Um Algoritmo de Escalonamento para Redução do Consumo de Energia em Computação em Nuvem

Pedro Paulo Vezzá Campos

Monografia apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Bacharel em Ciência da Computação

Programa: Bacharelado em Ciência da Computação

Orientador: Prof. Dr. Daniel Macêdo Batista

10 de setembro de 2013

Resumo

CAMPOS, P P. V. **Um Algoritmo de Escalonamento para Redução do Consumo de Energia em Computação em Nuvem**. 2013. 36 p. Monografia (Graduação) – Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2013.

Com o contínuo barateamento de insumos computacionais tais como poder de processamento, armazenamento e largura de banda de rede há uma tendência atual de migração de serviços para nuvens computacionais, capazes de processar grandes quantidades de dados (*Big data*) gerando resultados a um custo aceitável.

Um dos maiores custos envolvidos na operação de uma nuvem é o custo energético necessário para manter o parque de servidores operando e refrigerado. Para reduzir tais custos este trabalho de conclusão de curso apresenta um algoritmo de escalonamento de tarefas para computação em nuvem que reduz tal consumo energético.

Um simulador de computação em nuvem foi utilizado juntamente com cargas de trabalho realísticas para gerar experimentos que evidenciem o comportamento do algoritmo em diferentes condições de uso.

Palavras-chave: computação em nuvem, escalonamento, consumo energético, CloudSim.

Abstract

CAMPOS, P. P. V. **A Scheduling Algorithm for Energy Usage Reduction in Cloud Computing**. 2013. 36 p. Dissertation (Graduation) – Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2013.

With the continuous cheapening of computational resources such as processing power, storage and bandwidth there is a current trend of migration of services to cloud providers, capable of processing large amounts of data (Big data) generating results at a reasonable price.

One of the biggest costs involved in the operation of a cloud is the energetic cost necessary to keep the server pool operating and refrigerated. To reduce such costs this work presents a task scheduling algorithm for cloud computing environments that reduces such energy consumption.

A cloud computing simulator was used together with realistic workloads to generate experiments that highlight the algorithm's behavior in different usage conditions.

Keywords: cloud computing, task scheduling, energy consumption, CloudSim.

Sumário

Parte I

Parte Objetiva

Capítulo 0

Cronograma

0.1 Tarefas Realizadas

1. Leitura da bibliografia sobre:
   - Vantagens da virtualização de servidores em termos energéticos: [BH07] [BB10] [BGD+09] [FC07] [Mur08] [VMwa] [RSR+07]
   - Arquitetura interna dos simuladores utilizados: CloudSim e WorkflowSim [CRB+11] [CD12]
   - Escalonamento de tarefas em computação em grade (*grid computing*) [CBdF11] [BCdF11]
2. Experimentos iniciais com quatro simuladores de computação em nuvem/grade: CloudSim, GridSim, SimGrid e WorkflowSim
3. Simulação energética de experimentos-controle no PowerWorkflowSim, o simulador que será um dos resultados do TCC
4. Estudo do código fonte C++ produzido no artigo [CBdF11]
5. Redação de 19 seções da monografia de um total de 29

0.2 Tarefas em Andamento

1. Implementação do algoritmo HEFT
2. Desenvolvimento do PowerWorkflowSim, que une as funcionalidades da biblioteca de simulação energética do CloudSim com o processamento de workflows científicos do WorkflowSim

0.3 Tarefas a fazer

1. Implementar requisitos de software no WorkflowSim
2. Implementar o algoritmo de [CBdF11] no WorkflowSim
3. Trabalhar com Elaine Watanabe, aluna do mestrado em Ciência da Computação do IME, na concepção de um novo algoritmo de escalonamento

Capítulo 1

Introdução

Esta monografia desenvolvida durante o ano de 2013 para a disciplina MAC0499 – Trabalho de Formatura Supervisionado apresenta os trabalhos realizados no estudo e experimentação com técnicas de escalonamento de tarefas em ambientes de computação em nuvem sob a orientação do professor Daniel Macêdo Batista.

Em conjunto com a aluna de mestrado Elaine Watanabe, foi desenvolvido e avaliado um novo algoritmo que fosse energeticamente eficiente. O escalonador deve atender aos requisitos da aplicação e ao mesmo tempo buscar uma alocação de recursos próxima da ótima em termos de de economia de energia. Enquanto a aluna focou na concepção do algoritmo, o aluno dedicou-se a adaptar simuladores existentes para validar o algoritmo e realizar experimentos para estudar seu comportamento diante de diferentes cargas de trabalho.

A parte objetiva deste trabalho está organizado da seguinte forma: No Capítulo 2 são apresentados brevemente os conceitos que fundamentam pesquisas na área e que são necessários para a compreensão dos capítulos seguintes. Posteriormente, no Capítulo 3 o foco é direcionado para os trabalhos desenvolvidos especificamente para este TCC. O algoritmo desenvolvido é apresentado juntamente com resultados experimentais comparativos. Conclusões e considerações finais são descritas no Capítulo 4.

Já a parte subjetiva está organizada em dois tópicos principais: No Capítulo 5 há uma reflexão acerca do processo de produção deste trabalho e sua aplicação. Já no Capítulo 6 há uma reflexão mais ampla sobre as experiências vividas em cinco anos de graduação em duas universidades distintas e uma previsão dos próximos passos a seguir.

1.1 Motivação

Desde a década de 1970 a oferta de poder computacional, armazenamento e comunicação vem crescendo em um ritmo exponencial com o tempo. Até o fim da década de 1990 essas necessidades vinham sendo supridas com aperfeiçoamentos nas arquiteturas dos computadores e melhorias no processo produtivo. A Lei de Moore continuava se mostrando válida, duplicando o poder dos computadores e servidores a cada 18 meses e, junto com esse aumento, impondo uma necessidade energética cada vez maior para manter o computador funcionando e refrigerado. Porém, nos 2000 percebeu-se que o projeto de processadores encontrou uma barreira de potência. Processadores da época, tal como o Pentium 4, dissipavam 100W de potência e sua eficiência energética era baixa. [PH12] Assim, surgiu uma nova tendência, processadores mais simples e mais paralelos utilizando novas técnicas de economia de energia. Em suma, surgiu uma demanda por uma computação mais "verde" (*green computing*), que valoriza a sustentabilidade dos seus processos e a economia de recursos.

Iniciou-se, assim, uma tendência de concentração do poder de computação e armazenamento em torno dos grandes *data centers* e *data warehouses*. A computação em nuvem (*cloud computing*) passou a ser apresentada como uma solução para a redução de custos e desperdícios através da racionalização de recursos computacionais. Na Seção 2.1 são apresentadas as inovações presentes

nesse modelo de computação. Porém, o sucesso dessa metodologia depende de estratégias inteligentes que permitam gerenciar os recursos disponíveis permitindo realizar uma economia de escala sem afetar os usuários.

Em um nível mais técnico, uma nuvem é projetada para executar tarefas, estas subdivididas em subtarefas. Cada subtarefa pode possuir uma demanda específica de ambiente para ser executada, sistema operacional, programas instalados, poder mínimo de processamento, armazenamento, etc. Ainda, subtarefas podem depender de que uma subtarefa anterior tenha sido concluída antes de poder ser executada. Em [CBdF11] e [BCdF11] Chaves e Batista mostraram que é possível modelar tais tarefas como digrafos acíclicos (DAGs) que incorporem as demandas de ambiente. Ainda, desenvolveram uma heurística para escalonar as subtarefas em computação em grade, similar à computação em nuvem, visando diminuir o tempo de conclusão da tarefa através da redução do tráfego de rede.

1.2 Objetivos

Este trabalho tem por objetivo implementar e validar uma nova heurística para o problema apresentado que reduza o consumo energético sem grandes prejuízos ao tempo de execução da tarefa. A heurística foi desenvolvida por Elaine Watanabe, aluna de Mestrado em Computação do IME/USP. O desempenho desse algoritmo é comparado com a heurística proposta por Chaves e Batista em [CBdF11] e [BCdF11] e com outros algoritmos com objetivos similares encontrados na literatura. Para isso será feito uso de um simulador de computação em nuvem, o WorkflowSim, a ser detalhado na Seção 2.4.

1.3 Desafios

As dificuldades enfrentadas neste artigo vem de duas fontes: A primeira é tecnológica, há diversos simuladores de computação em nuvem ou em grade, porém o tópico de simulação energética é relativamente recente como atividade de pesquisa. Sendo assim, de todos programas de simulação estudados, CloudSim, GridSim, SimGrid e WorkflowSim, apenas o primeiro possui tal funcionalidade disponível para ser utilizada no momento. O simulador escolhido, WorkflowSim, apesar de ser baseado no CloudSim não tem por padrão a API de simulação energética disponível. Uma das tarefas da monografia foi justamente resolver tal problema.

O outro desafio é uma questão computacional mais fundamental: O problema de escalonar tarefas em diversos processadores (Num sentido mais amplo do que seria um processador) é NP-difícil [Sin07]. Para contornar esse problema diversas heurísticas já foram propostas, inclusive a apresentada nesta monografia. Um trecho do livro *Task Scheduling for Parallel Systems* de Oliver Sinnen resume a relação custo-benefício que deve ser ponderada para a descoberta de um escalonamento ótimo:

> Unfortunately, finding a schedule of minimal length (i.e., an optimal schedule) is in general a difficult problem. This becomes intuitively clear as one realizes that an optimal schedule is a trade-off between high parallelism and low interprocessor communication. On the one hand, nodes should be distributed among the processors in order to balance the workload. On the other hand, the more the nodes are distributed, the more interprocessor communications, which are expensive, are performed. In fact, the general decision problem (a decision problem is one whose answer is either “yes” or “no”) associated with the scheduling problem is NP-complete. [Sin07]

Capítulo 2

Conceitos

2.1 Computação em Nuvem

Computação em nuvem é um nicho de mercado que se tornou atrativo com o barateamento de insumos necessários à computação, como energia, poder de processamento, armazenamento e transmissão de dados, permitindo uma economia de escala. [AFG+09] Através desse paradigma computacional, surgiu a noção de computação como um serviço, elástico, virtualmente ilimitado e pago apenas pela porção realmente utilizada, muito similar ao sistema de distribuição elétrica.

O objetivo final da computação em nuvem é concentrar dados para prover um serviço ubíquo ao usuário, empresas ou pessoas físicas, que delegariam a gestão dessa informação a terceiros competentes para prover um serviço de qualidade e seguro. Grandes empresas da área de tecnologia possuem soluções de computação em nuvem, dentre as quais podemos citar Amazon [1], Google [2], Microsoft [3] e IBM [4].

Uma importante vantagem de *cloud computing* é que com essa concentração de dados e serviços é possível desenvolver técnicas de otimização do uso de grandes *data centers*. Segundo estudo realizado por Barroso e Hölzle [BH07] em 5000 servidores do Google, raramente eles permanecem completamente ociosos e dificilmente operam próximos da sua utilização máxima. Na maior parte do tempo estão trabalhando entre 10% e 50% do nível máximo. Os autores mostram que justamente nessa faixa de utilização tais servidores são menos eficientes energeticamente.

Computação em nuvem é uma candidata a ajudar a melhorar essa perspectiva. Através de virtualização e reposicionamento automático de máquinas virtuais no data center de maneira transparente, uma funcionalidade disponível em produtos pagos como o VMware vSphere [VMwb] e softwares livres como o Xen [AU09], é possível dimensionar qual parcela do data center estará ativa em um dado momento dependendo da demanda. Servidores com pouca utilização podem ser virtualizados em um único servidor físico de modo que este trabalhe com uma utilização que seja mais eficiente.

Vale ressaltar que em uma situação ideal, toda essa consolidação de servidores é transparente ao usuário final. Em caso de um pico na demanda por um determinado serviço, o provedor da nuvem deve garantir que haja uma resposta rápida da infraestrutura para suportar a nova carga requisitada. Dessa forma, são respeitados os acordos de nível de serviço (SLA - *Service level agreement*) estabelecidos entre o usuário e o fornecedor da nuvem.

2.2 Consumo Energético

Seguindo a tendência comentada na Seção 1.1, os serviços de TI vem apresentando um forte crescimento devido à contínua migração de serviços, análise de dados (*Big Data*, *Business Intelli-*

[1] Amazon Elastic Compute Cloud (Amazon EC2): http://aws.amazon.com/pt/ec2/
[2] Google Cloud Platform: https://cloud.google.com/
[3] Windows Azure: http://www.windowsazure.com/pt-br/
[4] IBM SmartCloud: http://www.ibm.com/cloud-computing/us/en/

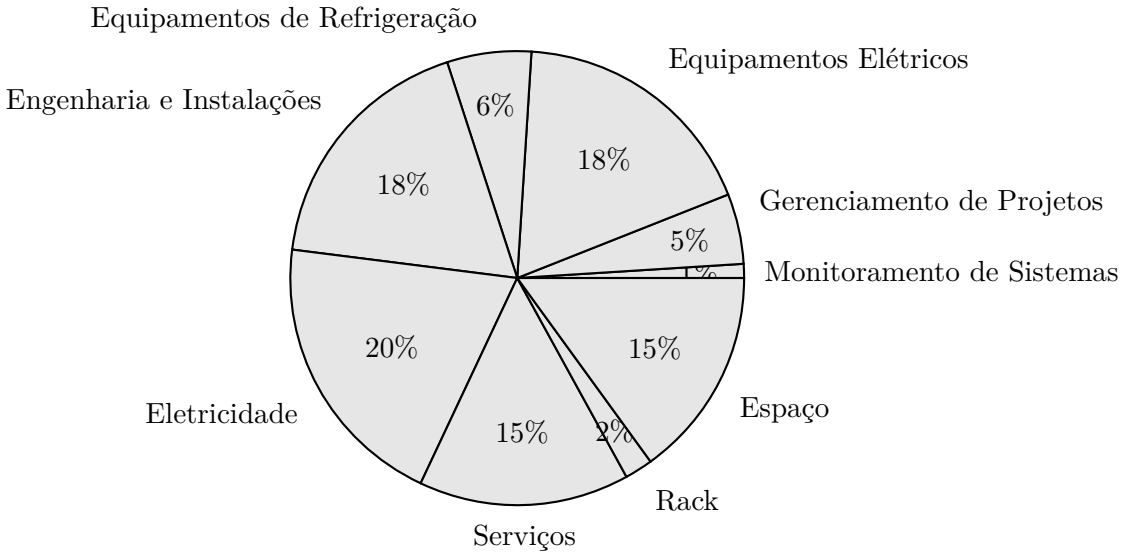

Figura 2.1: *Custo total de posse de um* rack *em um* data center *típico de alta disponibilidade [Ras11]*

gence, etc.) e processamentos científicos para grandes data centers. Com isso, o consumo energético total também tem aumentado. Em uma análise do custo total de posse de um *data center* de alta disponibilidade, cada *rack* [5] possui um custo de US$120.000 ao longo de 10 anos [Ras11]. Este custo está dividido conforme apresentado na Figura 2.1.

Como é possível ver na Figura 2.1, os custos relacionados à eletricidade mais os gastos com equipamentos que apenas cumprem o propósito de garantir que o servidor permaneça ligado e refrigerado totalizam 44%. Portanto, uma redução nestes gastos gera um impacto tanto econômico quanto ambiental, com a redução dos recursos naturais necessários para sustentar um *data center*.

2.2.1 Migração de Máquinas Virtuais

Uma das grandes vantagens de manipular máquinas virtuais em um ambiente como o de computação em nuvem é o fato que é relativamente simples fazer uma realocação de máquinas virtuais de uma máquina hospedeira para outra, mesmo enquanto a máquina virtual está funcionando [VMwb] [AU09].

O objetivo desse processo é o de equalizar a infraestrutura utilizada efetivamente com a demanda atual. Em momentos de poucas requisições é possível minimizar o número de nós físicos que estão atendendo a carga de trabalho atual. Enquanto isso, os nós ociosos ficam livres para serem desligados ou colocados em algum estado de hibernação profunda, com baixo consumo energético [BB10]. Quando há um aumento na demanda, é possível realizar o caminho inverso, distribuindo as máquinas virtuais em mais hospedeiros, garantindo o cumprimento do SLA definido entre o provedor da nuvem e o usuário.

2.2.2 Dimensionamento Dinâmico de Tensão e Frequência

A estratégia de dimensionamento dinâmico de tensão e frequência, do inglês *dynamic voltage and frequency scaling*, é uma tática bastante difundida entre os fabricantes de processadores como uma forma pouco invasiva de economizar energia elétrica. Tecnologias como o *Intel Speed Step* e o *AMD Coll'n'Quiet*, que ajustam automaticamente, em hardware, a tensão e frequência dos processadores de acordo com suas cargas de trabalho. Se há altas cargas de trabalho, o processador aumenta os níveis de tensão e frequência, caso contrário diminui esses níveis. [LMB12]

[5] O autor define um *rack* como um gabinete vazado de tamanho padronizado (O mais comum é a versão de 19 polegadas de largura e 42 unidades de altura. Cada unidade equivale a 1,75 polegada) e também gabinetes que contém *mainframes* e unidades de *storage*.

2.3 Escalonamento de Tarefas

Segundo [LMB12], o escalonamento de tarefas pode ser visto através de duas óticas: A do usuário da nuvem, que deseja que sua tarefa execute o mais rapidamente possivel e com o menor custo e a perspectiva do provedor da nuvem, interessado em reduzir os recursos utilizados, gerando economias na manutenção e energia elétrica. Esta monografia foca no escalonamento pelo ponto de vista do provedor.

2.3.1 *Heterogeneous Earliest Finish Time*

O algoritmo *Heterogeneous Earliest Finish Time* – HEFT é um algoritmo de escalonamento de fluxos de trabalho modelados utilizando um digrafo acíclico (Directed acyclic graph – DAG) para um número limitado de computadores heterogêneos [KH01]. Como o problema de escalonamento de tarefas é NP-difícil, ele é frequentemente utilizado como referência na literatura para comparar o desempenho de novas propostas, como por exemplo em [BCdF11].

O HEFT recebe como entrada um conjunto de tarefas modeladas como um DAG, um conjunto de nós computacionais, os tempos para executar uma tarefa em um dado nó e os tempos necessários para comunicar os resultados de uma tarefa para cada uma de suas tarefas filhas no DAG. Como saída o algoritmo gera um escalonamento, mapeando cada tarefa a uma máquina.

O HEFT consiste de duas fases principais:

Fase de priorização Definição das prioridades das tarefas e a seleção de tais tarefas com base nas suas prioridades

Fase de seleção Mapeamento e escalonamento de cada tarefa selecionada em um processador

Fase de Priorização

Nesta fase do algoritmo HEFT, cada tarefa deve ser priorizada considerando o comprimento do caminho crítico (Ou seja, o maior caminho) de uma dada tarefa até a tarefa final no fluxo de trabalho. (Passos 1 a 5 no algoritmo HETEROGENEOUS-EARLIEST-FINISH-TIME). A lista de tarefas a serem executadas é então ordenada pela ordem decrescente do comprimento do caminho crítico. Com essa ordem, produzimos uma ordenação topológica das tarefas, preservando as restrições de precedência do DAG.

A prioridade de uma tarefa n_i é definida recursivamente como:

$$rank_u(n_i) = \overline{w_i} + \max_{n_j \in succ(n_i)} (\overline{c_{i,j}} + rank_u(n_j))$$

onde n_i representa a i-ésima tarefa, $\overline{w_i}$ é uma média do custo computacional da tarefa i entre todos os processadores, $succ(n_i)$ é o conjunto de todos as tarefas que dependem imediatamente da tarefa n_i e $\overline{c_{i,j}}$ é o custo de comunicação dos dados transferidos entre as tarefas n_i e n_j.

Note que o cálculo de $rank_u(n_i)$ depende do cálculo do $rank$ de todas as suas tarefas filhas. A noção intuitiva por trás do $rank$ é que ele deve representar a distância esperada de qualquer tarefa até o fim da execução do *workflow*.

HETEROGENEOUS-EARLIEST-FINISH-TIME()

1 Defina os custos computacionais das tarefas e os custos de comunicação das arestas com valores médios
2 Calcule $rank_u$ para todas as tarefas varrendo o grafo de "baixo para cima", iniciando pela tarefa final.
3 Ordene as tarefas em uma lista de escalonamento utilizando uma ordem não crescente de valores de $rank_u$.
4 **enquanto** há tarefas não escalonadas na lista
5 Selecione a primeira tarefa, n_i da lista de escalonamento.
6 **para** cada processafor p_k no conjunto de processadores ($p_k \in P$)
7 Calcule o tempo mais cedo de conclusão da tarefa n_i
8 Defina a tarefa n_i para executar no processador p_j que minimiza o tempo mais cedo de conclusão da tarefa n_i.

Fase de Seleção

Para selecionar qual processador irá executar uma dada tarefa, calculamos o tempo mais cedo de conclusão (*earliest finish time* – EFT), de uma dada tarefa. Normalmente, o tempo mais cedo para um processador p_j tornar-se disponível para executar uma tarefa é o momento que p_j termina a execução da última tarefa designada a ele. (Caso não haja tarefa sendo executada, o valor será zero.) Com o algoritmo HEFT, a busca por um espaço de tempo vago em um processador p_j começa a partir do momento que o processador p_j torna-se vago. A busca continua até que seja possível encontrar o primeiro *slot* de tempo grande o suficiente para suportar a computação da tarefa n_i.

Análise de Complexidade do HEFT

No algoritmo HETEROGENEOUS-EARLIEST-FINISH-TIME, o passo 1 toma tempo $O(e \times p)$ para computar as médias enquanto o passo 2 toma tempo $O(e)$ para computar o comprimento do caminho crítico, onde e é o número de arestas no DAG e p o número de processadores. Para n tarefas a serem escalonadas, o passo 3 necessita de um tempo $O(n \log n)$ para ordenar as tarefas pelo comprimento de seus caminhos críticos. Seja a o número de tarefas que tem n_i como predecessora no DAG, então os passos 5-9 ocupam um tempo $O(a \times p)$ para uma tarefa n_i, assim, o laço enquanto necessita de um tempo $O(e \times p)$. Portanto, a complexidade do algoritmo HEFT é $O(e \times p)$.

Exemplo de Execução

2.3.2 Embutindo Requisitos de Software no Escalonamento

Notamos neste momento que tarefas a serem executadas em um ambiente de computação em nuvem ou em grade pode requisitar algum software específicos para sua execução. Em [BCdF11] é apresentada uma técnica a ser descrita nesta seção para modificar o DAG de dependências entre as tarefas para incorporar tais requisitos de software.

Trivialmente há duas possibilidades para resolver o problema: A primeira possibilidade é alocar apenas uma máquina virtual para cada software distinto a ser executado no fluxo de trabalho. Esta ideia tem o problema de gerar falsas dependências entre as tarefas, já que apenas uma tarefa pode executar por vez em uma máquina e, assim, coisas que poderiam executar em paralelo passam a ter que ser processadas sequencialmente. Outra alternativa é alocar uma máquina virtual para cada tarefa. Esta abordagem além de ser custosa em recursos alocados não necessariamente garante o menor tempo de execução possível pois o tráfego de rede entre os nós de processamento pode não ser o ótimo.

Há uma quantidade exponencial de diferentes formas de alocar máquinas para executar um dado fluxo de trabalho. Esse número vem do fato que alocar tarefas em máquinas é equivalente a encontrar o número de partições de um conjunto, modelado pelo número de Bell. Assim, os autores apresentam uma heurística para o problema.

$$B_0 = 1$$
$$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k$$

O objetivo da heurística proposta é reduzir o tráfego de rede necessário e, ao mesmo tempo, evitar aumentar o caminho crítico dos fluxos de trabalho. Isto é feito através da busca das tarefas que possuem os mesmos requisitos de software em um dado caminho do DAG. Dessa forma, há a instanciação de apenas uma VM para cada dependência de software em um caminho.

2.4 Simuladores de computação em nuvem e fluxos de trabalho

Para diferentes algoritmos e abordagens para problemas em computação em nuvem há algumas alternativas: Uma possibilidade seria a execução de instâncias reais dos problemas em provedores reais de *cloud computing*. Essa alternativa possui como problema o custo monetário envolvido na instanciação de máquinas virtuais por um tempo prolongado. Ainda, não há o controle sobre o ambiente de simulação, prejudicando a reprodutibilidade dos experimentos. Outra opção seria a de instalar um ambiente próprio para experimentos, novamente esbarrando em problemas financeiros.

Uma alternativa mais factível é a utilização de ambientes de simulação computacional. Estas ferramentas abrem a possibilidade de avaliar uma hipótese em um ambiente totalmente controlado e facilmente reproduzível. Ainda, há um ganho na facilidade de simular diferentes variações de ambientes, facilitando a busca por gargalos na eficiência dos algoritmos utilizados. Esses benefícios vem ao custo da simplificação dos modelos utilizados para simular o ambiente proposto.

Nesta seção serão apresentados os principais simuladores utilizados nos experimentos descritos no Capítulo 3: CloudSim, WorkflowSim e PowerWorkflowSim.

2.4.1 CloudSim

CloudSim [CRB+11] é um simulador para computação em nuvem, reconhecido academicamente com citações em mais de 300 de trabalhos indexados pelo Google Scholar [Goo13]. Ele provê as funcionalidades de simulação de máquinas virtuais, *hosts*, *data centers*, políticas de provisionamento de recursos, tarefas a serem executadas em máquinas virtuais (*cloudlets*) além de efetuar análises de duração de simulações e consumo de energia. Na Figura 2.2 é possível ver a arquitetura projetada pelos desenvolvedores do CloudSim.

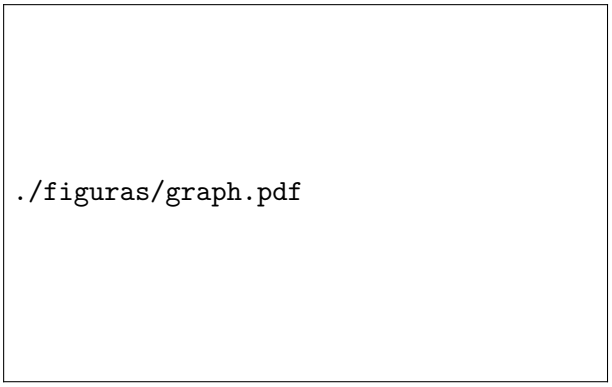

Figura 2.2: *Arquitetura do CloudSim*

A arquitetura em camadas do CloudSim permite uma separação clara das diferentes possibilidades de extensão e experimentação em um ambiente de computação em nuvem. Por exemplo, todo o código responsável pela simulação energética do CloudSim é na verdade uma especialização das

classes originais do simulador. Assim, a classe `PowerDatacenter` e `PowerHost` são classes herdadas de `Datacenter` e `Host` respectivamente.

Com a boa aceitação do CloudSim como uma ferramenta de simulação, passaram a surgir simuladores mais especializados, como por exemplo o WorkflowSim, assunto da Seção 2.4.2.

Modelagem de uma Nuvem Computacional

Os serviços de infraestrutura providos por nuvens podem ser simulados através da extensão da classe `Datacenter` do CloudSim. Esta entidade gerencia diversos `Hosts`. Cada `Host` possui uma capacidade de processamento pré determinada, medida em milhões de instruções por segundo (MIPS) e podendo ser com apenas um único ou vários núcleos de processamento, memória e armazenamento. Um `Host` pode incorporar uma ou mais `Vm` (Máquina virtual) de acordo com as políticas de alocação definidas nele pelo administrador da nuvem. Uma política de alocação é definida como as operações relacionadas a uma `Vm` durante seu ciclo de vida: Escolha de um `Host`, criação, migração e destruição.

De maneira similar, cada `Vm` possui capacidades de processamento, memória, largura de banda e número de processadores definidos no momento da sua criação. Ainda, pode executar uma ou mais tarefas, `Cloudlets`, obedecendo-se as políticas de provisionamento de aplicações.

Modelagem do Consumo Energético

Como discutido anteriormente, o CloudSim possui uma extensão que incorpora a modelagem do consumo energético das máquinas utilizadas na simulação. Nessa extensão cada elemento de processamento (Tipicamente um núcleo) inclui um objeto que estende o tipo abstrato `PowerModel`, responsável por gerenciar o consumo energético. Isso garante um desacoplamento entre o processamento e a estratégia de modelagem energética empregadas. Por exemplo, um PowerModel pode levar em conta a estratégia de DVFS descrita na Seção 2.2.2 enquanto outro não. O CloudSim apresenta algumas implementações concretas da classe `PowerModel`. A técnica de modelagem empregada é descrita em [BB12].

2.4.2 WorkflowSim

Apesar de ser bem consolidado como ferramenta de simulação de computação em nuvem, o CloudSim não possui certas características necessárias à simulação de *workflows* científicos como por exemplo as ineficiências causadas pelo uso de sistemas heterogêneos e falhas. Ainda, percebe-se a falta de suporte a técnicas de otimização de execução de *workflows* amplamente utilizadas tais como o *clustering* de tarefas. De forma a resolver essas demandas o WorkflowSim foi criado tendo como base o CloudSim [CD12].

Enquanto o CloudSim se concentra na execução de uma única carga de trabalho, o WorkflowSim foca no escalonamento do *workflow* e sua execução. O último processa *workflows* modelados como DAGs, realizando um escalonamento que obedece a precedência imposta pelo DAG. Ainda, é simples implementar diferentes algoritmos de escalonamento para avaliar suas eficiências. Uma das atividades desse TCC foi implementar o algoritmo HEFT comentado na Seção 2.3.1 no WorkflowSim. [6]

Na parte de modelagem de falhas, os autores decidiram focar em falhas transientes, mais frequentes que falhas permanentes. O simulador permite ao projetista de algoritmos de escalonamento desenvolver algoritmos tolerantes a tais falhas transientes.

Como é possível ver na Figura 2.3, há múltiplas camadas de componentes envolvidos na preparação e execução do *workflow*: Um `Workflow Mapper` que mapeia *workflows* abstratos em *workflows* concretos, dependentes do local de execução; uma `Workflow Engine` que gerencia a dependência de dados e um `Workflow Scheduler` para associar tarefas aos processadores. Outros componentes

[6] Outros algoritmos de escalonamento tais como FCFS, MinMin, MaxMin, MCT e *Data aware* já estão implementados na versão atual (1.0)

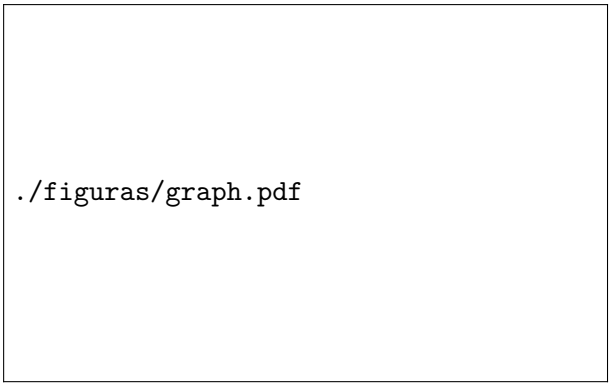

Figura 2.3: *Arquitetura do WorkflowSim*

incluem uma `Clustering Engine`, que é responsável tarefas menores em um pacote maior, um `Provenance Collector`, dedicado a registrar a história da execução de uma tarefa e um `Workflow Partitioner`, que divide o *workflow* em diversos *sub-workflows*.

Capítulo 3

Experimentos

Este capítulo é dividido em quatro momentos distintos dos experimentos realizados para esta monografia. Primeiro, na Seção 3.1 é apresentado o simulador desenvolvido para esta monografia, o PowerWorkflowSim; Depois, na Seção 3.2 são apresentadas as configurações do ambiente experimental; Resultados dos experimentos de controle são exibidos na Seção 3.3. Em seguida, na Seção 3.4 é feito um estudo detalhado sobre a heurística encontrada para minimizar o consumo energético e o algoritmo correspondente implementado; Por fim na Seção 3.5 são apresentados os resultados experimentais que mostram o ganho energético com o uso do algoritmo proposto.

3.1 PowerWorkflowSim

Durante a implementação do WorkflowSim os desenvolvedores não tomaram o cuidado de manter a API de simulação energética do CloudSim acessível ao usuário final. De maneira a resolver esse problema, o autor da monografia desenvolveu uma versão alternativa do WorkflowSim, o PowerWorkflowSim.

Como tanto a API energética do CloudSim quanto o WorkflowSim são basicamente *wrappers* do CloudSim, o trabalho necessário foi o de garantir que a funcionalidade de um não conflitasse com a do outro. O diagrama de classes resultante do PowerWorkflowSim está representado abaixo.

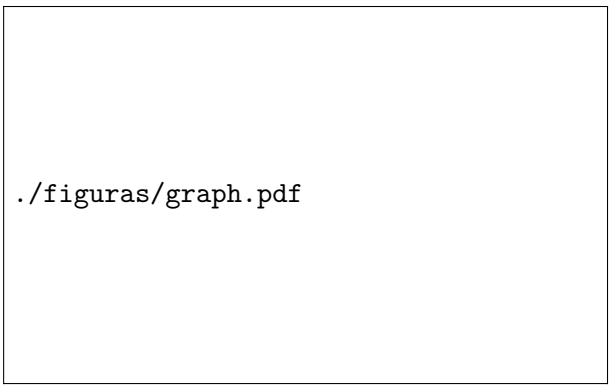

Figura 3.1: *Diagrama de Classes do PowerWorkflowSim*

3.2 Ambiente Simulado

Esta seção apresenta os valores adotados para os diversos parâmetros disponíveis a uma simulação do PowerWorkflowSim. As configurações estão divididas por entidades: Na Tabela 3.1 são apresentadas as configurações de uma máquina virtual. Já nas Tabelas 3.2 e 3.3 estão presentes as configurações das máquinas físicas simuladas, duas versões de servidores HP ProLiant ML110, um da geração 3 e outro da geração 5. Ainda, na Tabela 3.4 são apresentadas as configurações que

Parâmetro	Valor
Número de máquinas	20
Número de *Cores*	1
Velocidade de um Core	1000 MIPS
Tamanho da Imagem	10 GB
RAM	512 MB
Largura de Banda	1 Gbps
Gerenciamento de tarefas	Compartilhadas no espaço[1]
Gerenciador de VMs	Xen

Tabela 3.1: *Configurações das máquinas virtuais*

Parâmetro	Valor
Número de máquinas	10
Número de *Cores*	2
Velocidade de um Core	1860 MIPS (Xeon 3040)
RAM	4 GiB
Largura de Banda	1 Gbps
Armazenamento	1 GB
Modelo de consumo energético	[Sta08a]

Tabela 3.2: *Configuração de um host HP ProLiant ML110 G3*

abrangem o Datacenter simulado. Por fim, na Tabela 3.5 são vistas as configurações do simulador como um todo.

3.3 Experimentos de controle

Para facilitar a análise de algoritmos que operam sobre fluxos de trabalho, tal como os algoritmos descritos nesta monografia, o projeto Pegasus Workflow Management System desenvolveu uma ferramenta geradora de *workflows* científicos. Estes *workflows* artificiais são gerados a partir de informações extraídas de instâncias reais das aplicações juntamente com conhecimentos prévios sobre seu funcionamento interno [Peg12].

Os fluxos gerados pelo projeto Pegasus foram então executados no PowerWorkflowSim com as configurações apresentadas na Seção 3.2. Os resultados são apresentados na Tabela 3.6.

[1] Tarefas compartilhadas no espaço possuem fatias do recurso desejado reservadas a cada uma delas. Outra alternativa é o compartilhamento no tempo no qual cada tarefa faz uso de dos recursos por uma fatia de tempo determinada em esquema *round-robin*, por exemplo.

Parâmetro	Valor
Número de máquinas	10
Número de *Cores*	2
Velocidade de um Core	2660 MIPS (Xeon 3075)
RAM	4 GiB
Largura de Banda	1 Gbps
Armazenamento	1 GB
Modelo de consumo energético	[Sta08b]

Tabela 3.3: *Configuração de um* host *HP ProLiant ML110 G5*

Parâmetro	Valor
Arquitetura	x86
Sistema Operacional	Linux
Gerenciador de VMs	Xen
Fuso Horário	-3
Custo de Processamento	3 unidades
Custo da RAM	0.05 unidade
Custo do Armazenamento	0.1 unidade
Custo da Banda	0.1 unidade
Largura de	0.1 unidade

Tabela 3.4: *Configurações do* datacenter

Parâmetro	Valor
Escalonador	*First Come First Served*
Estratégia de Economia de Energia	DVFS (Vide Seção 2.2.2)

Tabela 3.5: *Configurações do PowerWorkflowSim*

Experimento	Número de tarefas	Tempo simulado (s)	Consumo energético (kWh)
CyberShake	30	246.65	0.13
CyberShake	50	267.20	0.15
CyberShake	100	306.94	0.17
CyberShake	1000	1265.71	0.69
Epigenomics	24	5596.59	3.04
Epigenomics	46	7743.25	4.20
Epigenomics	100	34975.01	18.99
Epigenomics	997	207426.20	112.62
Inspiral	30	1335.62	0.73
Inspiral	50	1411.25	0.77
Inspiral	100	1519.47	0.82
Inspiral	1000	11712.90	6.36
Montage	25	46.99	0.03
Montage	50	66.76	0.04
Montage	100	102.91	0.06
Montage	1000	907.13	0.49
Sipht	30	4412.76	2.40
Sipht	60	4642.49	2.52
Sipht	100	4479.34	2.43
Sipht	1000	9646.45	5.24

Tabela 3.6: *Resultados dos experimentos de controle*

3.4 Algoritmo Proposto

3.5 Resultados Experimentais

Capítulo 4

Conclusões

4.1 Considerações Finais

Parte II

Parte Subjetiva

Capítulo 5

O Trabalho de Conclusão de Curso

5.1 Desafios e frustrações

5.2 Observações sobre a aplicação real de conceitos estudados

Capítulo 6

A Graduação em Ciência da Computação

Realizo neste capítulo um balanço sobre os cinco anos de graduação em Ciência da Computação realizados em duas universidades bastante distintas: A Universidade Federal de Santa Catarina, por dois anos, e a Universidade de São Paulo, por três anos. A primeira graduação, por estar localizada juntamente com os cursos de engenharia da UFSC, tem foco bastante voltado à parte tecnológica da Computação, com grupos fortes fazendo pesquisa em Banco de Dados, Sistemas Operacionais e Engenharia de Software. Já a segunda, localizada dentro do Instituto de Matemática e Estatística, foca muito mais nos estudos de Matemática e conteúdos de fundamentos de Computação como Algoritmos, Grafos e Autômatos.

Felizmente, essas as diferenças se tornam complementares quando há interesse do aluno de tentar absorver o que cada lugar tem de melhor. Me considero um sortudo por ter a chance de abraçar essa oportunidade e espero ter aproveitado ao máximo o que me foi oferecido. De fato, encerro esse ciclo com a sensação de dever cumprido, seja em termos acadêmicos, concluindo a graduação com boas notas e cumprindo com meus deveres estudantis, seja em termos sociais, cultivando boas amizades em cada lugar e aproveitando o que a Universidade oferece a seus alunos: restaurante universitário, biblioteca, cursos de idiomas, esportes, viagens acadêmicas, festas universitárias, iniciação científica etc.

6.1 Disciplinas cursadas relevantes para o desenvolvimento do TCC

Desenvolver este TCC foi uma amostra bastante relevante de como os conteúdos em Ciência da Computação estão relacionados, em maior ou menor grau. Felizmente, a área de Redes de Computadores é bastante eclética nas fundações que utiliza para gerar seus resultados. (E que resultados! A Internet é fruto dos esforços de inúmeros pesquisadores em Redes e sempre gerou fascínio em mim. Compreender seu funcionamento e a sua capacidade de mudança foram alguns dos motivos que me fizeram escolher Computação como a carreira que quero perseguir profissionalmente.)

Nesta seção é apresentada uma lista de cursos que fizeram a diferença para o desenvolvimento da minha monografia, direta ou indiretamente, em ordem cronológica.

6.1.1 Programação Orientada a Objetos II

Universidade UFSC

Professor Luiz Fernando Bier Melgarejo

POO II é uma disciplina obrigatória do segundo semestre de Computação na UFSC. O propósito da matéria é cobrir os principais conceitos de Orientação a Objetos e pô-los em prática em um

projeto. O destaque dessa disciplina não é tanto o conteúdo mas sim a metodologia de ensino do professor.

Após uma rápida introdução ao *framework* a ser usado durante a disciplina, os alunos começam desenvolvendo mini-projetos de interesse próprio e usando os (Poucos) conceitos que conhecem de POO aprendidos em POO I. Durante o semestre é obrigação do aluno se oferecer para apresentar os códigos que vem produzindo no(s) projeto(s). É nesse momento que entra o professor, criticando ferozmente o trabalho do aluno, comentando brevemente sobre práticas de programação e padrões de projeto que o aluno não conhecia mas que poderiam ser utilizados. É importante notar que o professor nunca dava a solução do problema. Cabia aos alunos mudar seus códigos por conta própria, pesquisar boas soluções e discutir com colegas para chegar a um consenso do que deve ser apresentado como "solução" em uma aula posterior.

O curioso é que os alunos que sobreviviam a tal provação eram na grande maioria aprovados. Mas mais que apenas uma nota no histórico escolar, os alunos saiam muito mais unidos que antes graças à necessidade de companheirismo para enfrentar o "temido" professor. Além disso, absorviam muito mais conceitos do que numa aula expositiva simples, afinal, ninguém queria ser criticado na frente dos seus colegas então todos pensavam muito em seus códigos antes de fazer a apresentação.

Para este TCC, foi necessário analisar bastante código dos simuladores utilizados além de empregar técnicas vistas nessa disciplina para o desenvolvimento do PowerWorfkflowSim. Foi, também, nessa disciplina que criei a maior intimidade com programação Java, muito utilizada neste trabalho.

6.1.2 Organização de Computadores I

Universidade UFSC

Professor Luiz Cláudio Villar dos Santos

Organização de Computadores é uma disciplina obrigatória do terceiro semestre de Computação na UFSC. O propósito da disciplina é apresentar a interface hardware-software. Novamente, o destaque novamente vai para o professor. Por um lado, ele é respeitado pelo seu profundo conhecimento de Computação em geral e vivência profissional enquanto que por outro é temido pelo seu alto índice de reprovação.

O professor faz questão de ministrar seu curso tal como fazem as melhores universidades do país. O que pode ser problemático em termos de notas, novamente garante que os alunos que saiam da disciplina estejam preparados para enfrentar problemas com muito mais confiança que em outros lugares.

Para o trabalho de conclusão, foram fundamentais a conceituação de consumo energético e os *tradeoffs* envolvidos na questão velocidade $\times$ economia energética.

6.1.3 Algoritmos em Grafos

Universidade USP

Professor Arnaldo Mandel

Algoritmos em Grafos é uma disciplina obrigatória do quinto semestre de Computação do IME-USP. Aqui, são estudadas as formas de representação computacional de grafos, um pouco de modelagem de problemas usando grafos e diversos algoritmos fundamentais da área.

Grafos e seus algoritmos são uma verdadeira cornucópia de utilidades em Ciência da Computação. Uma estrutura simples e elegante é capaz de modelar e resolver inúmeros problemas e de uma maneira normalmente intuitiva ou palpável. Para um trabalho envolvendo Redes isso não pode ser diferente. Processei DAGs de fluxos de trabalho científicos, apliquei uma busca em profundidade para gerar uma ordenação topológica com o objetivo executar códigos em diversos nós interligados. Claramente, Grafos foram fundamentais para esse trabalho.

6.1.4 Programação para Redes de Computadores

Universidade USP

Professor Daniel Macêdo Batista

Programação para Redes de Computadores é uma disciplina optativa do curso de Computação do IME-USP. Aqui é feita uma abordagem *top-down* do modelo TCP-IP (Internet) de Redes de Computadores. São estudados os conceitos fundamentais de cada camada, protocolos relevantes e realizados experimentos para reforçar os conceitos aprendidos.

Apesar de já ter cursado Redes de Computadores na UFSC, fazendo um estudo *bottom-up* e estudando redes OSI juntamente com TCP-IP, sentia que meus conhecimentos na área estavam ainda fracos e cursar Programação para Redes de Computadores foi uma ótima decisão. Por ser uma disciplina conjunta com a pós graduação há alunos muito capacitados e interessados no estudo da disciplina. Ainda, o professor consegue em seu curso um feito muito difícil disciplinas de graduação: Motivar os alunos a discutir os assuntos em aula, enriquecendo-a. Por fim, o professor mostra que tem conhecimento prático de programação em redes ao programar e simular os conceitos vistos diretamente na aula.

Para o TCC o professor Daniel aceitou a tarefa de ser meu orientador, muito profissional, trouxe sempre materiais que tornassem o meu trabalho melhor e contribuiu com comentários muito pertinentes.

6.2 Próximos Passos

Com o fim da graduação iminente, surgem as dúvidas de qual carreira seguir. Como comentado no começo do capítulo, considero que aproveitei a universidade nas várias formas possíveis. Na área de pesquisa, participei de duas iniciações científicas diferentes, uma na área de hardware na UFSC e outra na área de ensino de Computação na USP. Assim, considero que este é um bom momento para gerar uma mudança na minha vida profissional, ingressando na indústria de software.

Pretendo continuar trabalhando com Redes e Computação, mas agora em um contexto mais prático. Uma das coisas que mais sinto falta na carreira acadêmica é ver alguma ideia desenvolvida ser utilizada rotineiramente por milhares (Milhões? Bilhões?) de pessoas. Ou seja, ir além do *paper*, da dissertação e da tese. De qualquer forma, mantenho meu carinho especial pelos anos vividos na universidade, de onde levo conhecimento, contatos e amigos para o resto da vida.

Referências Bibliográficas

[AFG+09] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica e Matei Zaharia. Above the clouds: A berkeley view of cloud computing, Feb 2009. 7

[AU09] Gustavo P Alkmim e Joaquim Quinteiro Uchôa. Uma solução de baixo custo para a Migração de Máquinas Virtuais. *WPwerformance - VIII Workshop em Desenvolvimento de Sistemas Computacionais e de Comunicação*, páginas 2161–2175, 2009. 7, 8

[BB10] Anton Beloglazov e Rajkumar Buyya. Energy Efficient Allocation of Virtual Machines in Cloud Data Centers. *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, páginas 577–578, 2010. 3, 8

[BB12] Anton Beloglazov e Rajkumar Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers. *Concurrency and Computation: Practice and Experience*, 24:1397–1420, 2012. 12

[BCdF11] D.M. Batista, C.G. Chaves e N.L.S. da Fonseca. Embedding software requirements in grid scheduling. Em *Communications (ICC), 2011 IEEE International Conference on*, páginas 1–6, 2011. 3, 6, 9, 10

[BGD+09] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Q. Dang e K. Pentikousis. Energy-Efficient Cloud Computing. *The Computer Journal*, 53(7):1045–1051, Agosto 2009. 3

[BH07] L.A. Barroso e U. Hölzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, 2007. 3, 7

[CBdF11] C.G. Chaves, D.M. Batista e N.L.S. da Fonseca. Scheduling grid applications with software requirements. *Latin America Transactions, IEEE (Revista IEEE America Latina)*, 9(4):578–585, 2011. 3, 6

[CD12] Weiwei Chen e E. Deelman. Workflowsim: A toolkit for simulating scientific workflows in distributed environments. Em *E-Science (e-Science), 2012 IEEE 8th International Conference on*, páginas 1–8, 2012. 3, 12

[CRB+11] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose e Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exper.*, 41(1):23–50, Janeiro 2011. 3, 11

[FC07] Wu-chun Feng Wu-chun Feng e K W Cameron. The Green500 List: Encouraging Sustainable Supercomputing. *Computer*, 40(12):50–55, 2007. 3

[Goo13] Google. Calheiros: Cloudsim: a toolkit for modeling and simulation of ... - google scholar. http://scholar.google.com.br/scholar?cites=2720541035065929999&as_sdt=2005&sciodt=0,5&hl=pt-BR, 2013. [Online; acessado em 30 de agosto de 2013]. 11

[KH01] D. Kim e S. Hariri. *Virtual Computing: Concept, Design, and Evaluation.* Advances in Information Security. Kluwer Academic Publishers, 2001. 9

[LMB12] Daniel G Lago, Edmundo R M Madeira e Luiz Fernando Bittencourt. Escalonamento com Prioridade na Alocação Ciente de Energia de Máquinas Virtuais em Nuvens. *XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, páginas 508–521, 2012. 8, 9

[Mur08] San Murugesan. Harnessing Green IT : Principles and Practices. *IT Pro*, (February), 2008. 3

[Peg12] Pegasus Project. WorkflowGenerator. https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator, 2012. [Online; acessado em 1 de setembro de 2013]. 16

[PH12] D.A. Patterson e J.L. Hennessy. *Computer Organization and Design: The Hardware/-software Interface.* Morgan Kaufmann Series in Computer Graphics. Morgan Kaufmann, 2012. 5

[Ras11] Neil Rasmussen. Determining Total Cost of Ownership for Data Center and Network Room Infrastructure. Relatório técnico, Schneider Electric, Paris, 2011. 8

[RSR+07] Suzanne Rivoire, Mehul A Shah, Parthasarathy Ranganathan, Christos Kozyrakis e Justin Meza. Models and Metrics to Enable Energy-Efficiency Optimizations. (December):39–48, 2007. 3

[Sin07] O. Sinnen. *Task Scheduling for Parallel Systems.* Wiley Series on Parallel and Distributed Computing. Wiley, 2007. 6

[Sta08a] Standard Performance Evaluation Corporation. Hewlett-Packard Company ProLiant ML110 G3 (Historical). http://www.spec.org/power_ssj2008/results/res2011q1/power_ssj2008-20110127-00342.html, 2008. [Online; acessado em 1 de setembro de 2013]. 16

[Sta08b] Standard Performance Evaluation Corporation. Hewlett-Packard Company ProLiant ML110 G5. http://www.spec.org/power_ssj2008/results/res2011q1/power_ssj2008-20110124-00339.html, 2008. [Online; acessado em 1 de setembro de 2013]. 16

[VMwa] VMware. How VMware Virtualization Right-sizes IT Infrastructure to Reduce Power Consumption. 3

[VMwb] VMware. VMware vMotion: Migração em tempo real de máquina virtual. http://www.vmware.com/br/products/vsphere/features-vmotion. [Online; acessado em 10 de setembro de 2013]. 7, 8