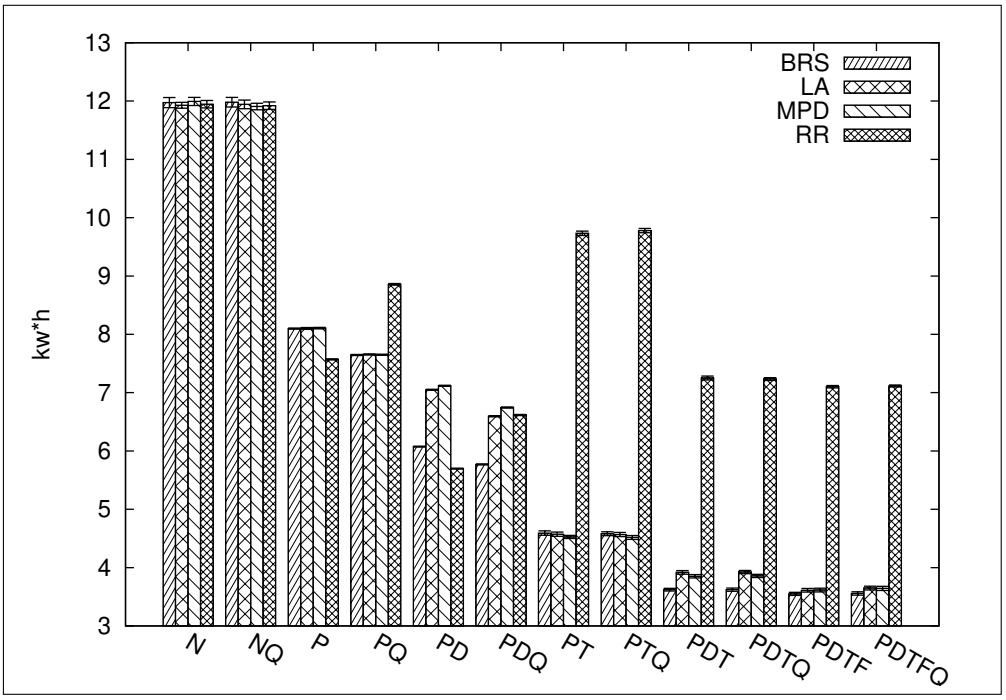


Figura 5.28: Cenário 2: Consumo de Energia em Data Center Médio Homogêneo

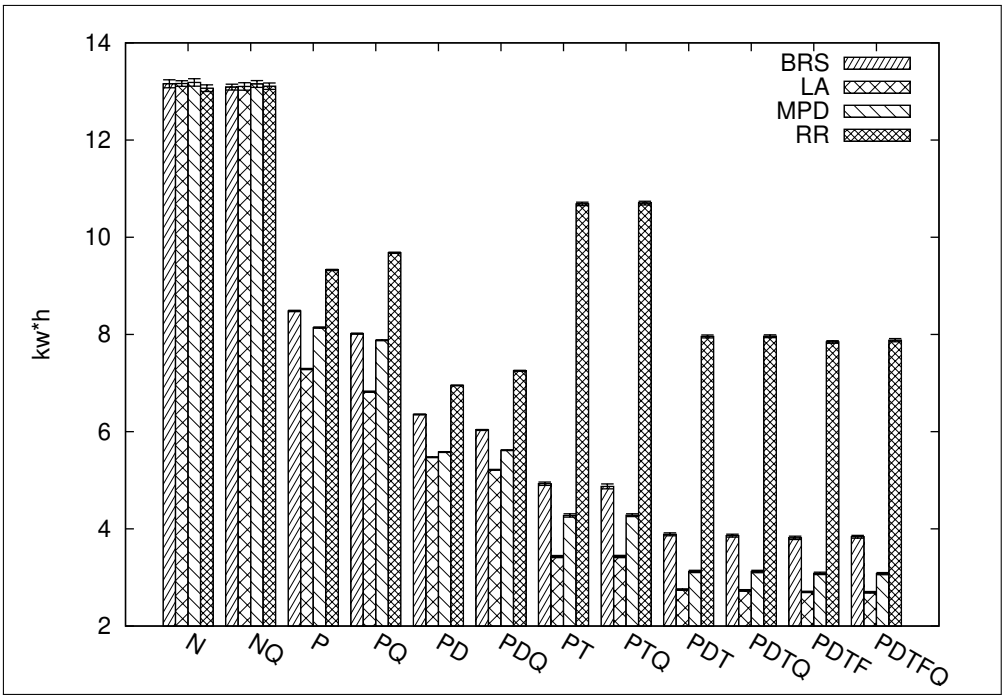


Figura 5.29: Cenário 2: Consumo de Energia em Data Center Médio Heterogêneo

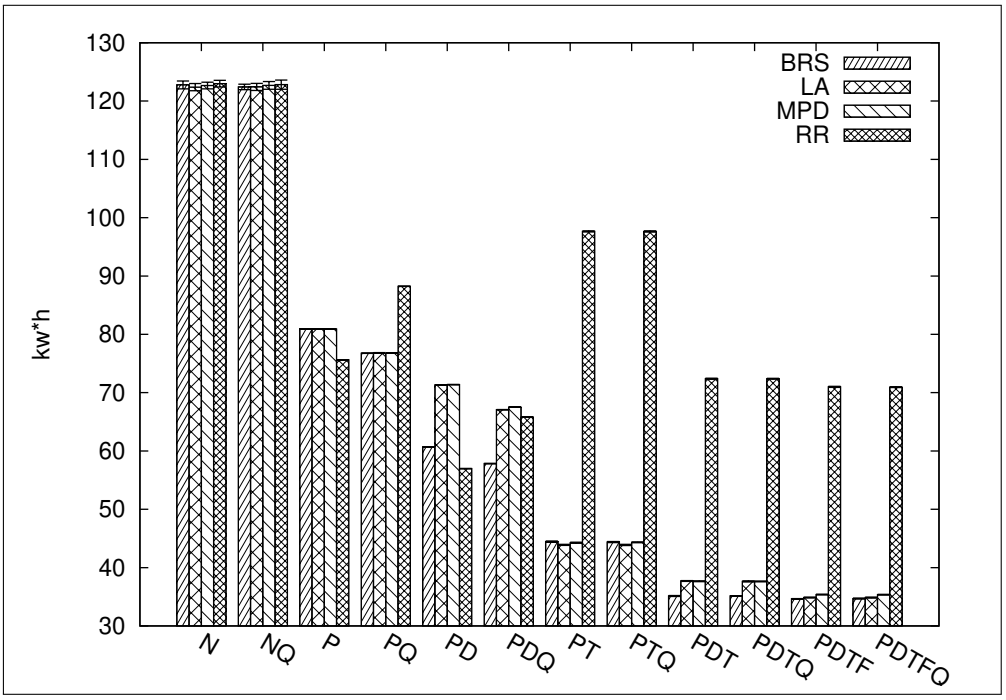


Figura 5.30: Cenário 2: Consumo de Energia em Data Center Grande Homogêneo

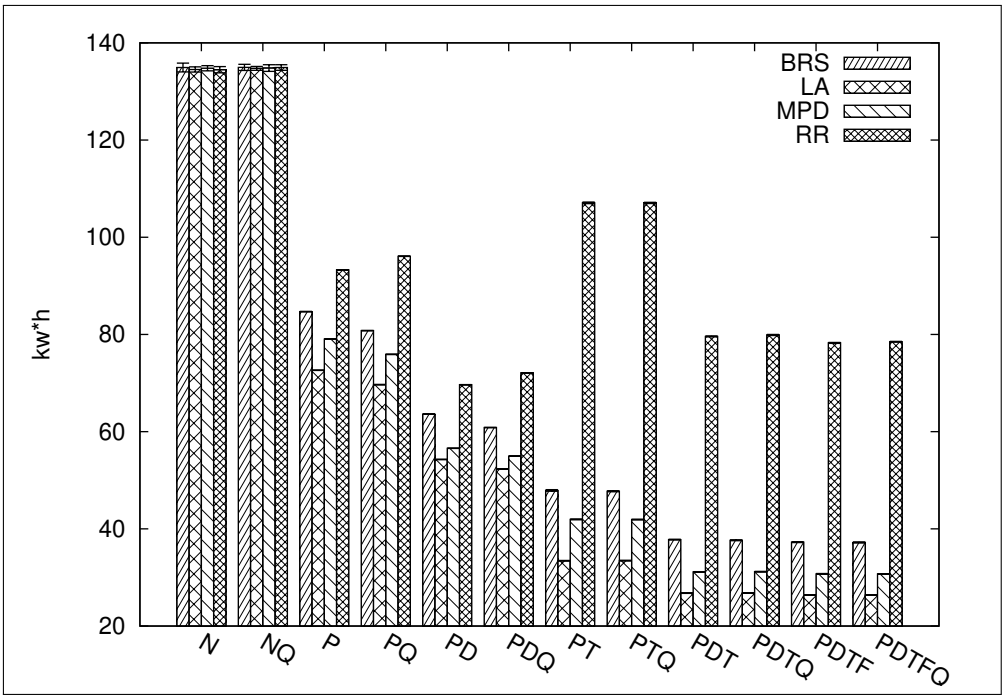


Figura 5.31: Cenário 2: Consumo de Energia em Data Center Grande Heterogêneo

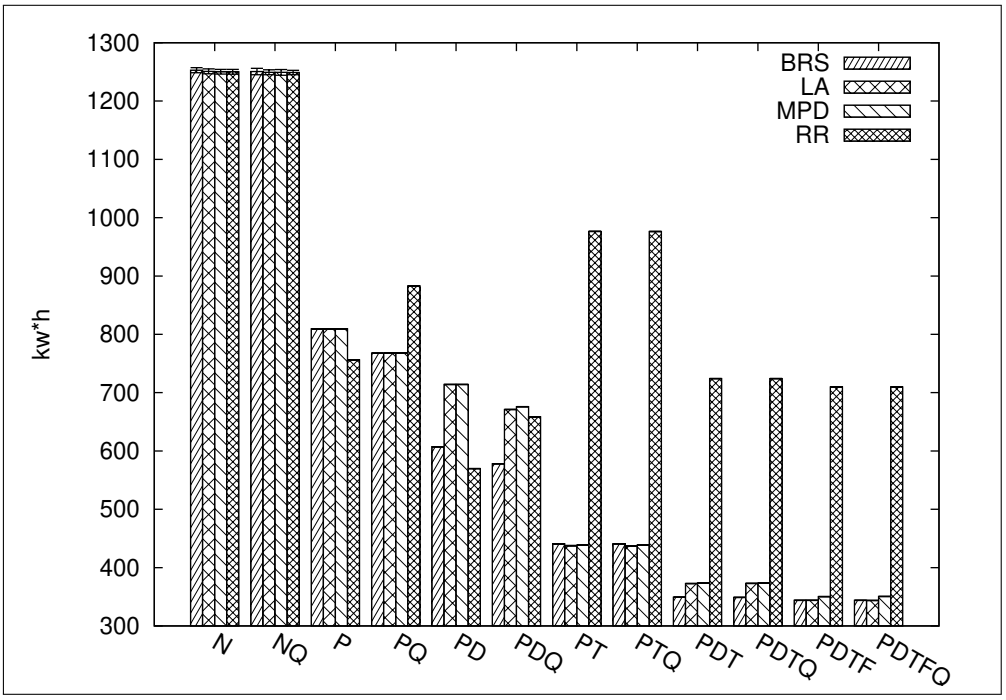


Figura 5.32: Cenário 2: Consumo de Energia em Data Center Gigante Homogêneo

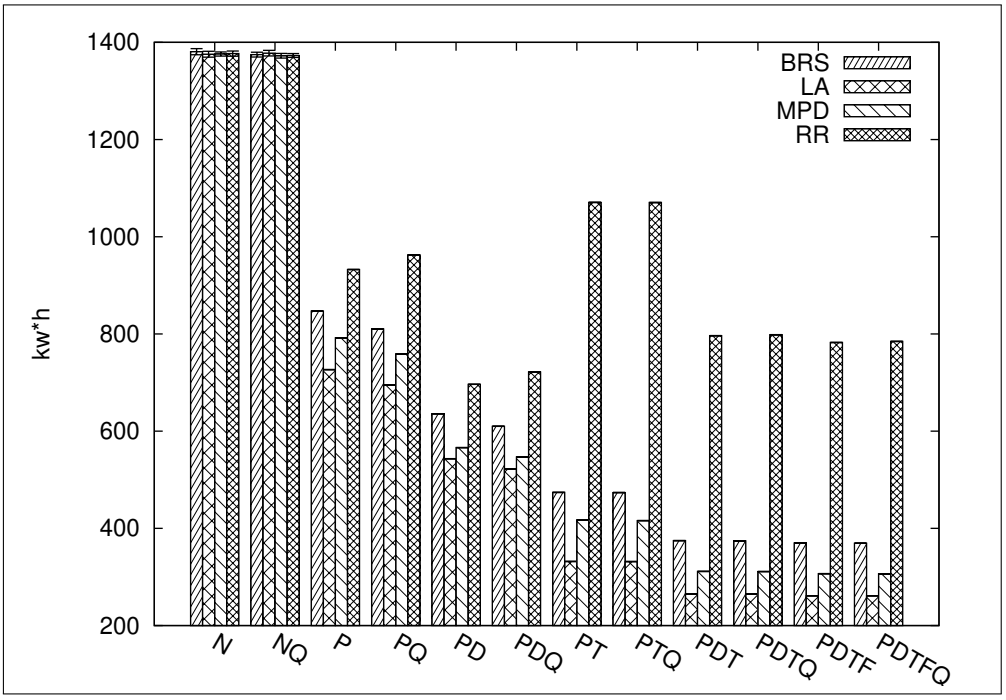


Figura 5.33: Cenário 2: Consumo de Energia em Data Center Gigante Heterogêneo

São apresentados nas Figuras 5.26 a 5.33 os resultados do consumo de energia dos algoritmos RR, BRS, MPD, LA, com configurações N, NQ, P, PQ, PD, PDQ, PT, PTQ, PDT, $PDTQ$, $PDTF$ e $PDTFQ$ resultantes das simulações. De forma geral o comportamento global é relativamente próximo, mas algumas diferenças são perceptíveis.

Comparados aos resultados de consumo de energia da bateria de simulações anterior, há um comportamento global semelhante entre os algoritmos. No entanto, uma diferença importante é percebida. Para nossa surpresa, com os parâmetros de grande escala e alto desempenho, o algoritmo BRS se comportou melhor em *data centers* homogêneos, em especial os que estavam com a configuração D habilitada, o que não ocorreu na bateria de testes anterior. Isso sugere que, para o caso de *data centers* homogêneos, os cálculos instantâneos durante a decisão de qual servidor hospedará uma determinada máquina virtual são menos importantes do que garantir uma minimização de migração de máquinas virtuais.

De fato, o BRS superou o LA em *data centers* homogêneos com configurações D, sendo mais econômico em média em 8,67% em *data centers* pequenos, 8,54% em médios, 8,19% em grandes e 7,87% em gigantes. Por outro lado, em *data centers* heterogêneos com configurações D o LA se mostrou superior, sendo mais econômico que o BRS em 58,13% em *data centers* pequenos, 32,99% em médios, 32,90% em grandes e 33,20% em gigantes. Considerando uma média de todas as configurações possíveis para *data centers*

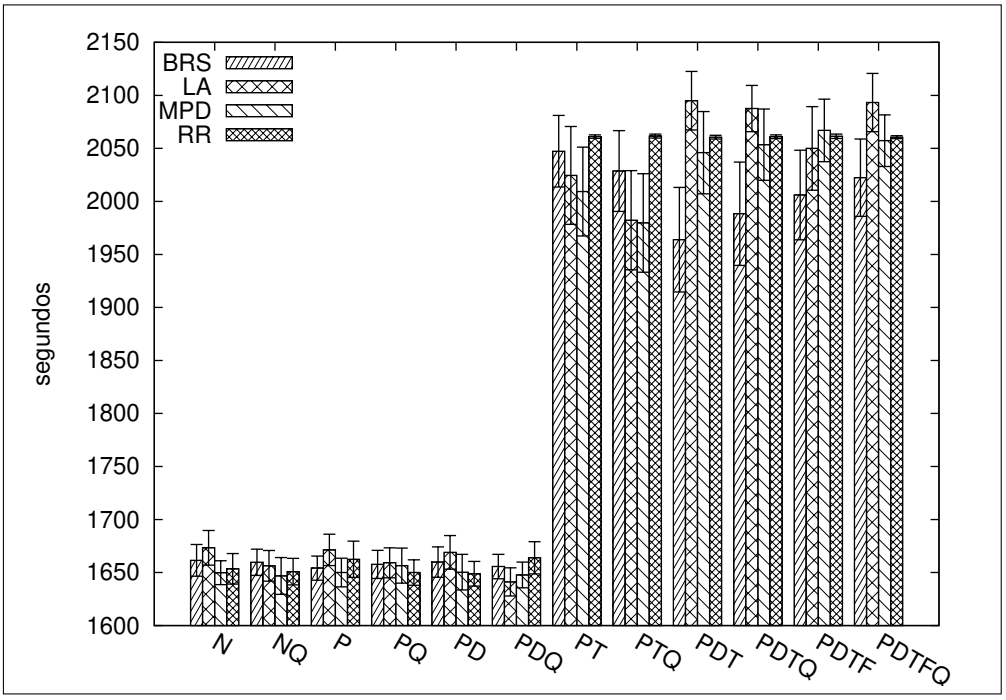


Figura 5.34: Cenário 2: Makespan em Data Center Pequeno Homogêneo

homogêneos estes valores diminuem um pouco: o BRS é mais econômico em apenas 3,71% em *data centers* pequenos, 4,17% em médios, 3,88% em grandes e 3,78% em gigantes. Já, considerando a média de todas as configurações possíveis para *data centers* heterogêneos, o LA manteve larga vantagem sendo em média 45,16% mais econômico nos pequenos, 26,45% em médios, 26,36% nos grandes e 26,5% nos gigantes.

Outro ponto relevante que foi possível perceber nos resultados apresentados é que a adição de qualidade de serviço, no processo de submissão de cargas de trabalho às máquinas virtuais, em qualquer um dos algoritmos apresentados, acarreta em alteração desprezível do consumo de energia. Isso implica que a adição deste tipo de qualidade de serviço, no âmbito da computação verde, pode ser empregado sem gerar malefícios substanciais.

São apresentados nas Figuras 5.34 a 5.41 os resultados do makespan dos algoritmos RR, BRS, MPD, LA, com configurações N, NQ, P, PQ, PD, PDQ, PT, PTQ, PDT, $PDTQ$, $PDTF$ e $PDTFQ$ resultantes das simulações.

Concernente ao comportamento de *makespan* dos algoritmos, os resultados destas baterias também apresentam um comportamento global muito próximo da bateria de simulações anterior. Vale ressaltar que o LA se comportou muito bem em todos os cenários, com todos os tipos de *data centers*, sendo regularmente melhor que os demais, exceto o RR, cujo uso não compensa devido ao elevado consumo de energia.

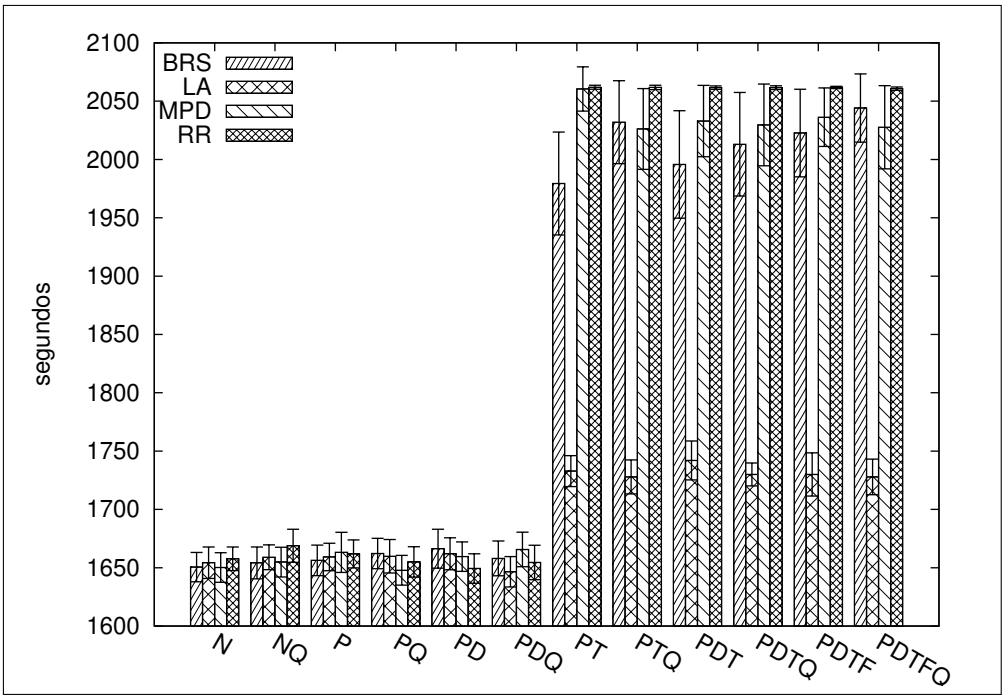


Figura 5.35: Cenário 2: Makespan em Data Center Pequeno Heterogêneo

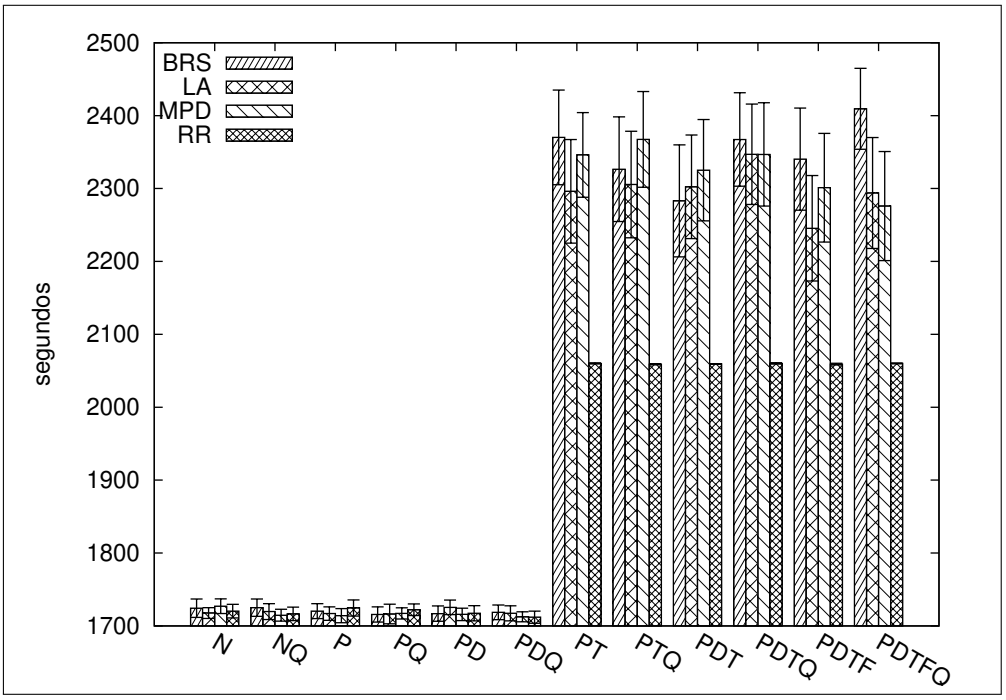


Figura 5.36: Cenário 2: Makespan em Data Center Médio Homogêneo

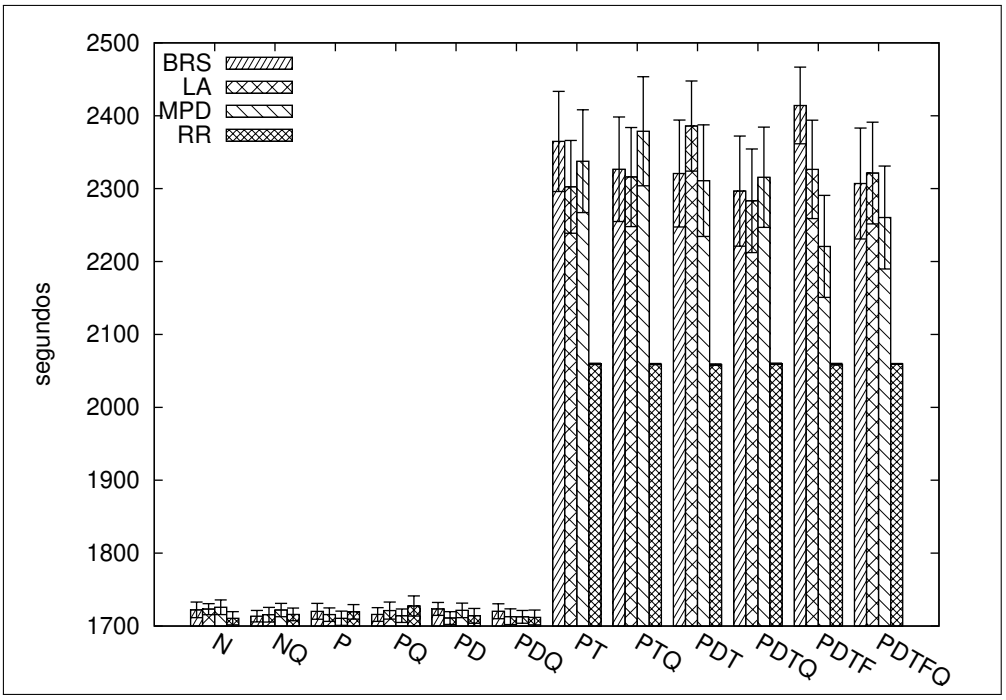


Figura 5.37: Cenário 2: Makespan em Data Center Médio Heterogêneo

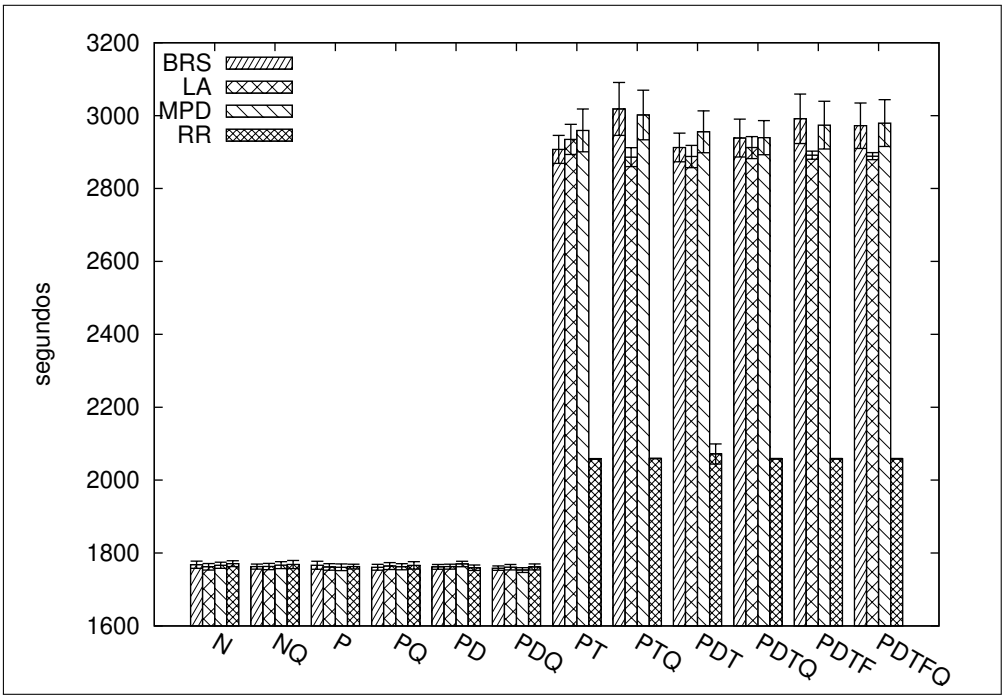


Figura 5.38: Cenário 2: Makespan em Data Center Grande Homogêneo

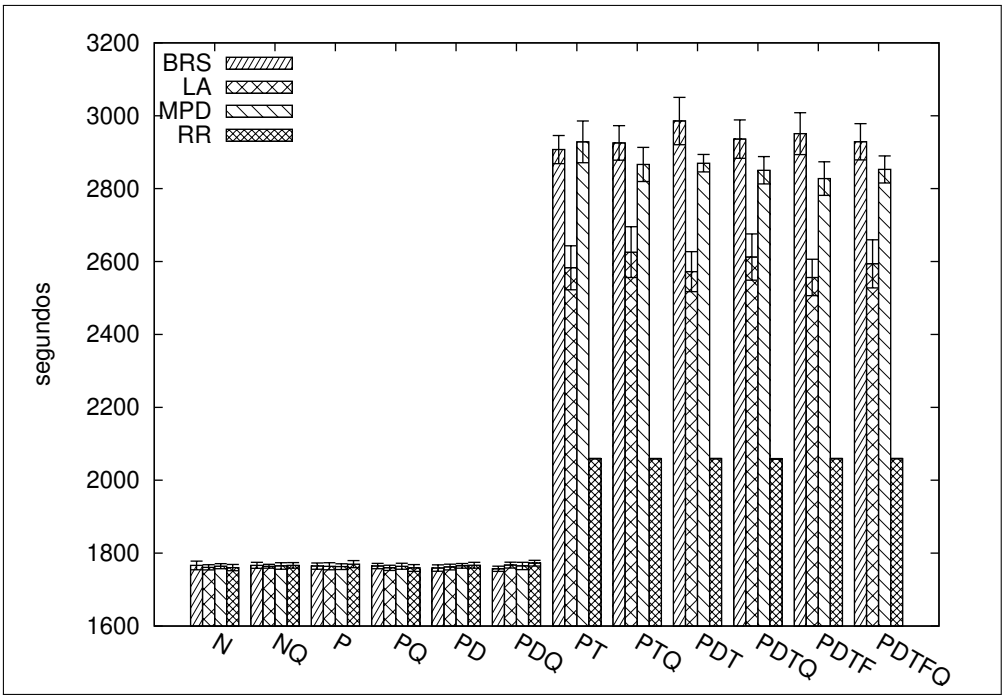


Figura 5.39: Cenário 2: Makespan em Data Center Grande Heterogêneo

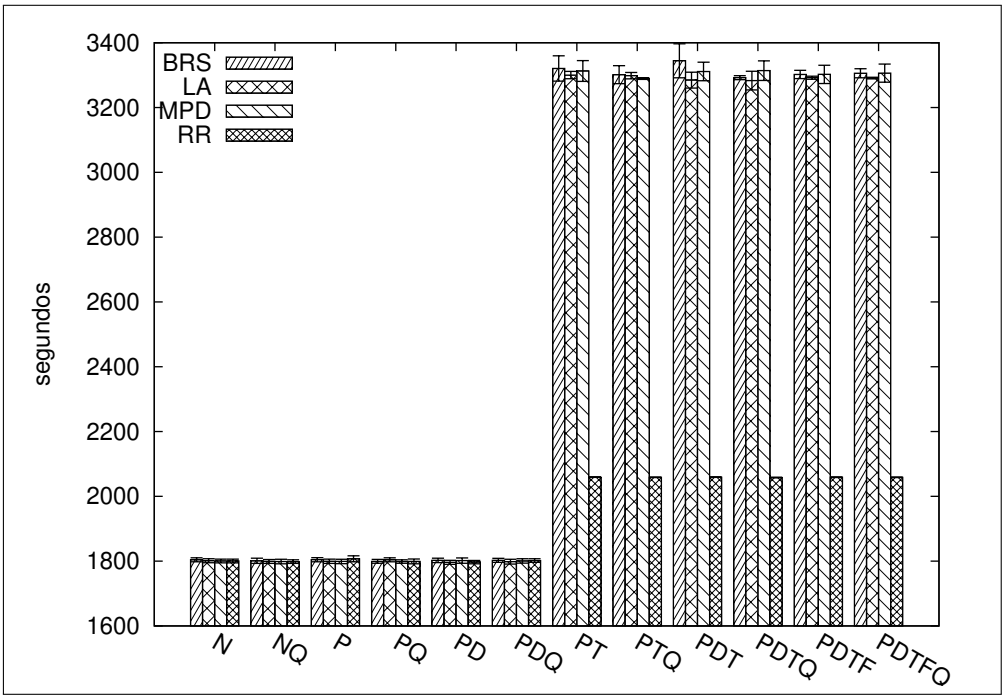


Figura 5.40: Cenário 2: Makespan em Data Center Gigante Homogêneo

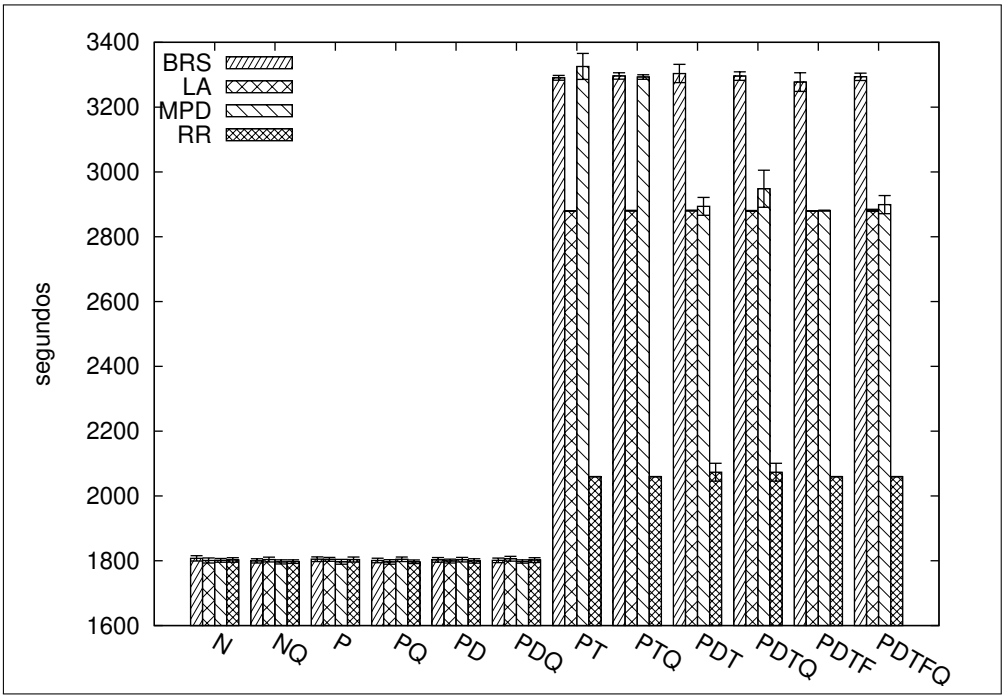


Figura 5.41: Cenário 2: Makespan em Data Center Gigante Heterogêneo

São apresentados nas Figuras 5.42 a 5.49 os resultados dos tempos médios de execução das cargas de trabalhos resultantes dos escalonamentos realizados pelos algoritmos RR, BRS, MPD, LA, com configurações N, NQ, P, PQ, PD, PDQ, PT, PTQ, PDT, $PDTQ$, $PDTF$ e $PDTFQ$ resultantes das simulações. Como é percebido, o comportamento global é semelhante.

Mesmo sob alterações de desempenho e escala o LA se manteve em um bom patamar.

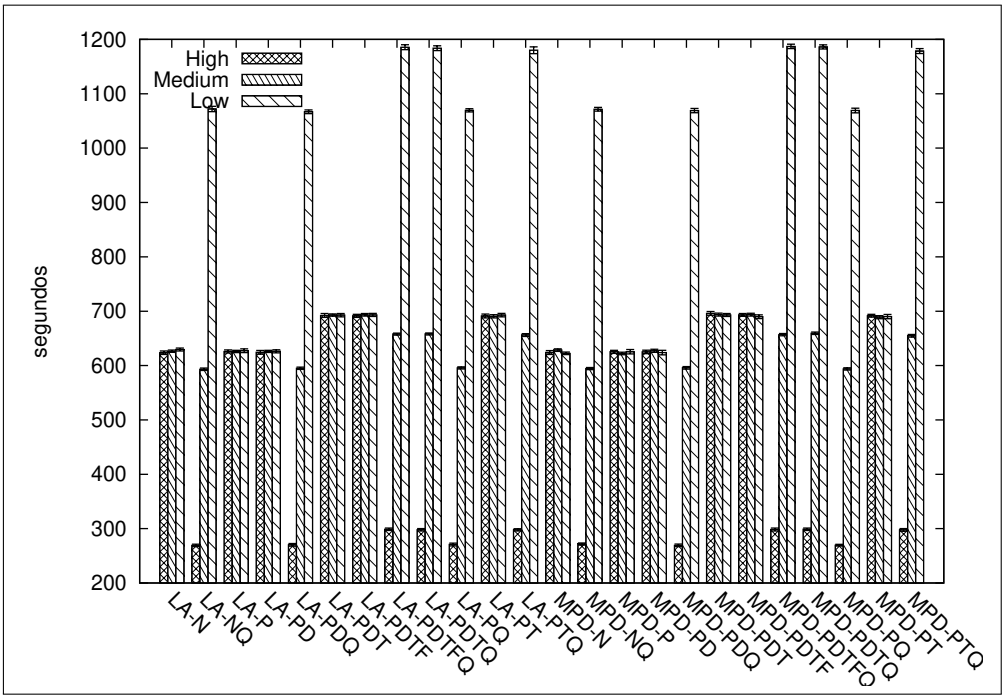


Figura 5.42: Cenário 2: Tempo Médio de Conclusão de Cloudlets em Data Center Pequeno Homogêneo

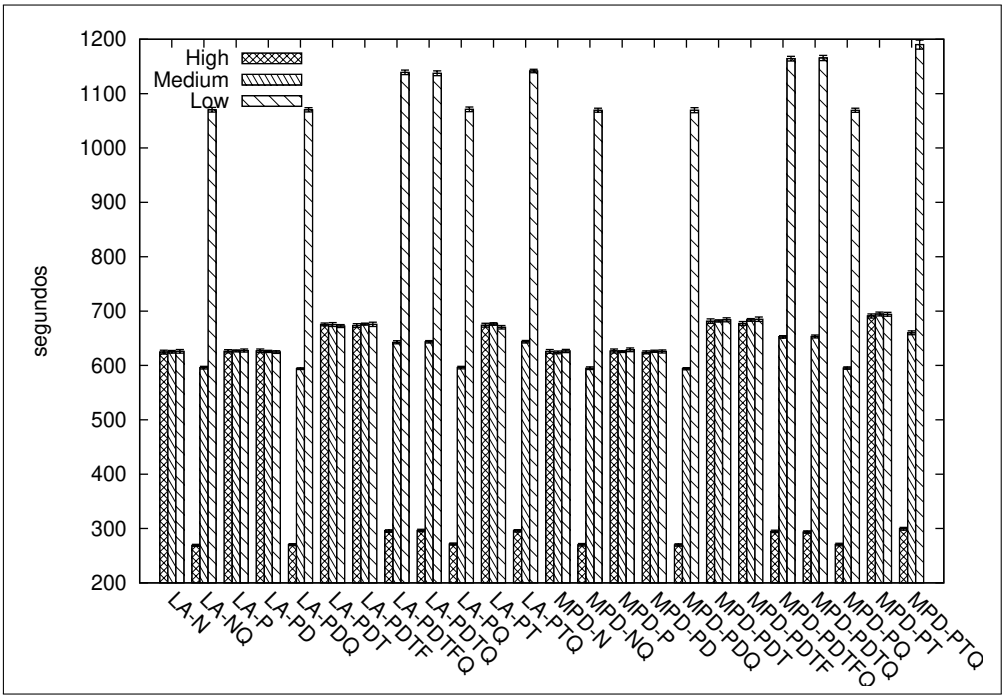


Figura 5.43: Cenário 2: Tempo Médio de Conclusão de Cloudlets em Data Center Pequeno Heterogêneo

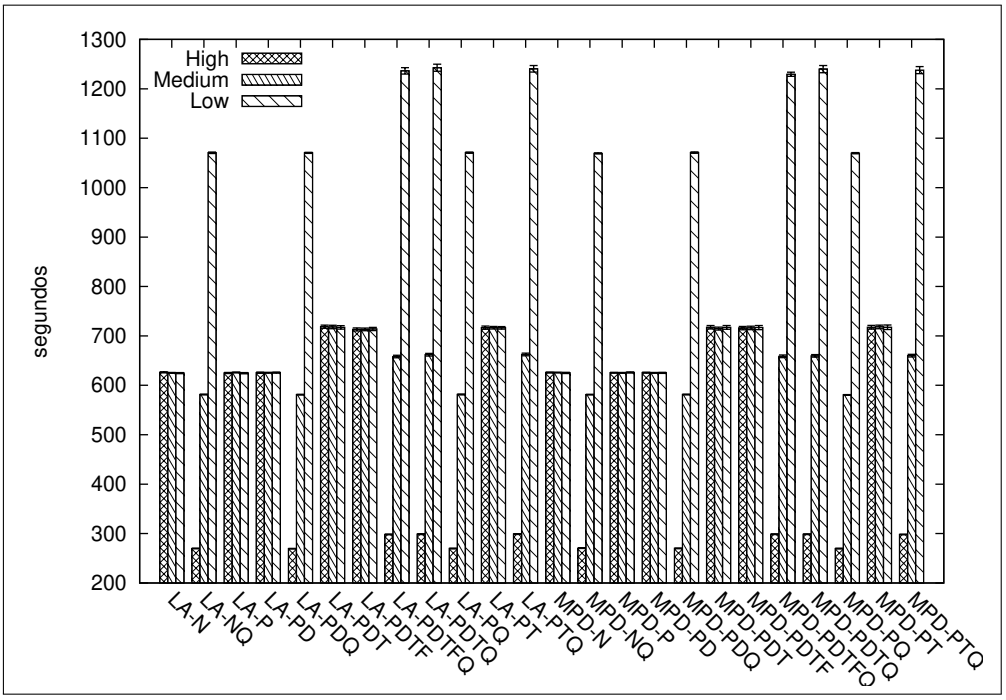


Figura 5.44: Cenário 2: Tempo Médio de Conclusão de Cloudlets em Data Center Médio Homogêneo

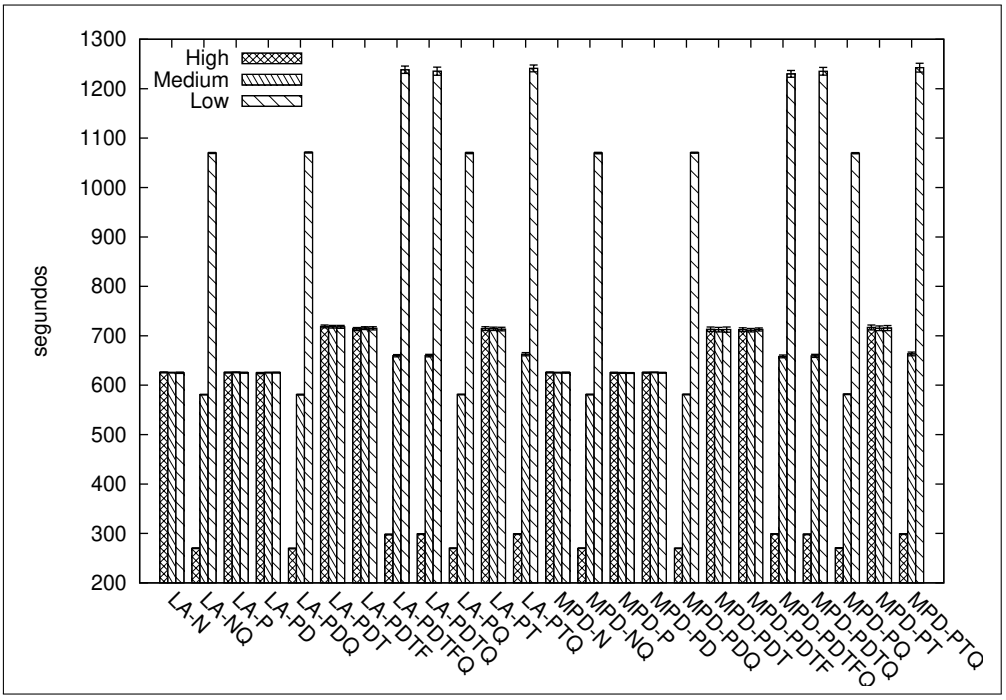


Figura 5.45: Cenário 2: Tempo Médio de Conclusão de Cloudlets em Data Center Médio Heterogêneo

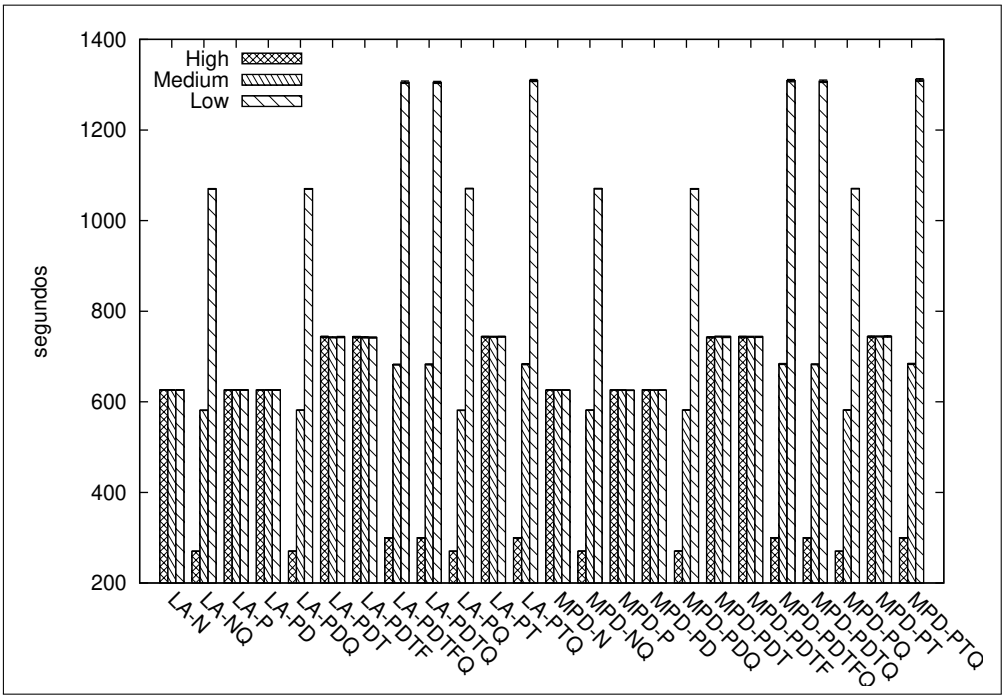


Figura 5.46: Cenário 2: Tempo Médio de Conclusão de Cloudlets em Data Center Grande Homogêneo

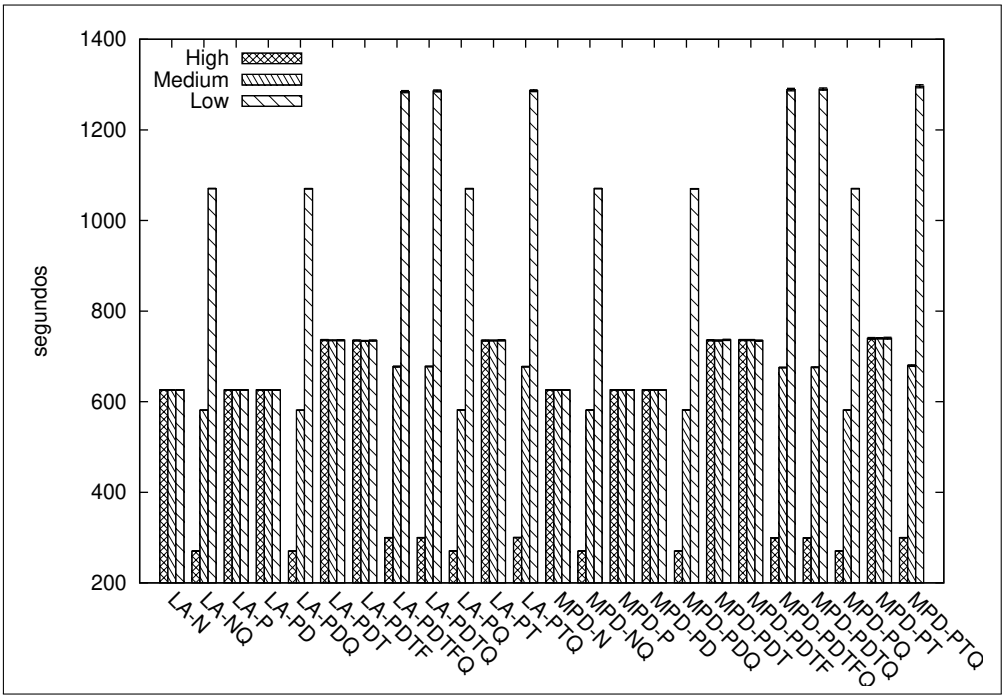


Figura 5.47: Cenário 2: Tempo Médio de Conclusão de Cloudlets em Data Center Grande Heterogêneo

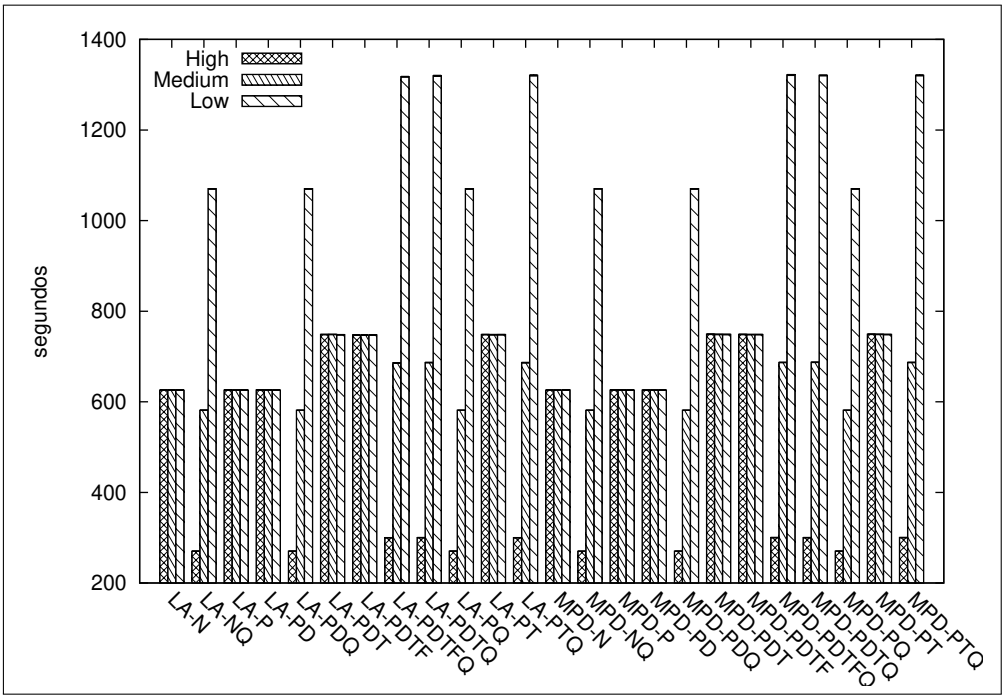


Figura 5.48: Cenário 2: Tempo Médio de Conclusão de Cloudlets em Data Center Gigante Homogêneo

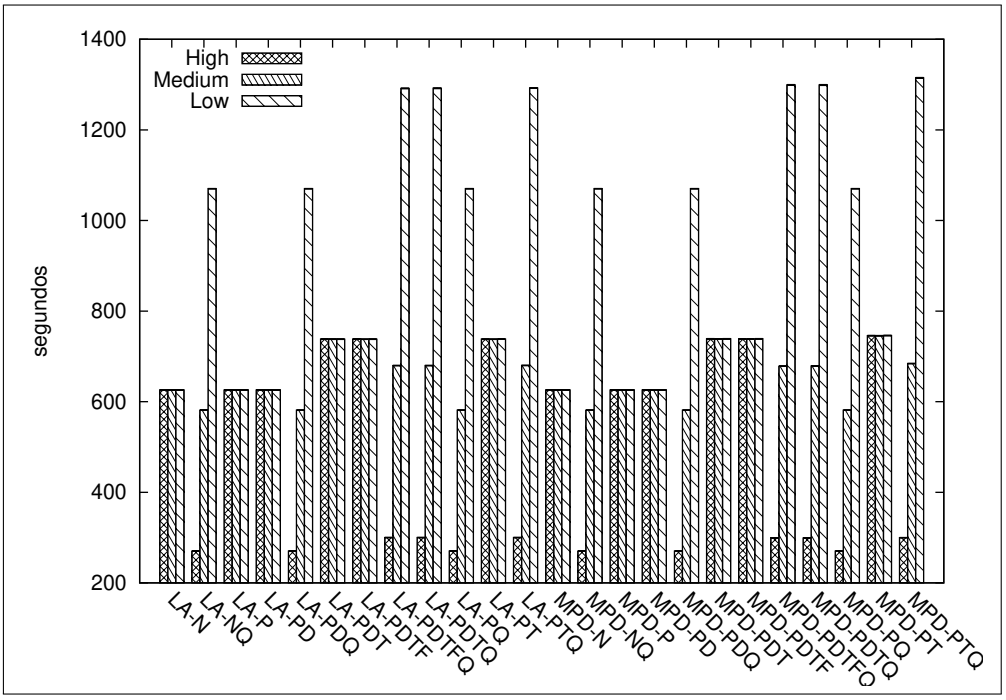


Figura 5.49: Cenário 2: Tempo Médio de Conclusão de Cloudlets em Data Center Gigante Heterogêneo

5.5 Análise de Resultados

Nesta seção são apresentados os resultados de algumas observações percebidas nas simulações.

É apresentado na Tabela 5.8 uma planilha contendo a redução média do consumo de energia por configurações de alto desempenho, considerando todos os algoritmos de escalonamento citados. Como pode-se observar, as configurações em si, independentemente dos algoritmos, têm um impacto diferenciado de acordo com o tamanho e configurações do *data center*. O menor impacto é observado em *data centers* pequenos e homogêneos, onde o impacto das configurações variam entre 31,83% e 54,65%. Os melhores ganhos obtidos com as configurações é quando é efetuada uma escala para *data centers* gigantes e heterogêneos, conseguindo uma redução de 68,78% no consumo de energia com as configurações $PDTF$.

Outro fato interessante constatado na referida tabela é que o dimensionamento dinâmico de tensão em frequência apresenta maior impacto em *data centers* pequenos, enquanto nos demais a migração de máquinas virtuais é de maior importância. No entanto, com ambos combinados em todos os casos foram obtidos melhores resultados. A ativação do mecanismo de *Fan Control* em cima das configurações completas trouxe pequena, mas significativa melhora.

Foi constatado também que a única configuração que impacta o *makespan* é o limiar de migração de máquinas virtuais. De fato, para todos os algoritmos, para os cenários de alta performance, a ativação da configuração T implicou em um aumento do tempo de processamento de 18,84% (em *data centers* pequenos e heterogêneos) a 66,5% (em *data centers* gigantes e homogêneos). Os valores colhidos sugerem que quanto maior o *data center*, maior é o impacto no incremento de tempo do *makespan* acarretado pelas migrações.

Um outro fato, surpreendente, foi que as configurações das máquinas em si impactaram e diferenciaram substancialmente os resultados. De fato, não era esperado inicialmente que escalar um *data center* de um modelo genérico para um modelo real de alto desempenho afetasse tanto o consumo de energia a ponto de tornar o BRS melhor que o LA em alguns casos. Apesar disso, conforme os resultados apresentados, esta melhoria é bastante limitada e, em contrapartida, no caso de *data centers* heterogêneos os ganhos obtidos com o uso de LA são muito maiores.

Existe uma métrica comumente usada na indústria denominada *Power Usage Effectiveness* (PUE) [79], utilizada para o cálculo de eficiência em uso de energia. O PUE para um *data center* pode ser calculado de acordo com a fórmula apresentada em 5.3.

$$PUE = \frac{TFP}{ITEP} \tag{5.3}$$

Tamanho	Configuração				
Data Center	P	PD	PT	PDT	$PDTF$
Homo. Pequeno	31,83%	45,93%	42,45%	54,18%	54,65%
Homo. Médio	33,37%	45,78%	51,03%	61,03%	62,62%
Homo. Grande	35,12%	46,94%	53,11%	62,75%	64,18%
Homo. Gigante	36,37%	47,94%	54,16%	63,63%	65,07%
Hetero. Pequeno	31,84%	50,44%	46,35%	60,69%	61,48%
Hetero. Médio	36,76%	53,65%	55,57%	66,24%	66,76%
Hetero. Grande	38,81%	54,69%	57,22%	67,45%	67,94%
Hetero. Gigante	40,13%	55,69%	58,37%	68,28%	68,78%

Tabela 5.8: Redução Média do Consumo de Energia por Configurações em Data Centers de Alto Desempenho (valores maiores são melhores)

Onde TFP (*Total Facility Power*) é a potência total consumida pelo *data center* (incluindo instalações e equipamentos) e $ITEP$ (*IT Equipment Power*) é a potência consumida pelos equipamentos de TI. Além disso existem categorias nas quais o PUE se encaixa, dependendo do tipo proveniente da sua fonte energética. Por exemplo, se a categoria de cálculo do PUE for 0, isto implica que os *data centers* que serão avaliados são totalmente elétricos, não consumindo outros tipos de energia (como gás natural, etc). Esta é uma métrica que se aplica normalmente a ambientes reais, tendo em vista que o ambiente de simulação não considera o consumo de energia das instalações, do tipo de resfriamento, entre outras características de consumo de energia de *data centers*. O cálculo de PUE não se aplica a este trabalho, devido a dependência de características de infraestrutura de *data centers* não contempladas em seu escopo. Pretende-se, em trabalhos futuros, considerar cenários nos quais são aplicáveis tal métrica.

Capítulo 6

Conclusão

O uso ineficiente de servidores em um *data center* acarreta altos custos de energia, equipamentos para refrigeração, espaço físico, além de diversos impactos no ambiente [14]. Neste trabalho foram apresentados algoritmos de escalonamento para computação em nuvem baseados em conceitos de computação verde. Estes algoritmos são capazes de realizar o escalonamento de tarefas em *data centers* não-federados, homogêneos e heterogêneos, com tamanhos de cargas de trabalho conhecidos ou não.

Além disso os algoritmos propostos permitem o uso de qualidade de serviço baseada em prioridades de cargas, focando a minimização dos tempos de execução de cargas com alta prioridade, sem impactar negativamente o consumo de energia dos *data centers*. Para atingir a referida qualidade de serviço, é efetuado um ranqueamento das cargas de trabalho de acordo com suas prioridades durante o processo de escalonamento, permitindo que cargas de alta prioridade sejam executadas mais rapidamente.

Resultados obtidos por simulação sugerem que os algoritmos LA e $LA-Q$ competem com os melhores algoritmos pesquisados em *data centers* homogêneos e, em todos os casos simulados, superando-os em *data centers* heterogêneos. Um fator constatado foi que, no caso específico de *data centers* homogêneos e de alto desempenho, o algoritmo clássico BRS apresentou resultados de consumo de energia melhores que o MPD e LA, se aproveitando melhor de $DVFS$. Os resultados também mostraram que o uso de controle de refrigeração ativo produz uma pequena, mas relevante melhoria.

Com relação à qualidade de serviço, os resultados obtidos indicam que o modelo de gerenciamento de prioridade de cargas adotado nesse trabalho permitiu reduzir o tempo de processamento de cargas com alta prioridade em mais de 40%. As cargas normais sofreram pouco impacto nas simulações, e as de baixa prioridade apresentaram um incremento no seu tempo de execução, todavia aceitável dada a natureza do tipo da carga. Mais do que isso, a qualidade de serviço desenvolvida é independente e pode ser acoplada a qualquer algoritmo, e não apenas ao LA. Os resultados obtidos mostraram que a qualidade

de serviço não impacta de forma significativa os princípios da computação verde, sendo recomendada, portanto, em todos os cenários analisados.

Em suma pode-se afirmar que o *Lago Allocator* é um algoritmo para escalonamento de computação em nuvem recomendável quando comparado aos demais, e que a adição de qualidade de serviço impacta muito pouco o seu consumo de energia, não modifica o *makespan* das cargas de trabalho, mas reduz substancialmente o tempo de execução das cargas que possuem maiores prioridades.

Além das supracitadas contribuições ao escalonamento com foco na minimização do consumo de energia, como a análise do impacto da adição da qualidade de serviço, redução do consumo especialmente em *data centers* heterogêneos, este trabalho também apresentou uma estratégia diferenciada para redução da migração de máquinas virtuais, uso efetivo de *DVFS* e aplicação do mecanismo de *Fan Control*, também contribuindo de forma significativa para redução do consumo.

É proposto como trabalho futuro adaptar os algoritmos apresentados para lidar com *data centers* federados. Também pretende-se estudar e avaliar heurísticas, como as aplicadas ao problema do empacotamento, para otimizar o algoritmo para o caso onde cargas virtuais possuem tamanhos estimáveis. Aspira-se, também, o estudo de técnicas para otimização dinâmica do valor do limiar de migração de máquinas virtuais, tornando a nuvem mais eficiente no consumo de energia. Pode-se, ainda, lidar com cenários de falhas, onde a máquina que hospeda o escalonador que executa os algoritmos propostos neste trabalho ou demais máquinas do *data center* falham durante o processo de escalonamento, além de casos onde uma única máquina virtual processa múltiplas cargas de trabalho. Também pretende-se abordar cenários que considerem aspectos da infraestrutura de *data centers* que permitam o cálculo de outras métricas aplicáveis ao contexto de produção, como *PUE*.

Referências Bibliográficas

[1] Luiz Bezerra. Cloud computing. Tecnologia e Gestão. http://tecnologiaegestao.wordpress.com/2010/06/28/cloud-computing/, 2010. Acessado em 05/01/2012.

[2] Peter Mell and Tim Grance. The nist definition of cloud computing. *National Institute of Standards and Technology*, 53(6):50, 2009.

[3] Institute of Electrical and Electronics Engineers. The 2012 ieee fifth international conference on cloud computing (cloud 2012), 2012.

[4] Matt Asay. 2010 the year of cloud-computing. CNET News. http://news.cnet.com/8301-13505_3-10422528-16.html, 2009. Acessado em 05/01/2012.

[5] Microsoft. Windows azure plataform. http://www.microsoft.com/windowsazure/, Acessado em 05/01/2012.

[6] Google. App engine. http://code.google.com/appengine/, Acessado em 05/01/2012.

[7] Amazon. Amazon elastic compute cloud (Amazon EC2). http://aws.amazon.com/ec2/, Acessado em 05/01/2012.

[8] Stephan Groß and Alexander Schill. Towards user centric data governance and control in the cloud. In Jan Camenisch and Dogan Kesdogan, editors, *Open Problems in Network Security*, volume 7039 of *Lecture Notes in Computer Science*, pages 132–144. Springer Berlin / Heidelberg, 2012. 10.1007/978-3-642-27585-2_11.

[9] Jonathan G. Koomey. Estimating total power consumption by servers in the U.S. and the world. Technical report, Lawrence Derkley National Laboratory, 2007.

[10] Planet Virtual. Green computing, helping save the planet. http://www.planet-virtual.com/green-computing. Planet Virtual, LLC, 2010. Acessado em 05/01/2012.

[11] Greenpeace. Make it green – cloud computing and its contribution to climate change. http://www.greenpeace.org/international/en/publications/reports/make-it-green-cloud-computing/. Greenpeace International, 2010. Acessado em 05/01/2012.

[12] Cheng-Jen Tang, Miau-Ru Dai, Hui-Chin He, and Chi-Cheng Chuang. Evaluating energy efficiency of data centers with generating cost and service demand. *Bulletin of Networking, Computing, Systems, and Software*, 1(1), 2012.

[13] Laurent Lefèvre and Jean-Marc Pierson. Energy savings in ict and ict for energy savings. *ERCIM NEWS número 79, Towards Green ICT, pg. 10411*, 2009.

[14] Gargi Dasgupta, Amit Sharma, Akshat Verma, Anindya Neogi, and Ravi Kothari. Workload management for power efficiency in virtualized data centers. *Commun. ACM*, 54:131–141, July 2011.

[15] R. Buyya, R. Ranjan, and R.N. Calheiros. Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities. In *International Conference on High Performance Computing & Simulation, 2009. HPCS'09.*, pages 1–11. IEEE, 2009.

[16] Eric Knorr and Galen Gruman. What cloud computing really means. InfoWorld. http://www.infoworld.com/d/cloud-computing/what-cloud-computing-really-means-031, 2008. Acessado em 07/01/2012.

[17] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55, December 2008.

[18] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.

[19] Vivek Kundra. Federal cloud computing strategy. *Office of the CIO, Whitehouse.*

[20] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *J. Internet Services and Applications*, 1(1):7–18, 2010.

[21] Luiz F. Bittencourt and Edmundo R. M. Madeira. Hcoc: a cost optimization algorithm for workflow scheduling in hybrid clouds. *Journal of Internet Services and Applications*, 2:207–227, 2011.

[22] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, pages 273–286, Berkeley, CA, USA, 2005. USENIX Association.

[23] VMware. vmotion for live migration of virtual machines. http://www.vmware.com/products/vmotion/, 2010. Acessado em 07/01/2012.

[24] S. Murugesan. Harnessing green it: Principles and practices. *IT Professional*, 10(1):24 –33, jan.-fev. 2008.

[25] Michael Dell. Setting a higher bar for climate change. Forbes Magazine. http://www.forbes.com/2009/12/02/copenhagen-energy-efficiency-technology-cio-network-michael-dell.html, 2009. Acessado em 07/01/2012.

[26] D. Wang. Meeting green computing challenges. In *International Symposium on High Density packaging and Microsystem Integration, 2007. HDP '07*, pages 1 –4, june 2007.

[27] OECD Working Party on the Information Economy. Towards green ict strategies: Assessing policies and programmes on icts and the environment. Organisation for Economic Co-operation and Development. http://oecd.org/dataoecd/47/12/42825130.pdf, 2009. Acessado em 07/01/2012.

[28] The Green Grid. Acessado em 07/01/2012. http://www.thegreengrid.org/, 2010.

[29] AMD Technical Whitepaper. Acp – the truth about power consumption starts here. Silicon Graphics International. http://www.sgi.com/partners/technology/downloads/43761D-ACP_PowerConsumption.pdf, 2010. Acessado em 07/01/2012.

[30] LLC Oakland Solutions. Computer hardware problems. http://www.oaklandsolutionsllc.com/computer-problems.htm, 2012. Acessado em 17/02/2012.

[31] Eduardo Pinheiro, Wolf dietrich Weber, and Luiz André Barroso. Failure trends in a large disk drive population. In *Proceedings of the 5th USENIX Conference on File and Storage Technologies*, 2007.

[32] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. *SIGOPS Oper. Syst. Rev.*, 37:164–177, 2003.

[33] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.*, 25:599–616, June 2009.

[34] William Voorsluys, James Broberg, Srikumar Venugopal, and Rajkumar Buyya. Cost of virtual machine live migration in clouds: A performance evaluation. In *Proceedings of the 1st International Conference on Cloud Computing*, CloudCom '09, pages 254–265, Berlin, Heidelberg, 2009. Springer-Verlag.

[35] Makoto Sakai (Tokyo, JP). Acpi sleep control, July 2001. Patent Number 6266776.

[36] Bui, Sang T. (Irvine, CA, US). Method and apparatus for performing wake on lan power management, Maio 2006. Patent Number 7047428.

[37] Jan M. Rabaey. *Digital integrated circuits: a design perspective*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[38] Intel. Enhanced intel speedstep® technology how to document. http://www.intel.com/cd/channel/reseller/asmo-na/eng/203838.htm, 2010. Acessado em 07/01/2012.

[39] AMD. Amd cool'n'quiet technology. http://www.amd.com/us/products/technologies/cool-n-quiet/Pages/cool-n-quiet.aspx, 2010. Acessado em 07/01/2012.

[40] Mike Chin. Review: Silenttek - aopen's mobo-embedded fan controller. http://www.silentpcreview.com/article64-page1.html, 2002. Acessado em 07/01/2012.

[41] ASUS. Q-fan smart cooling system enables quieter and more efficient fan operation. http://www.asus.com/999/html/events/mb/qfan.htm, 2008. Acessado em 07/01/2012.

[42] MSI. Dual core center. http://eu.msi.com/service/download/utility-5256.html, 2010. Acesso em 08/01/2012.

[43] Abit. μguru panel. http://www.abit.com.tw/upload/products/guru-panel.htm, 2010. Acessado em 07/01/2012.

[44] Gigabyte. Easy tune6. http://www.gigabyte.com/support-downloads/utility.aspx?cg=2, 2010. Acessado em 07/01/2012.

[45] Intel. Active monitor. http://www.intel.com/design/motherbd/active.htm, 2010. Acessado em 07/01/2012.

[46] Intel. Desktop utilities. http://www.intel.com/design/motherbd/software/idu/, 2010. Acessado em 07/01/2012.

[47] Christian Diefer. Dell inspiron/latitude/precision fan control utility. http://www.diefer.de/i8kfan/, 2007. Acessado em 07/01/2012.

[48] G. I. Meijer. Cooling Energy-Hungry Data Centers. *Science*, 328(5976):318–319, 2010.

[49] Chandrakant D. Patel, Cullen E. Bash, Ratnesh Sharma, Monem Beitelmal, and Rich Friedrich. Smart cooling of data centers. *ASME Conference Proceedings*, 2003(36908b):129–137, 2003.

[50] Ehsan Pakbaznia and Massoud Pedram. Minimizing data center cooling and server power costs. In *Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design*, ISLPED '09, pages 145–150, New York, NY, USA, 2009. ACM.

[51] Anton Beloglazov and Rajkumar Buyya. Energy efficient resource management in virtualized cloud data centers. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, CCGRID '10, pages 826–831, Washington, DC, USA, 2010. IEEE Computer Society.

[52] A Berl, E Gelenbe, M Di Girolamo, G Giuliani, H De Meer, M Q Dang, and K Pentikousis. Energy-efficient cloud computing. *The Computer Journal*, 53(7):1045–1051, 2009.

[53] Saurabh Kumar Garg, Chee Shin Yeo, Arun Anandasivam, and Rajkumar Buyya. Environment-conscious scheduling of hpc applications on distributed cloud-oriented data centers. *J. Parallel Distrib. Comput.*, 71:732–749, June 2011.

[54] Walter Binder and Niranjan Suri. Green computing: Energy consumption optimized service hosting. In *Proceedings of the 35th Conference on Current Trends in Theory and Practice of Computer Science*, SOFSEM '09, pages 117–128, Berlin, Heidelberg, 2009. Springer-Verlag.

[55] Ripal Nathuji and Karsten Schwan. Virtualpower: coordinated power management in virtualized enterprise systems. *SIGOPS Oper. Syst. Rev.*, 41:265–278, 2007.

[56] Liting Hu, Hai Jin, Xiaofei Liao, Xianjie Xiong, and Haikun Liu. Magnet: A novel scheduling policy for power reduction in cluster with virtual machines. In *IEEE International Conference on Cluster Computing, 2008*, pages 13 –22, 29 2008-out. 1 2008.

[57] Rerngvit Yanggratoke, Fetahi Wuhib, and Rolf Stadler. Gossip-based resource allocation for green computing in large clouds. In *7th International Conference on Network and Service Management (CNSM), 2011*, pages 1 –9, out. 2011.

[58] Truong Vinh Truong Duy, Y. Sato, and Y. Inoguchi. Performance evaluation of a green scheduling algorithm for energy savings in cloud computing. In *IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010*, pages 1 –8, abril 2010.

[59] Shekhar Srikantaiah, Aman Kansal, and Feng Zhao. Energy aware consolidation for cloud computing. In *Proceedings of the 2008 conference on Power aware computing and systems*, HotPower'08, pages 10–10, Berkeley, CA, USA, 2008. USENIX Association.

[60] Luciano Bertini, Julius C. B. Leite, and Daniel Mossé. Power optimization for dynamic configuration in heterogeneous web server clusters. *J. Syst. Softw.*, 83:585–598, April 2010.

[61] Meng Xu, Lizhen Cui, Haiyang Wang, and Yanbing Bi. A multiple qos constrained scheduling strategy of multiple workflows for cloud computing. In *IEEE International Symposium on Parallel and Distributed Processing with Applications, 2009*, pages 629 –634, ago. 2009.

[62] Daniel Guimaraes do Lago, Edmundo R. M. Madeira, and Luiz Fernando Bittencourt. Power-aware virtual machine scheduling on clouds using active cooling control and dvfs. In *Proceedings of the 9th International Workshop on Middleware for Grids, Clouds and e-Science*, MGC '11, pages 2:1–2:6, New York, NY, USA, 2011. ACM.

[63] K.Mukherjee and G.Sahoo. Article:green cloud: An algorithmic approach. *International Journal of Computer Applications*, 9(9):1–6, November 2010. Published By Foundation of Computer Science.

[64] Daniel Lago, Edmundo Madeira, and Luiz Fernando Bittencourt. Escalonamento com prioridade na alocação ciente de energia de máquinas virtuais em nuvens. In *SBRC 2012*, Ouro Preto, MG, abr 2012.

[65] Rajkumar Buyya and Manzur Murshed. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE (CCPE)*, 14(13):1175–1220, 2002.

[66] VMware. Vmware® distributed power management concepts and use technical whitepaper. http://www.vmware.com/files/pdf/Distributed-Power-Management-vSphere.pdf, 2010. Acessado em 17/05/2012.

[67] Intel Corporation. Enhanced intel speedstep technology for the intel pentium m processor. Technical Whitepaper, 2004.

[68] Inc. Advanced Micro Devices. Amd opteron 6200 series processors specifications. http://www.amd.com/us/products/server/processors/6000-series-platform/6200/Pages/6200-series-processors.aspx#5, Acessado em 26/07/2012.

[69] Intel Corporation. Intel xeon processor 7000 sequence. http://ark.intel.com/products/family/34348, Acessado em 26/07/2012.

[70] Intel Corporation. Intel xeon processor e7 family. http://ark.intel.com/products/family/59139, Acessado em 26/07/2012.

[71] Po-Kuan Huang and Soheil Ghiasi. Power-aware compilation for embedded processors with dynamic voltage scaling and adaptive body biasing capabilities. In *Proceedings of the conference on Design, automation and test in Europe: Proceedings*, DATE '06, pages 943–944, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association.

[72] G. Semeraro, G. Magklis, R. Balasubramonian, D.H. Albonesi, S. Dwarkadas, and M.L. Scott. Energy-eficient processor design using multiple clocks domains with dynamic voltage and frequency scaling. In *In proc. 8th Intl. Symp. on High-Performance Computer Architecture*, pages 29–40, Fevereiro 2002.

[73] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power provisioning for a warehouse-sized computer. *SIGARCH Comput. Archit. News*, 35(2):13–23, June 2007.

[74] Dara Kusic, Jeffrey O. Kephart, James E. Hanson, Nagarajan Kandasamy, and Guofei Jiang. Power and performance management of virtualized computing environments via lookahead control. In *Proceedings of the 2008 International Conference on Autonomic Computing*, ICAC '08, pages 3–12, Washington, DC, USA, 2008. IEEE Computer Society.

[75] Peter Johnson and Tony Marker. Data centre energy efficiency. Technical report, Equipment Energy Efficiency Committee (E3), 2009.

[76] Official Ubuntu Documentation. Ubuntu server (cli) installation recommended minimum system requirements. https://help.ubuntu.com/community/Installation/SystemRequirements, Acessado em 2/12/2012.

[77] Microsoft TechNet Library. Windows server 2003, standard edition: System requirements. http://technet.microsoft.com/en-us/library/cc738496%28v=WS.10%29.aspx, Acessado em 2/12/2012.

[78] Hyung Won Choi, Hukeun Kwak, Andrew Sohn, and Kyusik Chung. Autonomous learning for efficient resource utilization of dynamic vm migration. In *Proceedings of the 22nd annual international conference on Supercomputing*, ICS '08, pages 185–194, New York, NY, USA, 2008. ACM.

[79] Green Grid: 7x24 Change International, Ashrae, Silicon Valley Leadership Group, Program, Energy S. Epa, Gbc, and 2010. Recommendations for measuring and reporting overall data center efficiency. Technical report, Uptime Institute, 2010.