Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Dissertação de Mestrado

Estimando o Valor de uma Grade P2P usando Provedores de Infraestrutura como Serviço como Parâmetro de Comparação

Edigley Pereira Fraga

Campina Grande, Paraíba, Brasil
Agosto - 2011

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Estimando o Valor de uma Grade P2P usando Provedores de Infraestrutura como Serviço como Parâmetro de Comparação

Edigley Pereira Fraga

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Redes de Computadores e Sistemas Distribuídos

Francisco Vilar Brasileiro
Dalton Dario Serey Guerrero
(Orientadores)

Campina Grande, Paraíba, Brasil

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

F811e Fraga, Edigley Pereira.
Estimando o valor de uma grade P2P usando provedores de infraestrutura como serviço como parâmetro de comparação / Edigley Pereira Fraga. - Campina Grande, 2011.
85f.: il.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.
Orientadores: Prof. Francisco Vilar Brasileiro e Prof. Dalton Dario Serey Guerrero.
Referências.

1. Redes de Computadores. 2. Grades Computacionais. 3. Computação na Nuvem. 4. Sistemas P2P. I. Título.

CDU 004.7 (043)

"ESTIMANDO O VALOR DE UMA GRADE P2P USANDO PROVEDORES DE INFRAESTRUTURA COMO SERVIÇO COMO PARÂMETRO DE COMPARAÇÃO"

EDIGLEY PEREIRA FRAGA

DISSERTAÇÃO APROVADA EM 31.08.2011

FRANCISCO VILAR BRASILEIRO, Ph.D
Orientador(a)

DALTON DARIO SEREY GUERRERO, D.Sc
Orientador(a)

NAZARENO FERREIRA DE ANDRADE, D.Sc
Examinador(a)

HERMES SENGER, Dr.
Examinador(a)

CAMPINA GRANDE - PB

Resumo

Há anos as *Grades de Desktops* têm sido usadas como solução de baixo custo para a execução de aplicações massivamente paralelas (BoT, do inglês *Bag-of-Tasks*). A despeito do longo tempo de uso, até o presente momento tem sido difícil estimar o valor financeiro que uma grade de *desktops* fornece aos seus usuários. Recentemente, a *Computação na Nuvem*, por meio da provisão de *Infraestrutura como Serviço* (IaaS), emergiu como alternativa atraente para a execução de aplicações BoT, oferecendo, também a baixo custo, maior controle de *Qualidade de Serviço* e diferentes opções de execução. Diferentemente do ambiente de grade, na IaaS o conceito de valor monetário está umbilicalmente ligado à plataforma.

Neste trabalho, é realizada uma comparação de desempenho entre as duas plataformas ao serem utilizadas para a execução de BoT típicas de *e-science*. Após uma análise do *trade-off* custo/desempenho resultante das execuções nas diferentes opções na nuvem, são identificadas opções representativas e seus custos são utilizados para estimar o valor que uma grade de *desktops* fornece a seus usuários. Também são considerados os custos extras incorridos aos participantes, objetivando identificar se a plataforma de grade se mantém efetivamente como solução de baixo custo diante da onipresença do paradigma da computação na nuvem.

Os resultados obtidos confirmam a efetividade da grade de *desktops* como solução de baixo custo e, devido a limitações de escalabilidade dos atuais provedores de IaaS, a grade consegue obter uma valoração competitiva mesmo diante da melhor opção atualmente existente na nuvem. Ainda, para usuários preocupados majoritariamente com o custo, a grade se apresenta como opção superior à nuvem mesmo nos cenários com poucos participantes. Em contrapartida, para cenários de grades pequenas e usuários cuja principal preocupação está no desempenho, e não no custo, a grade não se mostra como opção efetiva e a nuvem se apresenta como a opção mais adequada.

Abstract

Desktop grids have been used as a low cost solution to run *Bag-of-Tasks* (*BoT*) e-science applications for at least the last fifteen years. Despite their best-effort characteristic, the appealing low cost of deploying and operating desktop grids make them a good fit for the execution of such embarrassingly parallel applications. Recently, the cloud computing paradigm, by means of Infrastructure as a Service (*IaaS*), has emerged as an alternative to run these kinds of applications, offering more control on the quality of service delivered, without requiring too large expenditures. Until now, it was not easy to assess the value that a desktop grid can bring to users of *BoT* applications using business-oriented metrics. Here, we try to shed some light on this issue by using a cloud computing platform as a cost reference.

In this work, we compare the performance yielded by both types of infrastructure for the execution of typical e-science *BoT* applications. Afterward, we analyse the trade-off performance/cost resulting from the executions on the different options in the cloud in order to identify the representative options. Then their execution costs are used to estimate the value that a desktop grid can bring to its users of *BoT* applications. We also consider the extra costs incurred by the peers in donating its idle resources to the grid aiming to verify its effectiveness as a low cost solution even before the omnipresent cloud computing paradigm.

Our simulation results corroborate the effectiveness of the grid as a low cost solution. Due to scalability limitations from the current *IaaS* providers, considering the cost-benefit relation, desktop grids manage to achieve a competitive valuation even before the best option currently offered by the cloud. To users whose the main concern is the cost, even in scenarios with just a few peers, the grid outperforms the cloud. On the other hand, to small grids and users whose main concern is the performance, the grid isn't a so effective solution and the cloud stands out as the best option.

Agradecimentos

Começo os agradecimentos pelos orientadores. *Dalton* e *Fubica* (*Francisco Brasileiro*) me deram a liberdade e a autonomia necessárias para que eu me sentisse bem realizando uma atividade de pesquisa. Ao mesmo tempo, cobraram resultados nos momentos certos e, sempre, de forma muito respeitosa: foram os chamados à responsabilidade. Quando surgiram indefinições e princípios de desespero, pude contar com a paciência e a experiência destes dois mentores, que souberam me tranquilizar e apontar os caminhos que levaram às soluções.

Além dos orientadores, contei com a ajuda de diversos colegas do LSD. Em alguns casos, em conversas mais formais, em outros, em um breve papo nos corredores ou na sagrada hora do café. Agradecimentos especiais vão para *Marcus Carvalho*, *Matheus Gaudencio*, *Lesandro Ponciano*, *Paulo Ditarso* e *Álvaro Degas*. Há também os que não participaram diretamente do trabalho, mas, de uma forma ou de outra, contribuíram para o andamento das atividades do mestrado. Dentre estes estão *Abmar Granjeiro*, *Adabriand Furtado*, *Ana Cristina*, *Carla Sousa*, *Cícero Alan*, *David Candeia*, *Fábio Jorge*, *Geraldo Abrantes*, *Giovanni Farias*, *Jaindson Valentim*, *Jonhnny Weslley*, *Katyusco Santos*, *Patrick Maia*, *Pryscilla Dóra*, *Renato Miceli*, *Ricardo Araújo*, *Rostand Costa* e *Thiago Emmanuel*. Poder trabalhar com tantas pessoas competentes me permitiu um aprendizado contínuo e profícuo.

Não podem faltar os agradecimentos aos responsáveis pelo suporte do LSD, os atuais, *Antônio Flávio* e *Heitor Meira*, e os anteriores, *Tomás de Barros* e *Vinícius Aguiar*, sempre pacientes com nossas demandas. Também agradeço a *Elayne Leal* e a *Josicleide Souza*, que ajudam a fazer do LSD um ambiente de trabalho de altíssimo nível.

Destaco também o corpo de professores do LSD, *Raquel Lopes*, *Lívia Sampaio*, *Andrey Brito* e *Nazareno Andrade*. Em diversos momentos, particularmente nas Conversas LSD, o trabalho recebeu valiosos comentários destes professores. Na reta final, *Nazareno*, membro da minha banca de avaliação, deu uma contribuição providencial para melhorar a qualidade desta dissertação. Agradeço também ao professor *Hermes Senger*, da Universidade Federal de São Carlos (UFSCar), examinador externo que participou da banca.

Por fim, agradeço à CAPES, por seu importante papel na expansão e consolidação da pós-graduação *stricto sensu* no Brasil.

Conteúdo

Lista de Símbolos

Notações e Símbolos Definidos e Utilizados ao Longo de Toda a Dissertação (Agrupados por Afinidade)

Notação	Descrição
G	Grade entre pares, um conjunto de *sites*
S	Um *site* (*peer*), um conjunto de máquinas
S_i	O i-ésimo *site* de uma grade G
M	Uma Máquina, uma n-tupla de processadores
P	Um processador, elemento de uma máquina M
ν	Capacidade computacional de um processador
C	Infraestrutura de nuvem de instâncias *spot*
Υ	Usuários da infraestrutura de nuvem
B	n-tupla com lances (*bid-values*) dos usuários da nuvem
O	Série temporal com a oscilação do *spot-price*
Π	*spot-price*: um instante de tempo e um preço
t	Instante de tempo discreto
π	Um valor monetário (preço)
L	Número máximo de instâncias por usuário na nuvem
$inst$	Um tipo particular de instância de C
$m1$	Família de instâncias de propósito geral (*Standard*)
$c1$	Família com proporcionalmente mais CPU que memória (*High-CPU*)
$m2$	Família com mais memória que CPU (*High-Memory*)

Continua na próxima página...

Notação	Descrição
W	Um *workload* de *jobs* do tipo *BoT*
W^s	O *workload* proveniente de um *peer* específico s
W^u	O *workload* proveniente de um usuário específico u
J	Um *job*
J_i	O i-ésimo *job* do *workload* W
J_i^s	O i-ésimo *job* do *workload* W^s
T	Uma tarefa
τ	O tempo de execução de uma tarefa
D^s	Desempenho comparativo obtido pelo *peer* s
$\overline{D}$	A média de D^s para todos os *peers*
D	Usado indistintamente para D^s e $\overline{D}$
CMT	Custo médio por tarefa
$user(J)$	Identificação do usuário que submeteu o *job* J
$peer(J)$	Identificação do *peer* ao qual $user(J)$ pertence
$ST(J)$	Tempo de submissão do *job* J
$FT(J)$	Tempo de término do *job* J
$FT^G(J)$	Tempo de término do *job* J quando executado na grade
$FT^C(J)$	Tempo de término do *job* J quando executado na nuvem
$makespan^G(J)$	*makespan* do *job* J na grade: $FT^G(J) - ST(J)$
$makespan^C(J)$	*makespan* do *job* J na nuvem: $FT^C(J) - ST(J)$
MIN_MKSP	O menor *makespan* obtido dentre todas as instâncias de C
$cost(J, i)$	Custo de execução da i-ésima tarefa do *job* J
$mksp(inst)$	O *makespan* obtido ao executar W em $inst$
$cost(inst)$	O custo referente à execução de W em $inst$
MIN_COST	O menor *custo* obtido dentre todas as instâncias de C
pc	A importância que o usuário confere ao custo da execução
pm	A importância conferida ao *makespan* ($pm = 1 - pc$)
$sat(inst, pc)$	Satisfação obtida ao se executar W em $inst$ considerando pc

Continua na próxima página...

Notação	Descrição
M_g	Soma dos *makespans* dos *jobs* de W na grade
M_c	Soma dos *makespans* dos *jobs* de W na nuvem
C_c	Custo total da execução de W na nuvem
V_g	Valor da grade
Eff^{V_g}	Indica quão efetivo é V_g (como se comporta em relação a C_g)
$VPHM$	Valor por hora-máquina proporcionado pela grade
$CDPM(S, m)$	Custo diário de energia da m-ésima máquina do *site* S
$cons^o(S, m)$	Cons. de energia da m-ésima máquina do *site* S quando em ociosidade
$cons^s(S, m)$	Cons. de energia de sobrecarga da m-ésima máquina do *site* S
$cons^t(S, m)$	Cons. de energia total da m-ésima máquina do *site* S
$CDPM$	Custo diário de energia por máquina homogênea
$CDPM_{min}$	Custo diário de energia em uma máquina eficiente energeticamente
$CDPM_{max}$	Custo diário de energia em uma máquina ineficiente energeticamente
c^o	Cons. de energia por máquina homogênea quando em ociosidade
c^s	Cons. de energia de sobrecarga por máquina homogênea
c^t	Cons. de energia total por máquina homogênea
k	Preço de 1 *kWh*
r	Taxa de ociosidade das máquinas em horário normal de trabalho
h	Total de horas diárias de trabalho
u	Taxa de utilização da máquina pela grade
d	Um intervalo de tempo em número de dias
np	Número de *peers* na grade: $np = \|G\|$
nm	Número de máquinas por *peer* em uma grade homogênea
C_g	Custo extra de manutenção da grade
C_g^{min}	C_g para uma grade cujas máquinas possuem $CDPM_{min}$
C_g^{max}	C_g para uma grade cujas máquinas possuem $CDPM_{max}$
USD	Abreviação para dólar americano (*United States Dollar*)
TCO	Custo Total de Propriedade, do inglês *Total Cost of Ownership*

Continua na próxima página...

Notação	Descrição
QoS	Qualidade de Serviço, do inglês *Quality of Service*
SLA	Acordo de Nível de Serviço, do *inglês Service Level Agreement*
$P2P$	abreviação do inglês *peer-to-peer*
NoF	Rede-de-Favores do OurGrid (*do inglês Network-of-Favors*)
WQR	Estratégia de Escalonamento do OurGrid (*WorkQueue with Replication*)
$IaaS$	Infraestrutura como Serviço, do inglês *Infrastructure as a Service*
TI	Tecnologia da Informação
AWS	*Amazon Web Services*
$EC2$	*Amazon Elastic Compute Cloud*
ECU	*EC2 Computing Unit*, unidade virtual de computação do *EC2*

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

O acesso a grandes quantidades de recursos computacionais tem impactado a forma como a ciência é conduzida em diversos campos do conhecimento. O uso intenso da computação para gerar e processar dados científicos, que recebe a denominação de *e-science* [4], é possível porque boa parte das aplicações pode ser facilmente executada de forma paralela em diversos recursos computacionais. Isto, por sua vez, permite acelerar consideravelmente ciclos de pesquisa que consistem em processos iterativos de geração e análise de dados.

Algumas destas aplicações são conhecidas como sacos de tarefas (ou *BoT*, do inglês *Bag-of-Tasks*) [36] em função de serem constituídas por uma grande quantidade de tarefas que podem ser executadas de forma independente, o que facilita consideravelmente as atividades de escalonamento e tolerância a falhas de tarefas. A característica mais importante dessas aplicações é a não exigência de uma rígida garantia de qualidade de serviço (*QoS* - do inglês *Quality of Service*) por parte da plataforma de execução. Apesar da simplicidade, aplicações *BoT* são utilizadas em diversas áreas, tais como mineração de dados, pesquisas massivas (como quebra de chave de criptografia), simulações que utilizam método de Monte Carlo, varredura de parâmetros e manipulação de imagens, apenas para citar algumas.

Desde a década de 90, grades de *desktops* têm sido amplamente utilizadas para a execução de aplicações *BoT*. Em particular, se consolidaram como as plataformas mais populares para a execução de aplicações de *e-science* [3; 59; 44]. Tais infraestruturas se baseiam no aproveitamento oportunista de recursos, utilizando ciclos ociosos de máquinas *desktops* não dedicadas. Muitos tipos de grades de *desktops* oportunistas foram propostas, desenvolvidas e implantadas. Um dos primeiros representantes destes sistemas foi o projeto Condor [32].

Inspirados pelo sucesso das primeiras grades de *desktops*, plataformas de computação voluntária, tais como a resultante do projeto pioneiro SETI@home [3] e de outros similares baseados no *middleware* BOINC [1], conseguiram agregar milhões de *desktops* de usuários domésticos espalhados por todos os continentes. Mais recentemente, grades entre pares, tais como as suportadas pelo *middleware* OurGrid [47], também têm sido implantadas e utilizadas para a execução de uma variedade de aplicações *BoT* da comunidade de *e-science* [63; 12; 11; 52].

Em meados da década passada, a computação na nuvem, na forma de provisão de Infraestrutura como Serviço (*IaaS* - do inglês *Infrastructure as a Service*), emergiu como uma alternativa atraente para a execução de aplicações *BoT*. Em particular, destaca-se a IaaS baseada no modelo de negócio *Spot Instances* [57], por apresentar, em troca de uma flexibilização na qualidade do serviço oferecido, os menores preços. Para o usuário, uma característica notável presente no paradigma da computação na nuvem é a elasticidade: capacidade de ser virtualmente e quase que instantaneamente expandida ou reduzida. Ainda, tal elasticidade se dá sob demanda e controle direto do cliente. Some-se a isso o fato de a benesse da elasticidade não implicar qualquer custo extra ou obrigação de longo prazo para o cliente [13]. A título de exemplo, em teoria, utilizar um único recurso computacional por 1.000 horas implica o mesmo custo resultante da utilização de 1.000 recursos computacionais durante uma hora. São essas facilidades, obviamente somadas aos baixos custos, que podem atrair usuários com aplicações *BoT*, em especial os da comunidade de *e-science*, para as ofertas de IaaS.

Contudo, todo serviço oferecido tem um custo subjacente associado. O fato de o paradigma da computação na nuvem liberar o usuário do planejamento de capacidade de longo prazo não significa que tal atividade deixe, magicamente, de existir: ela é apenas terceirizada para o provedor. O provedor de *IaaS* tem o desafio de lidar com os custos de sobreprovisão e com o risco de sub-provisão. Ao mesmo tempo, precisa manter a lucratividade do negócio. É por essa razão que os atuais provedores de *IaaS* impõem limites na quantidade de recursos que um único usuário pode alocar automática e simultaneamente [28; 22]. Por exemplo, a empresa *Amazon*, por meio do *Amazon Web Services (AWS)*, reconhecidamente um dos mais importantes e atuantes provedores do mercado, limita a 20 as instâncias que podem ser automaticamente adquiridas de seu serviço *Elastic Compute Cloud*

(EC2) [27] no modelo *on-demand* e a 100 instâncias no modelo *spot*. Apesar de pouco divulgada, a existência de tais limites na elasticidade tem impacto nos benefícios que um usuário de aplicações do tipo *BoT* pode obter, visto que a redução no tempo de resposta de tais aplicações está diretamente relacionada à quantidade de tarefas que executam simultaneamente.

A despeito do longo tempo de uso, até o presente momento tem sido difícil estimar o valor que uma grade de *desktops* oferece aos usuários de aplicações *BoT*. Trata-se de uma tarefa especialmente difícil caso se queira usar métricas de negócio (valor monetário), visto que a receita financeira resultante da execução de aplicações de *e-science* é comumente indefinida. Em particular, esta lacuna é ainda mais evidente nas grades entre pares, visto que são baseadas na federação de recursos pertencentes a diferentes domínios administrativos, cada um possuindo suas especificidades. Neste trabalho, com o objetivo de realizar tal estimativa, é utilizada uma abordagem comparativa entre os serviços providos por uma grade de *desktops* e uma plataforma de computação na nuvem. A ideia fundamental é usar os custos de execução na plataforma de computação na nuvem como referência para estimar o valor proporcionado pela grade de *desktops*. Diferentemente do ambiente de grade, na IaaS o conceito de valor monetário está umbilicalmente ligado à plataforma e todo serviço provido possui um custo (preço) explícito associado.

A metodologia adotada é baseada na simulação das duas plataformas de execução, sendo ambas submetidas a um mesmo *workload* (carga de aplicações *BoT*). Em razão da *expertise* de nosso grupo de pesquisa, o modelo assumido é de uma grade entre pares como representativa das grades de *desktops*. Para cada plataforma simulada é realizada a medição do tempo entre a submissão e a conclusão de cada aplicação executada, o que resulta na métrica conhecida como *makespan*. Para estimar o custo da execução do *workload* na nuvem, é utilizado o método de precificação e os preços efetivamente praticados pelo *AWS* [57].

O "valor" proporcionado por uma plataforma de execução é definido como sendo inversamente proporcional ao *makespan* e ao custo de execução associado. Além de estimar o valor da grade, a análise combinada das métricas *makespan* e custo nos permite estimar os limites do custo operacional extra em que incorrem os mantenedores de uma grade de *desktops* para que essa plataforma seja mais vantajosa do que a contratação de uma infraestrutura de computação na nuvem. Ao estimar também o custo da grade, pode-se obter o seu valor efetivo, isto é, o valor proporcionado menos o custo extra incorrido.

O objetivo geral do trabalho apresentado nesta dissertação é estimar o valor monetário de uma grade entre pares. Como objetivos específicos, resultantes da análise de desempenho e da valoração da grade, é avaliada a efetividade desta plataforma como solução de baixo custo para a execução de aplicações *BoT* típicas e aferido o desempenho obtido em relação às opções disponíveis na oferta de *Infraestrutura como Serviço*.

Os resultados obtidos confirmam a efetividade da grade como solução de baixo custo e, devido a limitações de escalabilidade dos atuais provedores de IaaS e considerando a relação custo/benefício, a grade consegue obter uma valoração competitiva mesmo diante da melhor opção atualmente existente na nuvem. Ainda, para usuários preocupados majoritariamente com o custo, a grade se apresenta como opção superior à nuvem. Para cenários de grades pequenas e usuários cuja principal preocupação está no desempenho, e não no custo, a grade não se mostra como opção efetiva e a nuvem, pelo menos nominalmente, apresenta-se como a opção mais adequada.

O restante da dissertação está organizado da seguinte forma: no Capítulo 2 é apresentada uma contextualização em relação aos sistemas sob estudo, introduzidos os modelos de negócio praticados na oferta de IaaS e discutidos os trabalhos relacionados. No Capítulo 3 os sistemas são formalizados e são introduzidas as notações utilizadas ao longo da dissertação. Por sua vez, o Capítulo 4 descreve como os modelos de sistema foram simulados. O Capítulo 5 define os parâmetros de simulação e apresenta as métricas utilizadas para a comparação entre os sistemas analisados. No Capítulo 6 são apresentados e discutidos os resultados da comparação entre os sistemas e realizadas as estimativas de custo e de valor para grades entre pares de diferentes tamanhos. Por fim, no Capítulo 7 são apresentadas as considerações finais, elencados potenciais efeitos das simplificações feitas no presente estudo e também levantados os possíveis trabalhos futuros.

Capítulo 2

Contextualização

2.1 Grades Oportunistas

Segundo Foster, uma grade computacional é um sistema que coordena uma infraestrutura composta por *hardware* e *software* que não está sujeita a um controle centralizado, utilizando recursos distribuídos em diversos domínios administrativos para compor serviços não triviais [33]. Embora bastante abstrata, essa definição deixa claro um aspecto fundamental das grades, que é o compartilhamento de recursos, isto é, uma grade se comporta como uma federação de recursos possíveis de serem acessados por usuários participantes da grade. As grades computacionais foram desenvolvidas com o objetivo de proporcionar uma infraestrutura para execução de aplicações paralelas ou armazenamento de grandes quantidades de dados em máquinas geograficamente dispersas. Também são possíveis serviços de mais alto nível, compostos a partir de serviços básicos como os exemplificados.

Muitos trabalhos se concentraram em aproveitar a capacidade ociosa de máquinas *desktops* individuais para montar grades computacionais [20; 32; 64; 3; 1]. Estes *desktops* podem estar localizados em escolas, laboratórios, escritórios e até mesmo nos lares de usuários da Internet. Às grades resultantes desta abordagem se deu o nome de Grades Oportunistas, devido ao caráter não dedicado de seus recursos subjacentes. Cabe, ainda, a distinção entre grades de *desktops* corporativas e grades voluntárias, sendo estas compostas basicamente por recursos computacionais ociosos de usuários domésticos e aquelas por recursos ociosos de instituições (empresas, escolas, universidades, etc.).

São exemplos de grades voluntárias os diversos projetos "@home", como o precursor

SETI@home [3], o Einstein@home [44] e o Folding@home [2], todos atualmente baseados no *middleware* BOINC (*Berkeley Open Infrastructure for Network Computing*) [1]. Uma particularidade das grades voluntárias é que os doadores de recursos não se utilizam da infraestrutura para executar suas próprias aplicações, agindo de forma puramente altruísta, no máximo tendo seus nomes incluídos em um *ranking* de doadores mais participativos. Como exemplos de grades de *desktops* corporativas existem as baseadas nos *middlewares* Condor [32] e XtremWeb [64], além de outras baseadas em *middlewares* proprietários.

Por fim, vale ressaltar que no contexto do presente trabalho serão consideradas as *grades de desktops corporativas entre pares*, isto é, uma evolução das grades de *desktops* corporativas no sentido de formarem comunidades de livre acesso sobre a Internet, referenciadas, por simplicidade, como *grades entre pares* ou ainda *grades P2P* (do inglês *peer-to-peer*).

2.1.1 Grade entre Pares

Diferentemente do que acontece nas grades voluntárias, em uma grade entre pares os usuários/doadores também podem executar suas próprias aplicações *BoT*. Desta forma, a participação em comunidades de grades P2P permite às instituições e aos laboratórios participantes reduzir o chamado Custo Total de Propriedade (TCO, do inglês *Total Cost of Ownership*). Em vez de adquirir novos recursos dedicados para satisfazer suas demandas computacionais, os laboratórios usam *desktops* já adquiridos para outros usos. Como o sistema se beneficia apenas de ciclos ociosos, o uso dos *desktops* se dá de forma não intrusiva, sem prejudicar os usuários locais destes. Além disto, por ser um nó da comunidade, o laboratório pode dispor dos ciclos ociosos dos *desktops* dos demais pares. Isto implica a disponibilidade de um número muito maior de recursos computacionais do que seria possível obter individualmente. Em troca, o laboratório também disponibiliza seus ciclos ociosos para os demais pares da comunidade.

Como visto, grades entre pares possibilitam, a um custo baixo, alavancar as comunidades de *e-science*. Exemplos de *middlewares* bem sucedidos no suporte a grades oportunistas em larga escala incluem o projeto Condor [31] e o OurGrid [21], descrito nos próximos parágrafos.

Na terminologia OurGrid, dá-se a denominação de *comunidade* ao conjunto de laboratórios participantes, também chamados de *sites*, para destacar sua função agregadora de recur-

sos, ou *peers* quando se quer enfatizar a natureza P2P da plataforma. O OurGrid é baseado em uma política *free-to-join*, em que a entrada de novos *sites* é estimulada ao se facilitar, para os novos participantes, o consumo imediato dos recursos disponíveis na infraestrutura da comunidade. Da mesma forma, os laboratórios são estimulados a compartilhar seus recursos de maneira independente e no momento em que o desejarem, sem a necessidade de estabelecer acordos e/ou termos de compromisso prévios com a comunidade. Após a sua adesão, o *site* passa a ser um nó (*peer*) no sistema e poderá, também de maneira autônoma, deixar a comunidade a qualquer momento.

Devido à sua natureza aberta, uma grade entre pares poderia despertar em alguns de seus participantes o comportamento conhecido como *free-riding*. Um *free-rider* (caroneiro) participa da comunidade com o único objetivo de consumir recursos, sem, em contrapartida, compartilhar os seus. Para desencorajar tal comportamento, a plataforma OurGrid prioriza o atendimento aos usuários de acordo com o histórico de doações. Este mecanismo simples e efetivo de incentivo é chamado de Rede-de-Favores (*NoF*, do inglês *Network-of-Favors*) [7] e age de maneira autônoma baseando-se na reciprocidade entre os pares, ou seja, um par doador depende apenas de seu próprio histórico de reciprocidade para decidir quanto de seus recursos cada par consumidor deve receber.

A Figura 2.1 ilustra um exemplo de uma comunidade *OurGrid* composta por 3 *sites*, destacando os principais componentes: *Peer*, *Broker*, *Worker* e *Discovery Service*.

O *Broker* é o componente a partir do qual o usuário submete sua aplicação BoT, sendo responsável por seu escalonamento, isto é, pela alocação de tarefas a máquinas da grade. Um *Broker* pode possuir diferentes estratégias de escalonamento, cada uma encapsulada em um componente *Scheduler* (não representado na figura).

O componente *Worker* é um agente que fica instalado em cada *desktop* pertencente à grade e é responsável por prover um ambiente seguro de execução, tanto para os processos locais à maquina quanto para as tarefas da grade. Por sua vez, o *Peer* é o componente que gerencia o conjunto de máquinas pertencente a cada domínio administrativo (*site*) e que foram disponibilizadas para a grade. Para se unir à grade, um *Peer* notifica um serviço de descoberta (no caso, o componente *Discovery Service*) a respeito de sua existência e é prontamente informado da existência de outros *peers* na grade.

Quando uma aplicação é submetida para o *Broker*, o *Scheduler* primeiramente solicita

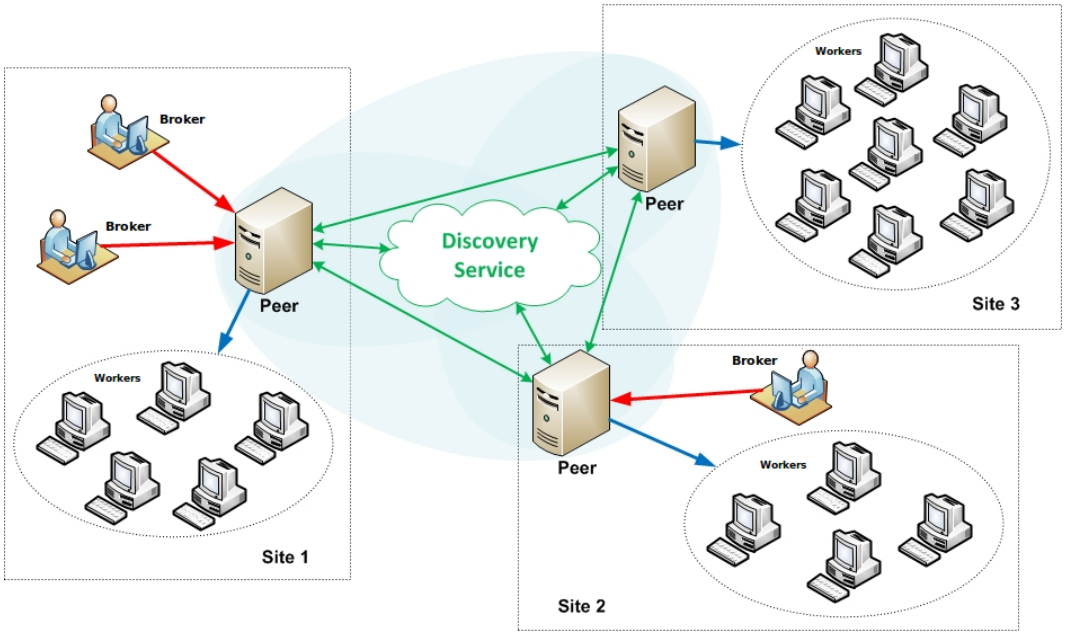


Figura 2.1: Visão Geral de uma Comunidade *OurGrid*

de seu *peer* local uma quantidade de recursos que satisfaça os requisitos da aplicação (por exemplo, 10 máquinas que rodem o sistema operacional *Linux*). Esta quantidade de recursos é normalmente definida em função do número de tarefas da aplicação. O *peer* tenta atender a solicitação com seus recursos locais e, caso estes não sejam suficientes, serão utilizados recursos de outros *peers*.

Com o propósito de garantir que a participação na grade jamais resulte em um desempenho inferior àquele que seria obtido com base apenas nos recursos locais, as tarefas provenientes do *peer* local possuem prioridade sobre as de *peers* remotos que porventura já estiverem executando nos recursos locais. Assim, caso os recursos locais disponíveis não forem suficientes para atender a solicitação feita pelo *Scheduler* e haja tarefas remotas executando em recursos locais, estas serão preemptadas e os respectivos recursos liberados para o *Scheduler*. Ao final, todos os recursos obtidos (locais e remotos) são entregues ao *Scheduler* e este utiliza uma determinada política, ou estratégia, para alocar os recursos para as tarefas.

A estratégia padrão de escalonamento utilizado do OurGrid é a WQR (*WorkQueue with Replication*) [54], que adiciona replicação ao algoritmo *WorkQueue*. Primeiramente, as tarefas são selecionadas aleatoriamente e alocadas em máquinas que estejam disponíveis até que todas as tarefas sejam alocadas. Ao passo que máquinas vão sendo liberadas devido ao

término de tarefas previamente alocadas, são criadas réplicas de tarefas que ainda estejam executando (escolhidas aleatoriamente) e alocadas nos recursos recém-disponíveis. Quando uma das réplicas termina, todas as demais são abortadas.

O principal trunfo da estratégia WQR é permitir que réplicas de uma tarefa que esteja executando em uma máquina lenta possam ser alocadas em máquinas mais rápidas e, desta forma, possibilitar que a tarefa seja concluída mais rapidamente. A análise realizada por Paranhos et al. conclui que a estratégia WQR, às custas de um maior desperdício de recursos, possui desempenho similar ao obtido por heurísticas mais sofisticadas e que utilizam informações sobre o ambiente de execução, informações estas que nem sempre são possíveis de serem obtidas com fidedignidade em ambientes largamente distribuídos e autônomos, como ocorre nas grades computacionais [54].

2.2 Infraestrutura como Serviço

Infraestrutura como Serviço (IaaS, do inglês *Infrastructure as a Service*) é uma das diversas tecnologias que se encontram sob o hiperônimo "computação na nuvem" (*cloud computing*), a exemplo de Software como Serviço (SaaS), Plataforma como Serviço (PaaS) e até mesmo Humanos como Serviço (HaaS) [42]. A "Nuvem" (*Cloud*) é uma metáfora para a Internet, baseando-se em como é diagramada uma rede, e é, ao mesmo tempo, uma abstração para a infraestrutura complexa que lhe é subjacente. Está relacionada com a capacidade de prover serviços, permitindo que usuários os acessem a partir da Internet, sem o conhecimento de onde estão localizados e sem o controle sobre a infraestrutura tecnológica que os suportam. De forma didática, *computação na nuvem* pode ser definida como um modelo no qual a computação (*software*, processamento e armazenamento) está disponível em algum lugar da rede de forma escalável, é acessada remotamente via Internet e seu pagamento se dá sob demanda.

No contexto deste trabalho, o foco está na oferta de Infraestrutura como Serviço, que é uma forma de oferecer infraestrutura computacional (poder computacional e de armazenamento) como um serviço, por meio de provisão imediata, pagamento sob demanda e liberação de compromissos futuros (sem investimentos antecipados). Neste modelo de provisão de infraestrutura de TI (Tecnologia da Informação), em vez de adquirir servidores, licenças

básicas (de Sistema Operacional ou de Gerenciador de Banco de Dados), equipamentos de rede e ter de alocar espaço físico e infraestrutura de suporte (energia elétrica, sistema de arrefecimento e custo de pessoal), o usuário terceiriza estes custos para o provedor de IaaS.

O modelo IaaS se diferencia de seus antecessores (serviços de hospedagem de servidores) na flexibilidade e agilidade com que a provisão do serviço é efetuada, tudo graças ao nível de abstração que é exibido ao usuário (uma máquina virtual e um endereço IP público) e à elasticidade permitida, podendo alocar ou liberar máquinas virtuais de acordo com a demanda. Como o serviço provê a ilusão de recursos computacionais infinitos e a cobrança reflete apenas o que foi consumido, o termo *Computação Utilitária* também é utilizado para a oferta de IaaS [38].

Dentre os principais provedores de IaaS está a empresa *Amazon* [5], com o produto EC2 (*Elastic Compute Cloud*) de sua linha AWS (*Amazon Web Services*) [14]. O *Amazon EC2* [27] é um serviço *web* que provê um ambiente virtual de computação na nuvem. Suas principais características incluem elasticidade (possibilidade de aumento e redução da capacidade sob demanda), total controle das instâncias alocadas (acesso como *root* - usuário administrador), flexibilidade (tipos variados de instâncias e sistemas operacionais), confiabilidade e segurança.

A *Amazon* oferece como garantia um SLA (*Service Level Agreement*) que estipula um nível de 99.95% de disponibilidade por ano de serviço, além de se comprometer com os aspectos de segurança, confiabilidade e escalabilidade anteriormente citados [56]. De forma objetiva, 99.95% de disponibilidade significa que, durante um ano, o serviço pode ficar indisponível por no máximo 4 horas e 23 minutos, aproximadamente. Caso o nível não seja atingido, o usuário é compensado com um crédito que pode variar de 10% (se a disponibilidade ficar entre 99% e 99.9%) a 25% (para disponibilidade inferior a 99%) em relação ao que ele gastou nos 365 dias anteriores à falha de serviço.

É importante destacar que o compromisso firmado é com o serviço como um todo, e não com a confiabilidade de uma alocação em particular. Assim, caso haja algum problema isolado de *hardware* ou de desempenho em uma máquina virtual alocada, a *Amazon*, de antemão, se exime da obrigação de ter que ressarcir o usuário.

Pelo mesmo serviço de alocação de máquinas virtuais, atualmente são oferecidas três formas de precificação, ou modelos de negócio, denominados *on-demand* (sob demanda),

reserved (modelo de reserva) e *spot* (*spot instances*). No primeiro, também chamado de *pay-as-you-go*, os usuários pagam pela hora de computação sem nenhum compromisso de longo prazo. No segundo, o usuário reserva os recursos computacionais por um prazo mais longo, mas paga um valor menor por hora, em comparação com o primeiro modelo. O terceiro modelo, denominado *spot*, é o que oferece os menores preços e consiste em uma espécie de leilão de recursos computacionais que não foram negociados nos outros dois modelos. Vale ressaltar que os preços se referem a uma hora de computação, que é o "grão" de cobrança utilizado pelo *Amazon EC2*. As próximas subseções discutem brevemente cada um desses modelos.

2.2.1 Modelos de Negócio

On-demand Instances (pay-as-you-go)

O modelo de instâncias *sob demanda* permite o pagamento por capacidade computacional sem nenhum compromisso antecipado. Esse modelo apresenta a vantagem de permitir a utilização de recursos computacionais sem a preocupação com os custos e a complexidade de planejar, adquirir e manter recursos de *hardware*. O principal problema com relação ao planejamento de capacidade é a necessidade de se preparar para o pior caso, e isso geralmente significa arcar com os custos de todos os equipamentos mesmo quando esses não estão sendo utilizados (em um momento de baixa demanda, por exemplo), representando uma subutilização de recursos.

A premissa do modelo *pay-as-you-go* é substituir um grande custo fixo de capital inicial por um custo operacional variável. A capacidade computacional oferecida é tarifada em função do tempo de uso e o valor cobrado depende do tipo de instância que se está adquirindo. Os diferentes tipos de instâncias variam de acordo com a quantidade de núcleos, velocidade de processamento, quantidade de memória, armazenamento em disco e o tipo de sistema operacional (detalhes na Subseção 2.2.2).

Reserved Instances

No modelo de *reserva* o usuário faz uso de um aporte inicial que lhe garante, além do benefício de um preço por hora inferior ao praticado no modelo *pay-as-you-go*, a certeza de obter o

recurso computacional a qualquer momento em que esse for necessário, independentemente da demanda de outros usuários. Cada reserva se refere a uma única instância e as opções de duração do contrato de reserva são de 1 ou 3 anos. Do ponto de vista técnico/tecnológico não há diferenças entre os modelos *on-demand* e *reserved*.

Spot Instances

No modelo *spot* [57] os usuários realizam ofertas pela capacidade ociosa de recursos do serviço *Amazon EC2*. Esta capacidade ociosa se dá, por exemplo, devido às instâncias reservadas que não estão sendo usadas ou por uma baixa demanda sobre as instâncias *pay-as-you-go*. Ao solicitar o serviço *spot*, o usuário especifica o tipo e a quantidade de instâncias que deseja alocar, bem como o preço máximo que está disposto a pagar por hora de uso de cada instância. Este valor é chamado de *lance* (ou *bid-value*, em inglês).

A *Amazon* mantém continuamente um preço mínimo para cada tipo de instância *spot* — valor denominado *spot-price*. Assim, se o lance do usuário for maior ou igual ao *spot-price*, a solicitação será atendida. As instâncias alocadas ficarão disponíveis para o usuário até que este as termine ou até que o *spot-price* supere o lance responsável pelas alocações (o que acontecer primeiro). É importante perceber que o preço cobrado pelas instâncias alocadas é o *spot-price* e não o valor do lance. Logo, o valor efetivamente pago será sempre menor ou no máximo igual ao valor do lance.

No caso em que a instância é terminada pelo próprio sistema Amazon EC2, a computação parcial realizada no tempo inferior a uma hora não é cobrada. Em contrapartida, nas terminações realizadas pelo próprio usuário, as horas parciais são cobradas como uma hora completa, tal como nos outros modelos. Todas as horas completas de computação são cobradas pelo valor do *spot-price* corrente, independentemente de, no futuro, a instância ser terminada pelo próprio EC2 ou pelo usuário.

O valor *spot-price* é definido pelo *Amazon EC2* e flutua em função da oferta e da demanda de instâncias *spot*. A Figura 2.2 apresenta um gráfico com a flutuação do *spot-price* para um período de dois meses (dezembro de 2010 e janeiro de 2011). Na figura, pode-se observar que não há uma periodicidade fixa para a mudança ou reavaliação do *spot-price*, podendo a oscilação ocorrer em um intervalo tão curto quanto alguns segundos ou tão longo quanto alguns dias. Além disto, percebe-se que no período que antecede o Natal os preços de

algumas instâncias são colocados em um patamar artificial, provavelmente para forçar preempções e permitir que os recursos sejam utilizados para outros fins, a exemplo de aumentar a disponibilidade de recursos para uso da própria *Amazon*, de forma a dar vazão ao aumento sazonal de acessos em sua atividade de *e-commerce* (Loja Virtual *Amazon*).

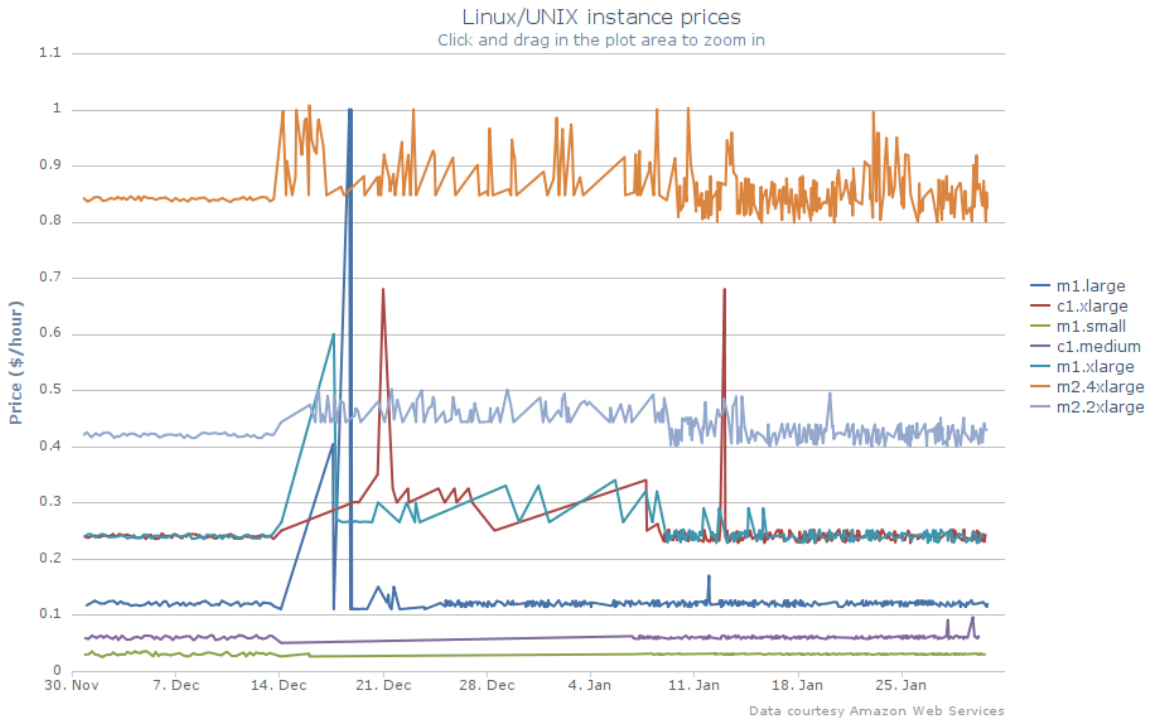


Figura 2.2: Visão Geral da Oscilação do *spot-price*

Um lance feito pelo usuário pode ser válido tanto para uma única vez como pode ser persistente, sendo que este último continua válido mesmo após a instância ter sido terminada devido ao lance ter sido superado pelo valor do *spot-price*. Isto é, no modelo uma-única-vez a oferta só pode ser atendida uma vez, enquanto no persistente a oferta pode ser atendida inúmeras vezes ao longo do tempo. A opção persistente é adequada para os casos de orçamento limitado para a execução, pois o ofertante tem a certeza que vai pagar, no máximo, o valor de seu lance, independentemente de quão flutuante tenha sido o valor do *spot-price*.

Para que um *workload* se adeque ao modelo *Spot Instances* é preciso que a aplicação responsável por executá-lo tenha ciência do caráter oportunista da execução e, por isso, utilize-se de mecanismos para salvar os trabalhos intermediários realizados, algo similar à estratégia de *checkpointing* utilizada, por exemplo, em *middlewares* de computação em grade como o Condor [32]. Além disto, a demanda não deve depender da alocação imediata das instâncias,

deve poder iniciar o trabalho sem intervenção manual e deve, também, estar apta a reiniciar o trabalho automaticamente após a execução ter sido interrompida.

Obviamente, caso o usuário faça um lance de valor alto, as preocupações anteriormente levantadas são mitigadas, embora, em um caso destes, o custo possa se aproximar ou até mesmo superar o praticado no modelo *on-demand*. Ainda assim, há um cenário em que pode ser mais vantajoso pagar por instâncias *spot* o preço praticado nas instâncias *on-demand* em vez de usar as próprias instâncias *on-demand*: caso de aplicações *BoT* cuja quantidade de tarefas supere o limite que cada usuário pode instanciar automaticamente no modelo *on-demand*. A Subseção 2.2.3 discute brevemente tais limitações de escalabilidade.

2.2.2 Tipos de Instâncias

Com relação aos tipos de instâncias, há seis famílias: *Standard, Micro, High-Memory, High-CPU, Cluster Compute e Cluster GPU*. Neste trabalho, o interesse está em apenas três destas famílias: *Standard* (*m1*), *High-Memory* (*m2* - proporcionalmente maior capacidade de memória que de CPU) e *High-CPU* (*c1* - proporcionalmente maior capacidade de CPU que de memória). Os prefixos *m1*, *m2* e *c1* são os utilizados pelo *Amazon EC2* para identificar as diferentes famílias de instâncias e também são utilizados com o mesmo propósito nesta dissertação.

As instâncias *m2* são indicadas para serviços com alto consumo de memória (como Sistemas Gerenciadores de Bancos de Dados, por exemplo) enquanto as *c1* são indicadas para computação de alto desempenho. Vale ressaltar que nada impede que haja aplicações BoT que também tirem vantagem, ou até mesmo necessitem, da maior quantidade de memória presente nas instâncias *m2*.

Detalhes de cada família de instâncias estão presentes na Tabela 2.1. Nessa tabela, o acrônimo ECU (*EC2 Computing Unit*) representa uma unidade virtual de computação definida pelo *Amazon EC2*, equivalendo aproximadamente a um processador *AMD Opteron* 2007 ou *Intel Xeon* 2007 de 1.0–1.2 GHz[1]. O significado de 1 *ECU* é, de fato, apresentado de forma bastante abstrata, conforme pode ser visto no sítio *web* do *Amazon Web Services* [29]. Ainda em relação à tabela, a coluna *CPU* exibindo, por exemplo, "8-ECU [4x2]"

[1]Como, na verdade, tais processadores nunca foram lançados pelos fornecedores citados, especula-se que 1 *ECU* representa uma fatia de 50% de tempo compartilhado de um processador de 2.0-2.4 GHz

deve ser interpretada como "uma instância possuindo capacidade de 8 *ECUs* (8 x 1.2GHz = 9.6GHz) divididos entre 4 núcleos virtuais com 2 *ECUs* (2 x 1.2GHz = 2.4GHz) em cada um.

Tabela 2.1: Tipos de Instâncias no Serviço *Amazon EC2*

Tipo[1]	**Mem**[2]	**CPU**[3]	**Preço por hora (em *USD*)** [4]		
			Spot	**Reserved**	**On-Demand**
$m1.small$	1.7	1-ECU [1x1]	0.030	0.03	0.085
$m1.large$	7.5	4-ECU [2x2]	0.124	0.12	0.34
$m1.xlarge$	15	8-ECU [4x2]	0.250	0.24	0.68
$c1.medium$	1.7	5-ECU [2x2.5]	0.059	0.06	0.17
$c1.xlarge$	7	20-ECU [8x2.5]	0.240	0.24	0.68
$m2.xlarge$	17.1	6.5-ECU [2x3.25]	0.170	0.17	0.50
$m2.2xlarge$	34.2	13-ECU [4x3.25]	0.435	0.34	1.00
$m2.4xlarge$	68.4	26-ECU [8x3.25]	0.822	0.68	2.00

[1] Prefixos: *m1 (Standard), m2 (High-Memory)* e *c1 (High-CPU) Instances* – [2] Em Gigabytes – [3] Em número de *ECU* – [4] Em fevereiro de 2011

2.2.3 Limitações

Algumas restrições, contudo, existem. Cada usuário está limitado a apenas 20 instâncias simultâneas nos modelos *on-demand* e *reserved* e a um limite de 100 instâncias no modelo *spot* [28]. Para usar um número maior de instâncias, o usuário deve fazer uma solicitação formal através de um formulário padrão, denominado *Amazon EC2 Instance Request*, explicando detalhadamente o seu caso de uso. Entre outros pontos, é necessário informar o número de instâncias além do limite, por quanto tempo elas serão utilizadas e fornecer uma justificativa plausível para o porquê de tais instâncias extras serem necessárias. A informação da justificativa é necessária para a Amazon se resguardar de um uso malicioso de seus recursos, a exemplo de quebras de chaves de criptografia ou ataques distribuídos de negação de serviço (*DDoS attack*, do inglês *distributed denial-of-service attack*) a terceiros. Posteriormente, o formulário é analisado pela equipe do *Amazon EC2* e a solicitação poderá, ou não, ser aprovada.

Tal nível de burocracia vai de encontro ao conceito de auto serviço, comumente utilizado na computação na nuvem e responsável tanto pela redução de custos por parte do provedor quanto pela possibilidade de acesso quase instantâneo aos serviços, bastando para isso a posse de um cartão de crédito.

A existência de tal limitação na escalabilidade, além dos aspectos de segurança anteriormente citados, também tem uma motivação relacionada à lucratividade do negócio, conforme verificado por Costa et al. [22]. Visto que o provedor tem o desafio de lidar com os custos de sobre-provisão e com o risco de sub-provisão, ao mesmo tempo em que precisa manter sua lucratividade, a imposição do limite é compreensível e este é escolhido de forma a balancear os aspectos de sobre e sub-provisão.

Para muitos usuários e aplicações, os limites praticados atualmente não são críticos, isto é, com um máximo de 100 ou mesmo de 20 instâncias todas as necessidades são atendidas. Um exemplo é o caso de manutenção de um serviço *web* que roda a maior parte do tempo em uma única instância e apenas em momentos de picos de acesso se faz necessário escalar mais instâncias para realizar um balanceamento de carga. Na prática, apenas algumas unidades a mais já é suficiente. Já para aplicações do tipo BoT, em que o fator de paralelização é o principal responsável pela diminuição do tempo de resposta, a existência deste limite pode gerar impactos maiores na satisfação do cliente.

Por fim, é importante frisar que tal limite é por região, já que o serviço é organizado por grandes áreas geográficas. Atualmente são cinco regiões: leste (*US - N. Virginia*) e oeste (*US - N. California*) dos Estados Unidos, oeste europeu (Irlanda, *EU - Ireland*), Ásia-Pacífico I (Cingapura, *APAC - Singapore*) e Ásia-Pacífico II (Tóquio, *APAC - Tokyo*).

2.3 Trabalhos Relacionados

Nessa seção são relacionados trabalhos que comparam grades e plataformas de *IaaS*, analisam a relação custo/benefício de execução de aplicações científicas na nuvem e que consideram aspectos de custo e de escalonamento de aplicações do tipo *BoT* em ambientes de computação utilitária. Também é realizada uma breve incursão em trabalhos da Economia que visam estimar o valor de produtos/serviços intangíveis ou de difícil mensuração, como é o caso das grades entre pares.

Kondo et al. [39] analisam a relação custo/benefício da computação na nuvem e de grades de *desktops* baseadas em computação voluntária, destacando os *trade-offs* de desempenho e de custo entre as duas plataformas. Também é verificada a possibilidade de uso conjunto de tais plataformas, objetivando melhorar a efetividade de custo e de desempenho. De maneira similar ao presente trabalho, é analisado o custo de manter uma infraestrutura de computação voluntária, como o SETI@Home ou Folding@Home, e este é comparado com o custo de manter a mesma infraestrutura em recursos oriundos de um provedor de *IaaS*.

Fundamentalmente, os dois trabalhos diferem no tipo de plataforma de grade considerada (voluntária e entre pares). Como consequência, as cargas de trabalho são diferentes e as formas de utilizar os recursos da nuvem também o são, o que impacta o custo final da alocação das instâncias. Kondo et al. consideram o uso ininterrupto das máquinas alocadas na nuvem (isto é, rodam como um serviço), enquanto neste trabalho a alocação é feita apenas enquanto há demanda para a execução. Pode-se dizer que, enquanto em uma grade voluntária a carga de trabalho possui uma única origem (do projeto responsável por sua manutenção), em uma grade entre pares a carga de trabalho se origina de diversas fontes (os usuários individuais da grade, distribuídos nos diversos *sites*).

Garfinkel [34] analisa o custo/benefício de utilizar, em larga escala, os serviços da *Amazon* para um projeto de computação forense, com intensiva demanda computacional e de armazenamento. Os experimentos de Garfinkel representam um caso concreto do problema que o limite imposto aos usuários individuais pode acarretar, visto que, na época (2007), apenas 10 instâncias poderiam ser simultaneamente utilizadas por um único usuário. Apenas após solicitar formalmente e apresentar uma justificativa, foi possível utilizar 100 instâncias de forma simultânea. Já Simmhan et al. [55] propõem um arcabouço para implantação e execução de aplicações científicas na nuvem e avaliam uma aplicação do projeto *Genoma* enquanto esta escala para até 20 máquinas virtuais, considerando os aspectos de portabilidade e de escala. Ambos os trabalhos ilustram o uso prático da nuvem para a execução de aplicações científicas, confirmando que, de fato, a utilização de *IaaS* é uma opção atraente para estas aplicações.

O trabalho realizado por Oprescu et al. [46] envolve o escalonamento com restrição de orçamento para aplicações *BoT* em múltiplos provedores de *IaaS* com diferentes desempenho de *CPU* e custo. Os resultados do trabalho apontam as diferenças de custo e desempenho

das diferentes instâncias disponíveis na nuvem para a execução de aplicações *BoT*, indicando que a escolha da instância deve ser feita de forma bastante criteriosa.

O foco principal do trabalho de Yi et al. [65] é a investigação de como o mecanismo de *checkpointing* pode ser utilizado para reduzir custos de execução de aplicações na nuvem *spot* e, ainda assim, manter alta confiabilidade. Por sua vez, Andrzejak et al. [9] propõem um modelo probabilístico para otimização de custo, desempenho e confiabilidade para a execução de aplicações *BoT* sobre a nuvem *spot*. Estes dois trabalhos ilustram como a nuvem *spot* pode ser utilizada para a execução confiável e de baixo custo de aplicações *BoT*.

De acordo com o levantamento bibliográfico realizado, com exceção da constatação prática de Garfinkel [34], não há nenhum trabalho que considere os efeitos do limite real na quantidade de máquinas que podem ser alocadas simultaneamente para um mesmo usuário da nuvem e a consequente implicação no aumento do *makespan* de aplicações *BoT*. Embora haja trabalhos que consideram problemas de desempenho da computação na nuvem quando comparada com plataformas de grades de serviço [62] e voluntárias [39], o presente trabalho contribui ao investigar a presença de tais problemas quando comparada com grades oportunistas de diferentes tamanhos.

De Vaus et al. [61] estimam, para a sociedade australiana, o valor monetário da contribuição realizada por pessoas idosas em atividades domésticas não remuneradas. Para a estimativa, é utilizado um método de valoração por comparação, denominado custo de substituição. Em essência, o custo de substituição (ou valor de substituição) é o quanto teria de ser pago para substituir um produto/serviço por um de qualidade similar. No trabalho de De Vaus, a valoração dos trabalhos domésticos dos idosos é estimado com base na remuneração que seria paga a terceiros caso o trabalho não fosse realizado pelos idosos.

O método de valoração pelo custo de substituição de produtos é uma das técnicas utilizadas pelas empresas seguradoras de bens para estimar o valor a ser pago em casos de sinistros [60]. Estudos ambientais também utilizam o método para estimar o valor de serviços intangíveis (por exemplo, o valor econômico de planícies alagáveis, de florestas ciliares, etc.) [24; 23; 16; 26; 15; 49; 25].

Conforme adiantado na Introdução (Capítulo 1) e explicado em detalhes na Seção 5.2.2, a estratégia adotada para estimar o custo da grade é baseada também em um custo de substituição. No caso, a substituição do serviço provido pela grade por um de qualidade similar,

provido pela plataforma de *IaaS*. O método utilizado é de valoração indireta [50], visto não haver nenhuma forma explícita de valorar uma grade entre pares.

Capítulo 3

Modelo de Sistema

As próximas seções descrevem formalmente os modelos dos sistemas sob estudo, além de apresentar a terminologia básica utilizada ao longo desta dissertação.

3.1 Modelo de *Workload*

Com o objetivo de formalizar o *workload* de aplicações *BoT*, adaptou-se a notação introduzida por Iosup et al. [37]. O termo *job*, comumente utilizado na modelagem de *workload*, será utilizado como sinônimo de aplicação *BoT*. Na notação, um workload W é uma sequência de *jobs* ordenada pelo tempo de submissão.

Um *job* J_i, por sua vez, consiste em um conjunto não vazio de tarefas T_i, representando, cada uma, uma demanda computacional de τ unidades de tempo submetida no tempo t. Considera-se, também, que as tarefas de um mesmo *job* estão ordenadas pelo tempo de submissão. O valor de τ considera a execução em uma máquina de referência.

$$W = \{J_1, J_2, ..., J_j\},\ j > 0$$

$$J = \{T_1, T_2, ..., T_k\},\ k > 0\ \wedge\ \forall_{i<k,\ j<k}, i < j \Rightarrow T_{i_1} \leq T_{j_1}$$

$$T = (t, \tau),\ t \in \mathbb{N},\ \ \tau \in \mathbb{N}^+$$

Além dos elementos básicos definidos anteriormente, faz-se necessária a introdução de

algumas funções. Como funções de identificação de proveniência, $user(J)$ e $peer(J)$ representam, respectivamente, o usuário que submeteu o *job* J e seu *peer* de origem. Após a introdução de tais funções, pode-se definir filtros para o *workload* original W com base nos usuários e *peers* de origem:

$$W^s = \{J \mid J \in W \wedge peer(J) = s\}$$

$$W^u = \{J \mid J \in W \wedge user(J) = u\}$$

Como pode ser visto nos subconjuntos do *workload* enunciados acima, pode acontecer a existência de um *peer* ou de um usuário no sistema que, ao longo de todo o tempo coberto pelo *workload* não submete *job* algum.

Da sequência W^u se obtém λ_u, o tempo de interchegada dos *jobs* pertencentes a um dado usuário u, ou, dito de outra forma, a taxa com que o usuário u submete *jobs* para execução. Da união dos *jobs* de todos os usuários em uma mesma sequência, o que remonta o *workload* original W, obtém-se uma nova taxa de interchegada λ.

Considerando que as funções $ST(J)$ e $FT(J)$ representam, respectivamente, os tempos de submissão e término do *job* J, o *makespan* é definido como $FT(J) - ST(J)$. Também são utilizadas as variantes $FT^G(J)$ e $FT^C(J)$ para indicar os tempos de término quando das respectivas execuções na grade e na nuvem.

É importante observar que, para um *job* específico J_i, em geral se tem $FT^G(J_i) \neq FT^C(J_i)$, embora $ST(J_i)$ seja o mesmo para ambos os casos. Desta forma, $makespan^G(J)$ e $makespan^C(J)$ representam os *makespans* do *job* J quando executado na grade e na nuvem, respectivamente.

Considera-se como $FT(J)$ o tempo de término da última tarefa pertencente ao *job* J a ser concluída. Por sua vez, $ST(J)$ é o instante de tempo em que o *job* J foi submetido para execução, sendo equivalente ao tempo de submissão da primeira tarefa de J. Por último, $cost(J, i)$ fornece o custo de execução, na nuvem, da i-ésima tarefa do *job* J.

3.2 Modelo de uma Grade entre Pares

Para o propósito de modelagem, uma grade entre pares G é formalmente definida como um conjunto de *sites* (representados individualmente como S_i). Por sua vez, um *site* consiste de um conjunto de máquinas M_i. Em um ambiente federado como o considerado neste estudo há outros recursos compartilhados, tais como espaço de armazenamento, largura de banda e outros serviços de alto nível, no entanto, devido ao tipo de *workload* considerado (aplicações *BoT* de computação intensiva), a modelagem considerando apenas recursos computacionais já é satisfatória.

$$G = \{S_1, S_2, ..., S_{\mathcal{s}}\},\ \mathcal{s} > 1$$

$$S = \{M_1, M_2, ..., M_{\mathcal{m}}\},\ \mathcal{m} > 0$$

Máquinas são modeladas como n-tuplas de processadores, enquanto um processador é uma 1-tupla, com o seu elemento unitário representando a capacidade computacional ν do processador. Atualmente, os processadores das máquinas do tipo *commodity* comercializadas apresentam, ainda, uma divisão em múltiplos núcleos, embora, por simplicidade, tenha-se preferido parar a modelagem formal no nível do processador. Para casos de máquinas com múltiplos núcleos, pode-se utilizar o mesmo modelo, bastando fazer o mapeamento de núcleo para processador.

$$M = (P_1, P_2, ..., P_{\mathcal{p}}),\ \mathcal{p} > 0$$

$$P = (\nu),\ \nu \in \mathbb{N}^+$$

Os conjuntos de recursos que cada *site* possui são disjuntos ($\forall_{i,j}, i \neq j, S_i \cap S_j = \emptyset$). Naturalmente, tal restrição também se aplica a diferentes máquinas e seus respectivos processadores. Ainda, como os recursos são oportunistas e, dessa forma, a volatilidade é algo preponderante, para cada *site* o conjunto de máquinas efetivamente disponíveis para a grade ao longo tempo é um subconjunto daquele originalmente existente.

A Figura 3.1 ilustra o modelo de sistema para a grade entre pares, além de sintetizar e esquematizar as notações utilizadas.

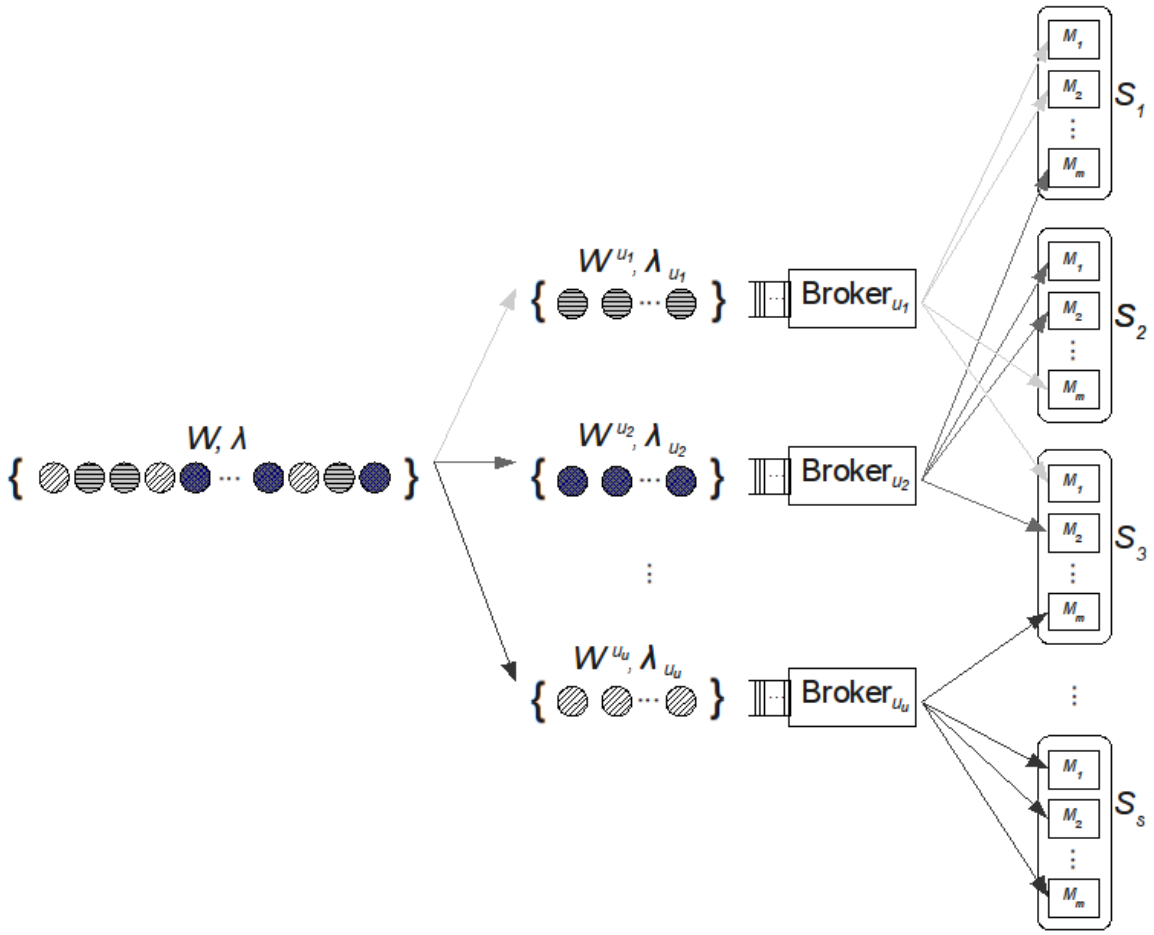


Figura 3.1: Modelo de Sistema para a Grade entre Pares

3.3 Modelo da Nuvem de Instâncias *Spot*

Para o presente trabalho, a plataforma de computação na nuvem C, ou nuvem de instâncias, é modelada como um conjunto de máquinas M_i (tal como definido para a modelagem de grade entre pares) e um limite $L > 0$ que indica o número máximo de instâncias que podem estar simultaneamente alocadas para um único usuário.

$$C = \{M_1, M_2, ..., M_c\},\ c \gg 0$$

Um *spot-price* Π_i é modelado por um par instante de tempo e preço. A oscilação do *spot-price*, O, é expressa como uma série de *spot-prices*.

$$O = (\Pi_1, \Pi_2, ..., \Pi_o),\ o \gg 0\ \wedge\ \forall_{i<o,\, j<o}, i < j \Rightarrow \Pi_{i_1} < \Pi_{j_1}$$

$$\Pi = (t, \pi),\ t \in \mathbb{N},\ \pi \in \mathbb{R}^{+}$$

Há, por fim, a necessidade de considerar os usuários do sistema, Υ, dado que o limite L na elasticidade é aplicado por usuário, e quanto cada usuário está disposto a pagar pela alocação de instâncias. Esta disponibilidade de pagamento, referente aos *lances* (ou *bid-value*, em inglês) de cada usuário, é representada por B.

$$\Upsilon = (\upsilon_1, \upsilon_2, ..., \upsilon_{\mathfrak{u}}),\ \mathfrak{u} > 0$$

$$B = (\beta_1, \beta_2, ..., \beta_{\mathfrak{u}}),\ \mathfrak{u} > 0$$

A Figura 3.2 ilustra o modelo de sistema para a nuvem de instâncias *spot*. Ao observar as Figuras 3.1 e 3.2, nota-se que, para o propósito deste trabalho, os dois sistemas diferem apenas quanto à origem dos recursos utilizados.

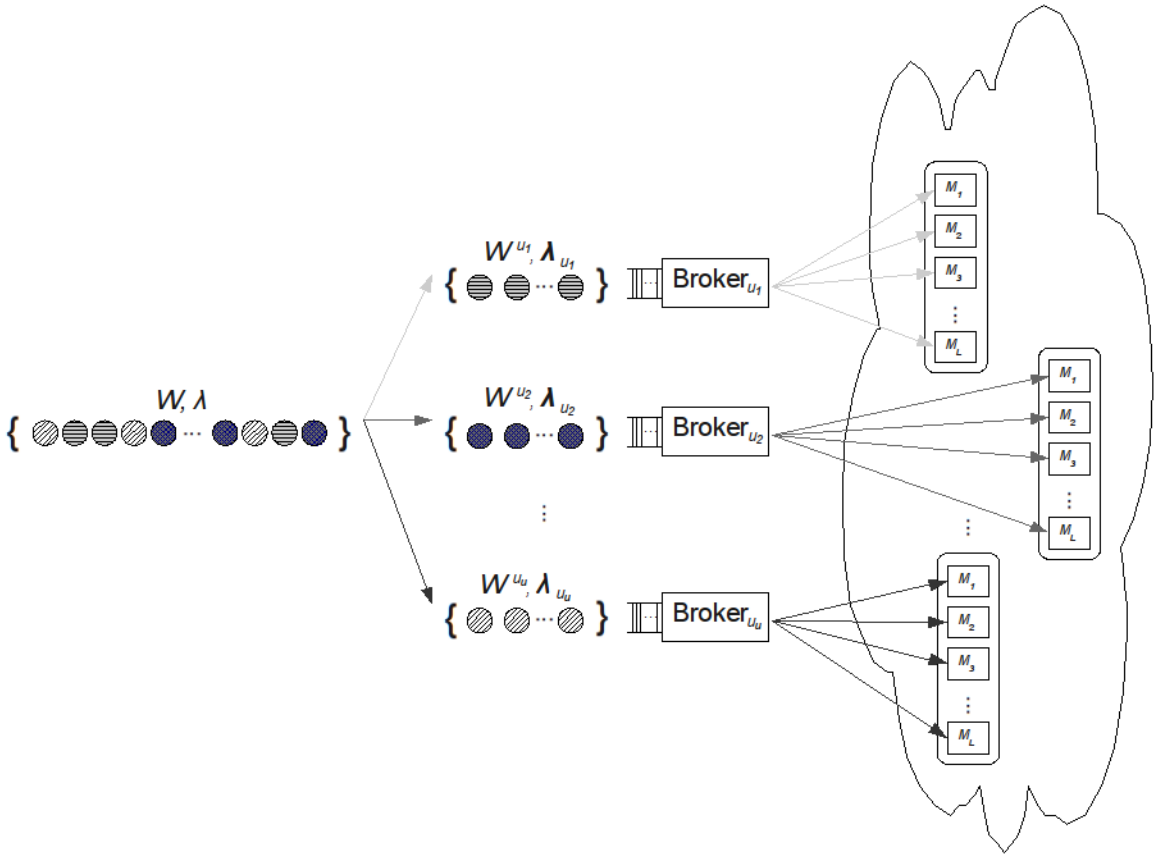


Figura 3.2: Modelo de Sistema para a Nuvem de Instâncias *Spot*

Capítulo 4

Modelo de Simulação

Para avaliar os modelos anteriormente apresentados, optou-se pelo uso de simulação baseada em eventos. Cada um dos sistemas foi simulado com base em suas características essenciais, abstraindo os detalhes que não eram relevantes para os objetivos do trabalho. Por exemplo, latência de rede e falhas de *hardware* ou de aplicações não foram consideradas nos modelos de simulação. As próximas seções apresentam os modelos considerados e o simulador utilizado para a implementação.

4.1 Modelo de Simulação da Grade entre Pares

Para o propósito de simulação, foram utilizados componentes representando *peers*, que agregam, cada um, uma coleção de máquinas heterogêneas, não dedicadas e voláteis. Em conjunto, os *peers* formam uma grade e a interação entre eles é contabilizada em uma implementação de um componente *Network-of-Favors* (*NoF*) [7].

Há um componente *JobScheduler*, responsável por despachar tarefas para as máquinas e gerenciar uma fila de tarefas em cenários de contenção. A estratégia de escalonamento utilizada, primeiramente, aloca todas as máquinas locais que estiverem disponíveis para as tarefas submetidas, restando um subconjunto do *workload* original W, digamos W', que será submetido para escalonamento remoto. Se durante a tentativa de alocação dos recursos locais houver alguma tarefa remota executando localmente e não houver mais recursos locais disponíveis, esta sofrerá uma preempção para dar preferência à demanda local, tal como é a política praticada em uma grade baseada no *middleware* OurGrid, conforme apresen-

tado na Seção 2.1.1. Vale ressaltar que, nas simulações executadas, não foram considerados mecanismos de replicação ou de *checkpointing*, de tal modo que, diante de um evento de preempção, o *JobScheduler* apenas ressubmete a tarefa preemptada.

Para alimentar o simulador com os eventos são utilizados arquivos de texto com registros, ordenados pelo tempo de ocorrência, representando *jobs* (para descrever o *workload* W) e eventos de disponibilidade/indisponibilidade (para descrever a volatilidade das máquinas). Além dos eventos, a configuração da grade (número de *peers* e descrição da capacidade computacional das máquinas de cada *peer*) também é descrita em um arquivo de texto.

4.2 Modelo de Simulação da Nuvem de Instâncias *Spot*

O principal componente no modelo de simulação da nuvem é o escalonador de instâncias *spot* (denominado *SpotScheduler*), que é responsável por registrar quantas instâncias cada usuário já alocou e, a partir do momento em que o limite tiver sido atingido, enfileirar as tarefas até que instâncias sejam liberadas, isto é, até que uma das tarefas pertencentes ao mesmo usuário e que esteja executando termine. Os Algoritmos 1 e 2 descrevem como essa estratégia foi implementada diante de eventos de submissão e término de tarefas, respectivamente. Também são contabilizadas as horas de alocação de cada instância (Algoritmos 2 e 4), de forma a calcular o custo de execução do *workload* na nuvem, além do monitoramento da oscilação do *spot-price* (Algoritmo 3).

Quanto ao Algoritmo 4, vale esclarecer que não há necessidade de contabilização caso a instância já tenha sido liberada, pois ou esta foi vítima de uma preempção e, portanto, horas parciais não são cobradas (conforme discutido na Seção 2.2.1), ou a instância foi liberada pelo próprio usuário e a contabilização já foi realizada antes de ser completada uma hora de uso (linha 11 do Algoritmo 2).

Além dos eventos relacionados aos *jobs*, também há os eventos de oscilação do *spot-price*, alterando o preço das instâncias ao longo do tempo (Algoritmo 3) e que podem disparar preempções caso o novo *spot-price* seja superior ao lance que originou a alocação da instância.

É importante observar que o *SpotScheduler* implementado utiliza uma estratégia gulosa: tenta alocar uma nova instância sempre que não há mais processadores (núcleos, no caso)

Algoritmo 1 TarefaSubmetida(Tarefa ts)

```
1:  if processadoresDisponíveis.size() > 0 then
2:      p ⇐ processadoresDisponíveis.removeFirst()
3:      alocaProcessador( ts, p )
4:  else if máquinasAlocadas.size() < L then
5:      m ⇐ adquireNovaInstância( inst, bid-value )
6:      escalonaContabilizaçãoDeGasto( 3600, m ) {uma hora a partir do tempo corrente}
7:      máquinasAlocadas.add( m )
8:      p ⇐ m.processadores[1]
9:      aloca( ts, p )
10:     n ⇐ m.processadores.length()
11:     processadoresDisponíveis.add( m.processadores[2...n] )
12: else
13:     tarefasEnfileiradas.add( ts )
14: end if
```

livres. Obviamente tal estratégia prioriza a redução no *makespan*, embora possa representar um aumento desnecessário no gasto. Por exemplo, imagine um *job* com 100 tarefas, sendo que 98 destas demandam, cada uma, 1 hora de computação, enquanto as duas restantes demandam meia hora cada. Diante deste *job*, o *SpotScheduler* irá, para um limite $L = 100$, alocar 100 instâncias (supondo que cada uma tenha apenas um núcleo) obtendo ao final um *makespan* de 1 hora para o *job*. É fácil perceber que o mesmo *makespan* poderia ser obtido com apenas 99 instâncias.

Além disto, outro ponto que pode ser otimizado na estratégia de escalonamento é a liberação de instâncias. Pode-se perceber na linha 10 do Algoritmo 2 que, tão logo não haja fila nem tarefas rodando em uma instância, esta é imediatamente liberada. Uma situação possível é esta liberação ocorrer, por exemplo, apenas 2 minutos após o usuário ter sido tarifado por uma hora completa naquela instância. Com essa estratégia de liberação imediata acabariam sendo desperdiçados 58 minutos de computação, visto que quando o próprio usuário termina a instância a cobrança de hora parcial se dá no valor de uma hora completa. Como o escalonamento é feito no contexto do usuário e não para um único *job* em particular, *jobs*

Algoritmo 2 TarefaConcluída(Tarefa tc)

```
1: p ⇐ liberaProcessador( tc )
2: if tarefasEnfileiradas.size() > 0 then
3:     te ⇐ tarefasEnfileiradas.poll()
4:     alocaProcessador( te, p )
5: else
6:     m ⇐ p.máquina
7:     if estãoTodosLiberados( m.processadores) then
8:         máquinasAlocadas.remove( m )
9:         processadoresDisponíveis.remove( m.processadores )
10:        liberaInstância( m )
11:        contabilizaGasto( spot-price )
12:    end if
13: end if
```

Algoritmo 3 NovoSpotPrice(SpotPrice *new-spot-price*)

```
1: spot-price ⇐ new-spot-price
2: for all m in máquinasAlocadas do
3:     if bidValue( m ) < spot-price then
4:         forçaPreempção( m )
5:     end if
6: end for
```

Algoritmo 4 ContabilizaçãoDeGasto(Instância m)

```
1: if máquinasAlocadas.contains( m ) then
2:     contabilizaGasto( spot-price )
3: end if
```

subsequentes poderiam se beneficiar desta hora não completada para reduzir seu custo. Os trabalhos realizados por Yi et al. [65] e Andrzejak et al. [9] analisam diferentes estratégias que podem ser utilizadas para escalonar aplicações na nuvem *spot* de forma a reduzir o custo

final tirando proveito das características inerentes a este modelo de negócio.

Como o objetivo do presente trabalho não está na utilização de um escalonador otimizado, considera-se a estratégia utilizada satisfatória, embora, para uma análise mais criteriosa, os pontos levantados anteriormente deveriam ser levados em consideração. Tal como na simulação da grade entre pares, todos os eventos primários (aqueles que são definidos *a priori*) são passados para o simulador por meio de arquivos de texto.

4.3 Simulador Utilizado

Os modelos de simulação foram implementados no *framework* OurSim [48], idealizado para a simulação de grades entre pares e que oferece suporte à simulação de grades oportunistas. O OurSim foi desenvolvido no Laboratório de Sistemas Distribuídos da UFCG e é utilizado nos trabalhos de pesquisa dos integrantes do laboratório. Por se tratar de um arcabouço de *software*, nele são providos alguns ganchos, para representar diferentes políticas e estratégias de simulação, tornando-o adequado para simular também o modelo simplificado da nuvem de instâncias *spot*, descrito no Capítulo 3. Em particular, o gancho utilizado foi o do componente de escalonamento e a estratégia adotada foi a detalhada na Seção 4.2.

A Figura 4.1 ilustra os principais componentes do *framework*, agrupados em 4 pacotes. O pacote `entities` engloba as entidades básicas de uma grade computacional, a exemplo de *peers*, máquinas, *jobs* e tarefas. No pacote `policies` estão as políticas utilizadas para a tomada de decisão, a exemplo de políticas de escalonamento de tarefas e de compartilhamento de recursos. Conforme apontado na Seção 4.1, as estratégias utilizadas foram a `NoFSharingPolicy` (da Rede-de-Favores), para o compartilhamento, e a `WQSchedulePolicy` (de *Worque Queue*) para o escalonamento.

A simulação é alimentada por *traces* representando a demanda computacional, relativa aos *jobs* e suas tarefas, a disponibilidade de recursos, representada pela volatilidade das máquinas (*workers*) e a caracterização da grade (isto é, os *peers* participantes e a quantidade de recursos disponibilizados por cada um). Estes *traces*, abstraídos por entidades de entrada e saída (membros do pacote `io`), são carregados sob demanda e cada registro é convertido em um evento de sistema (do pacote `events`), sendo este adicionado a uma Fila de Eventos (`EventQueue`). Em última instância, é esta fila a responsável por guiar a simulação, o que

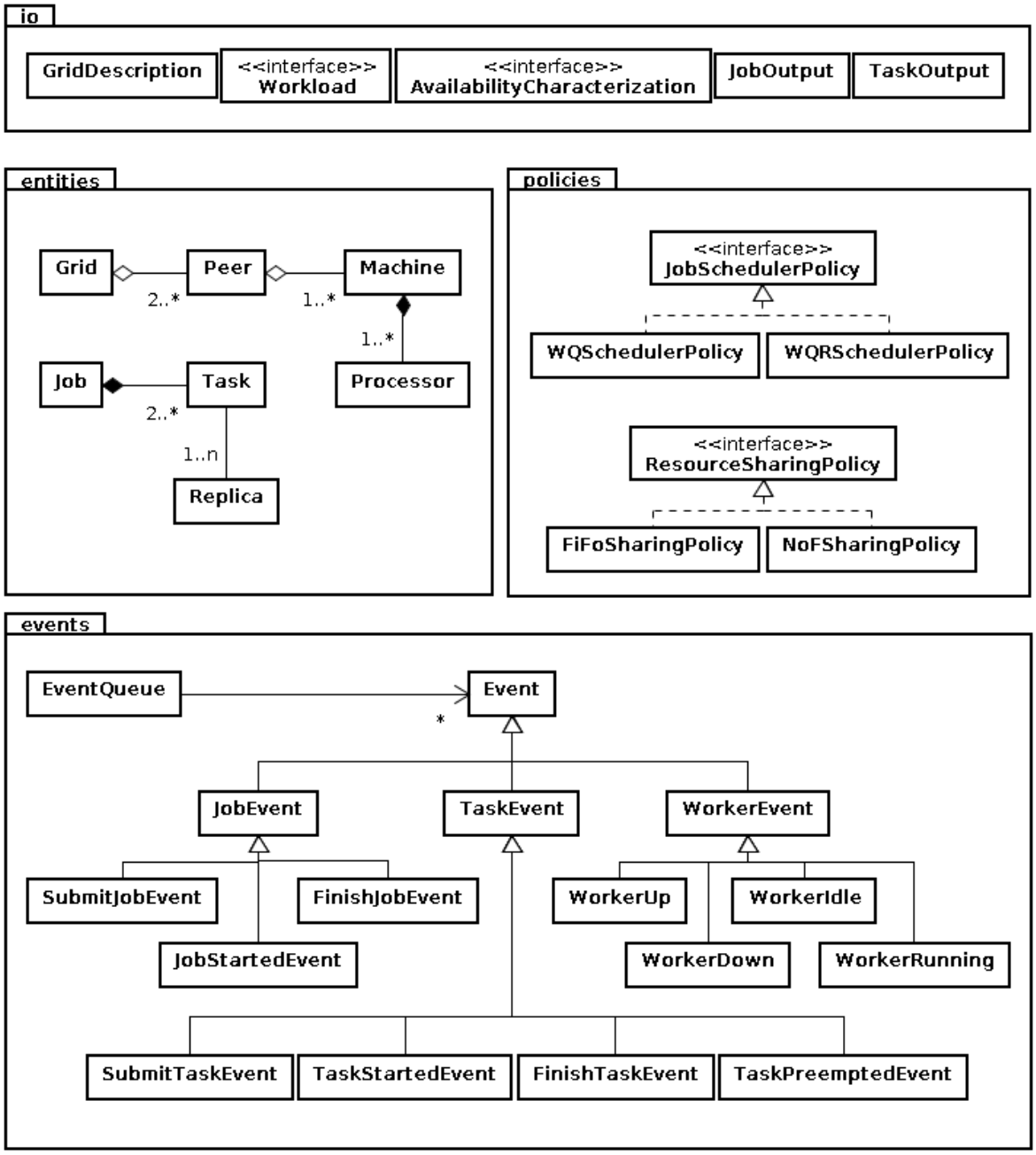


Figura 4.1: Visão Geral dos Principais Componentes do Simulador OurSim

pode ser observado no Algoritmo 5, que ilustra o ponto de entrada da simulação.

O Algoritmo 6 ilustra a estratégia de escalonamento utilizada. Na linha 2 pode-se observar o momento em que os *peers* são ordenados para a escolha do *site* alvo para executar uma dada tarefa. Como uma tarefa oriunda de um *peer* específico tem prioridade de execução nos recursos locais, o primeiro *peer* é sempre o *peer* de origem. Os demais são ordenados aleatoriamente. Já na linha 4 é verificado se um dada *peer* pode, em seus recursos, executar

Algoritmo 5 Iniciar(FilaDeEventos eq, Escalonador sc, Workload wl, Disponibilidade av)

```
1:  while not eq.isEmpty() do
2:      eq.add( av.nextEvents() )
3:      eq.add( wl.nextEvents() )
4:      tempoCorrente ⇐ eq.peek().getTime()
5:      while not eq.isEmpty() and eq.peek().getTime = tempoCorrente do
6:          proximoEvento ⇐ eq.poll()
7:          proximoEvento.doAction()
8:      end while
9:      sc.escalone(tarefas, peers, eq)
10: end while
```

uma dada tarefa. Caso haja recurso disponível e este atenda aos requisitos da tarefa, a resposta é positiva. Se a tarefa tem origem no próprio *peer* e há tarefas remotas executando nos recursos locais, estas serão preemptadas para priorizar a demanda local e a resposta também será positiva. Ainda, se a tarefa não for local mas vier de um *peer* que já colaborou com o *peer* alvo no passado, poderá haver preempção de tarefas remotas em benefício de um *peer* que tenha maior saldo de colaboração. Por fim, caso não haja recursos disponíveis e não seja possível uma preempção, a resposta será negativa e a tarefa não será alocada, permanecendo em fila e podendo ser alocada em escalonamentos futuros.

A implementação do modelo de simulação, o gerador de workload, os arquivos de entrada, os arquivos resultantes da simulação e os *scripts* para análise dos resultados estão empacotados e disponíveis no endereço eletrônico *http://redmine.lsd.ufcg.edu.br/projects/edigley*.

Algoritmo 6 escalone(Tarefas $tarefas$, Peers $peers$, FilaDeEventos eq)

```
1:  for all terefa in tarefas do
2:      pPeers ⇐ tarefas.getSourcePeer().prioritizePeers( peers )
3:      for all peer in pPeers do
4:          if peer.canExecute( tarefa ) then
5:              peer.execute( tarefa )
6:              tarefas.remove( tarefa )
7:              eq.addTaskStartedEvent( tarefa )
8:          end if
9:      end for
10: end for
```

Capítulo 5

Experimentos e Descrição das Métricas para Análise

No estudo apresentado neste trabalho são consideradas duas opções diferentes para a execução de *workloads* típicos da comunidade de *e-science*: grade oportunista entre pares e um ambiente de nuvem. A primeira opção se baseia no compartilhamento de recursos pertencentes aos diferentes participantes, formando uma federação de recursos, enquanto a segunda é baseada na existência de um mercado de computação utilitária, com provedores alugando seus recursos para satisfazer as demandas de clientes. Os parâmetros definidos para ambos os cenários são descritos ao longo deste capítulo.

5.1 Parâmetros de Simulação

5.1.1 *Workload* dos Usuários

Para representar o *workload* dos usuários foi utilizado o modelo de geração de *workloads* sintéticos para grades computacionais P2P proposto por Carvalho [19]. Neste modelo, o autor divide os atributos referentes ao *workload* nas categorias *comportamento dos usuários* e *características das aplicações*. A primeira envolve o intervalo entre submissão de dois *jobs* consecutivos pelo mesmo usuário, enquanto a segunda inclui o tempo de execução das tarefas e a quantidade de tarefas do *job*.

Trata-se de um modelo do tipo hierárquico, em que o *workload* referente à grade é ba-

seado no comportamento dos usuários que a compõem. Para a calibração do modelo, o autor utilizou *traces* de seis grades reais e identificou, a partir de cada um deles e por meio de agrupamento de usuários e de aplicações com características semelhantes, 5 perfis de usuários, sendo criado um modelo para cada perfil. Para cada perfil foram realizados ajustes de funções de distribuição de probabilidade aos dados históricos. Por meio desta metodologia, o modelo final do *workload* é obtido a partir da composição de modelos referentes aos diversos atributos citados anteriormente, capturando os diferentes perfis de usuários e de aplicações existentes nos *traces* que serviram para calibrar os modelos individuais.

Para a geração de um *workload* sintético a partir do modelo final, apenas dois parâmetros são informados: a quantidade de *peers* e o número de usuários por *peer*. Já o perfil de submissão de cada usuário é escolhido aleatoriamente dentre os 5 perfis identificados. Nas simulações realizadas neste trabalho, o número de usuários por *peer* foi fixado em 10, enquanto para a quantidade de *peers* é realizada uma varredura de parâmetros (detalhada na Seção 5.1.2).

A escolha de 10 usuários por *peer* advém da tentativa de relacionar a quantidade de usuários por *peer* e a quantidade de recursos compartilhados por *peer*. Inicialmente, imaginou-se uma relação de igualdade, isto é, a quantidade de usuários sendo a mesma da quantidade de recursos compartilhados, no entanto, para cenários com poucos *peers* (10, 20 e 30) e com a quantidade fixa de 30 máquinas por *peer*, a execução do *workload* gerado se tornava impraticável em tão poucas máquinas. Com a relação de 3 máquinas para cada usuário se obtém cenários com um nível considerável de contenção, mas propício de ser executado em grades com poucos *peers*. Como na prática nem todos os usuários locais das máquinas compartilhadas são usuários ativos da grade, a relação utilizada não se distancia da realidade.

Por fim, vale salientar que os resultados obtidos nesta dissertação estão intimamente relacionados com o tipo de *workload* utilizado e o interesse em gerar *workloads* realistas para grades P2P presente no trabalho realizado por Carvalho [19] vai ao encontro do objetivo do presente trabalho.

5.1.2 Configuração dos Sistemas

Configuração da Grade entre Pares

Com o objetivo de analisar cenários envolvendo grades de diferentes tamanhos, uma forma de variar a contenção do sistema, foi realizada uma varredura de parâmetros para $|G|$, com início em 10, término em 150 e com passo de 10 unidades.

Nas simulações, considera-se que cada *peer* possui e compartilha 30 máquinas com a grade. Esta quantidade foi escolhida por representar aproximadamente a média de máquinas compartilhadas pelos *peers* na comunidade OurGrid (*http://status.ourgrid.org*).

Para definir a distribuição estatística da capacidade computacional das máquinas foi realizada uma análise em dois *sites* representativos da comunidade OurGrid (*peers lsd.ufcg.edu.br* e *gmf.ufcg.edu.br*). A Tabela 5.1 apresenta estatísticas básicas referentes aos dados coletados. Em conjunto, esses *sites* agregam cerca de 80 máquinas heterogêneas do tipo *desktop*, utilizadas primariamente por alunos de graduação e pós-graduação, além de funcionários dos laboratórios.

Tabela 5.1: Estatísticas da Capacidade Computacional de dois *Sites* OurGrid

Site	**Máquinas**	**Capacidade (em GHz por núcleo)**				
		Mín.	**Máx.**	**Mediana**	**Média**	**Desvio Padrão**
lsd	67	0,80	3,0	2,80	2,55	0,500
gmf	18	1,00	3,0	2,37	2,22	0,520
ambos	85	0,80	3,0	2,40	2,48	0,518

Por sua vez, a Figura 5.1 apresenta a distribuição da capacidade computacional das máquinas, em que se pode perceber a existência de algumas poucas máquinas mais antigas, com baixa capacidade, e a maioria das máquinas com capacidade superior a 2.0 GHz, para ambos os *sites*. Vale destacar a inexistência de uma norma minimamente centralizada no caso do *site lsd*.

Os recursos de outros *sites* pertencentes à comunidade OurGrid possuem capacidades praticamente idênticas, pois são provenientes de salas de aula, em que as máquinas são todas adquiridas em um mesmo lote e são utilizadas, em geral, até o fim de suas vidas úteis, momento em que um novo lote é adquirido.

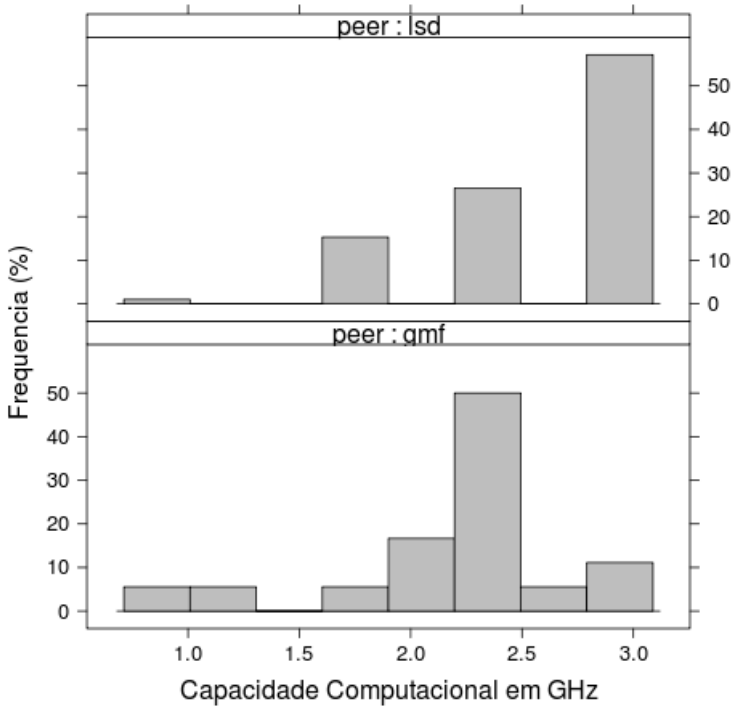


Figura 5.1: Velocidade das Máquinas de dois Sites OurGrid

A título de comparação, a Figura 5.2 apresenta a distribuição da capacidade computacional normalizada de duas grades de *desktops* (*lri* e *sdsc*), reportadas por Kondo et al. [41], além dos *sites lsd* e *gmf*. Nos histogramas se pode observar que as distribuições são diferem bastante entre os vários domínios administrativos, até mesmo com a grade *lri* apresentando um padrão parecido com o do *site lsd*.

A despeito das diferenças entre as distribuições de capacidade nos diferentes domínios, caso sejam observados todos os recursos em conjunto, a distribuição apresenta o padrão representado na Figura 5.3, que lembra uma distribuição normal, tendendo a confirmar, nas grades entre pares, resultados semelhantes aos de grades de computação voluntária [6].

Como a grade entre pares é formada pela agregação de recursos oriundos de diferentes domínios administrativos, e com base nas constatações anteriores, resolveu-se modelar a capacidade computacional dos recursos da grade entre pares como uma distribuição normal, com média de 2 *ECUs* ($\approx$ $2,4$ GHz) e desvio padrão de 0,5. A Figura 5.4 apresenta a comparação entre as distribuições acumuladas empíricas dos dois *sites* analisados (isoladamente e com os recursos agregados) e a distribuição normal.

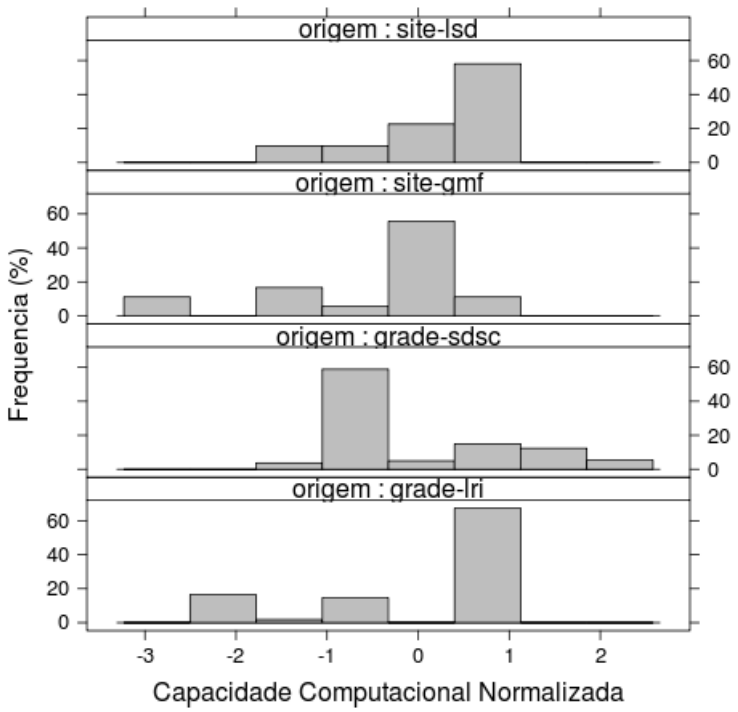


Figura 5.2: Velocidade Normalizada das Máquinas de Diferentes Grades de *Desktops*

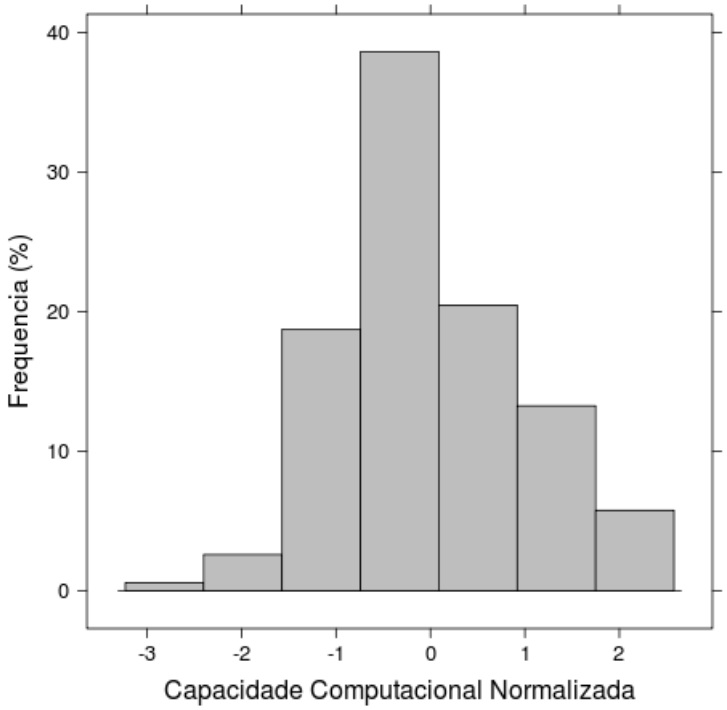


Figura 5.3: Velocidade Normalizada das Máquinas de Todos os Domínios em Conjunto

Embora os dados referentes aos dois *sites* não sigam uma distribuição normal (não passam em um teste de normalidade com nível de confiança de 90%), o uso de uma distribuição conhecida permite uma maior independência entre os resultados obtidos e a capacidade particular dos dois *sites* analisados. Destaca-se, ainda, que os parâmetros da distribuição normal foram escolhidos de forma a se aproximarem das distribuições empíricas (vide Tabela 5.1).

Vale ressaltar que não são utilizados valores resultantes da distribuição $N(2,4;\ 0,5)$ inferiores a 0,8 ou superiores a 3,0. No primeiro caso, porque é a capacidade mínima identificada nos *sites*, enquanto no segundo porque, na prática, os fabricantes de processadores não estão mais concentrando esforços em aumentar os ciclos por núcleo dos processadores (devido a problemas de sobreaquecimento), e sim na quantidade de núcleos por pastilha. Por fim, apenas por simplicidade na caracterização, nas simulações realizadas as máquinas da grade são todas consideradas como monoprocessadas.

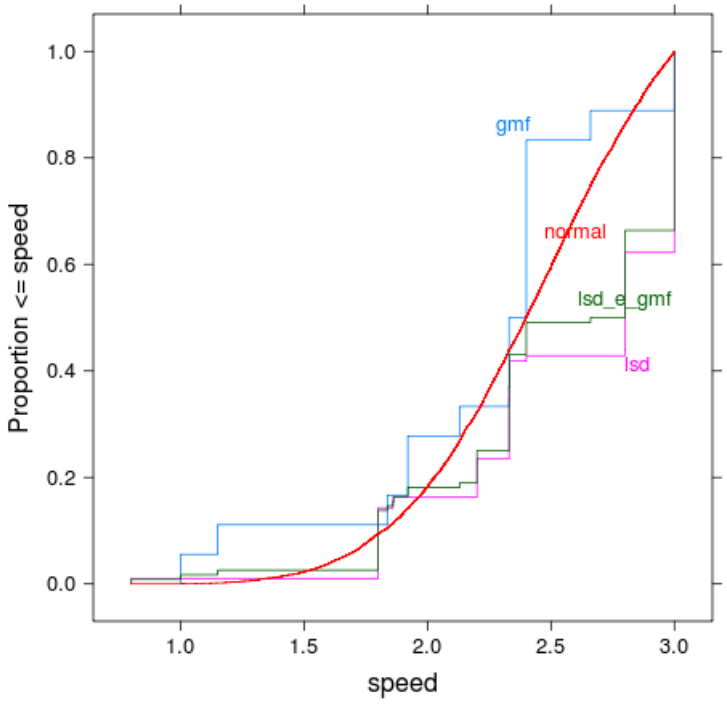


Figura 5.4: Comparação da Distribuição dos dois *Sites* OurGrid e a Distribuição Normal

Configuração da Nuvem de Instâncias *Spot*

Para o ambiente de nuvem de instâncias *spot*, foi considerado um limite $L = 100$, o mesmo praticado pelo serviço *Amazon EC2* [28]. Para a oscilação dos *spot-prices* (Π) foram utilizados os históricos de preço informados pelo *AWS* [14] e também disponibilizados publicamente no sítio *web http://cloudexchange.org* [58]. O histórico de preços constrói toda a série temporal O, sendo uma para cada tipo de instância presente na Tabela 2.1. Foram utilizados os históricos referentes à região leste dos Estados Unidos, identificada no *AWS* como *US - N. Virginia*. Também são utilizadas apenas as instâncias com sistema operacional *Linux*, que possuem preços mais acessíveis.

Os *traces* com os históricos dos *spot-prices* utilizados abrangem um pouco mais de um ano (desde dezembro de 2009 a fevereiro de 2011). Como as simulações realizadas se referem a um *workload* cujas submissões de *jobs* espalham-se por 7 dias, a cada execução é escolhido aleatoriamente um instante entre o início do trace e 15 dias antes do final e este instante é utilizado como ponto inicial da simulação. Como são realizadas diversas execuções para cada cenário e é calculado um intervalo de confiança, os valores obtidos tendem a não refletir anomalias sazonais presentes nos *traces*.

Como usuários da nuvem Υ, aos quais é aplicado o limite L, são extraídos todos os usuários que submetem algum *job* para o sistema ($J \in W \Rightarrow user(J) \in \Upsilon$). Como o objetivo do presente trabalho é valorar a grade, foi considerado que os lances presentes em B são altos o suficiente para superar o *spot-price* atual (momento de submissão) e de todos os futuros *spot-prices*, isto é, até que a última tarefa do *job* seja concluída.

Em um cenário real, em que há restrição orçamentária, a abordagem mais prudente é, aliado ao uso de *checkpointing* na nuvem [65], o uso de um lance persistente, que fixa um valor máximo que o usuário está disposto a pagar e, caso o *spot-price* supere o lance, este ainda continuará ativo nas futuras oscilações, podendo originar outra alocação. O processo continua até que a execução de toda a demanda do usuário se complete ou o lance seja explicitamente abortado pelo usuário. O ponto negativo desta abordagem é que o *makespan* da aplicação pode se tornar extremamente longo.

5.1.3 Padrão de Disponibilidade das Máquinas da Grade

Como as grades em questão são oportunistas, os modelos de simulação precisam considerar a volatilidade dos recursos de forma a manter fidelidade com os sistemas reais. Há alguns trabalhos que analisaram a volatilidade de grades oportunistas [40; 41; 8; 18; 17]. Embora tais resultados pudessem ser utilizados, preferiu-se realizar uma análise do padrão de disponibilidade das máquinas da comunidade OurGrid, visto que as análises citadas anteriormente não foram feitas considerando grades entre pares.

Foram analisados dados de disponibilidade referentes a um período de 4 meses (de outubro de 2009 a janeiro de 2010) originados de 4 *sites* (englobando em torno de 150 máquinas). O objetivo da análise foi encontrar uma função de distribuição de probabilidade que representasse os dados referentes à duração dos intervalos de disponibilidade e de indisponibilidade.

Inicialmente foram escolhidas as distribuições *exponencial*, *normal*, *log-normal* e *Weibull*, por serem comumente utilizadas para tal propósito e apresentarem baixa complexidade. Os valores dos parâmetros das distribuições foram estimados com base no método Estimação por Máxima Verossimilhança (mais conhecido como *MLE*, abreviação de *Maximum-Likelihood Estimation*) [30]. Para determinar qual das distribuições candidatas melhor se adequava aos dados, foi realizado o teste de *Goodness-of-Fit* (GoF) baseado no método *Kolmogorov-Smirnov* (*KS-test*) [43] para um nível de significância $\alpha = 0.05$. Considerando o parâmetro *p-value* para o teste *Kolmogorov-Smirnov*, as distribuições que apresentaram os melhores resultados foram as log-normais. Os resultados para todas as distribuições analisadas estão nas Tabelas 5.2(a) e 5.2(b).

5.2 Métricas de Comparação

Em cada simulação são registrados os *makespans* de cada *job* tanto para a execução na grade quanto na nuvem. Além do *makespan*, no caso da nuvem, também é computado o custo de execução de cada *job*.

Tabela 5.2: Padrão de Disponibilidade e Indisponibilidade na Comunidade OurGrid

(a) Disponibilidade

Distribuição	***p-value***	***param-1***	***param-2***
normal	0.00008	0.0002	90071.08
log-normal	0.5	7.95	2.12
Weibull	0.4	0.4	7832.76
exponencial	0.001	0.00004	-

(b) Indisponibilidade

Distribuição	***p-value***	***param-1***	***param-2***
normal	0.0032	0.0026	4620.40
log-normal	0.1119	7.2409	1.04
Weibull	0.0440	0.8776	2387.48
exponencial	0.0785	0.0003	-

5.2.1 Métricas de Desempenho

Como os *peers* participantes da grade possuem, cada um, *workloads* heterogêneos, assim como os *jobs* são bem diferentes um do outro (um pode ter 10 tarefas enquanto um outro 1.000, por exemplo), se faz necessária uma métrica de desempenho que normalize a variação de demanda entre diferentes *peers* e *jobs* de forma que seja possível comparar os desempenhos obtidos nas diferentes plataformas.

Para tanto, é utilizada a razão entre a agregação dos *makespans* nas duas plataformas, denominada D^s e calculada para cada *peer* s, como pode ser visto na Equação 5.1.

$$D^s = \frac{\sum_{j=1}^{|W^s|} makespan^C(J_j^s)}{\sum_{j=1}^{|W^s|} makespan^G(J_j^s)} \tag{5.1}$$

Também é calculada a média para esta métrica ($\overline{D}$), considerando todos os nós do sistema, de tal forma a representar o bem-estar geral que a grade oferece a seus participantes:

$$\overline{D} = \frac{1}{|S|} \sum_{s=1}^{|S|} D^s \tag{5.2}$$

Ambas as métricas (D^s e $\overline{D}$), tratadas de forma indistinta simplesmente como D, flutuam

em torno de uma linha de indiferença de desempenho ($y = 1$, em um plano cartesiano no qual o eixo-x representa, por exemplo, o tamanho da grade). A métrica D deve sempre ser interpretada considerando essa linha de indiferença, o que pode resultar em três casos que guiam a decisão a respeito de qual plataforma fornece, no geral, o melhor desempenho. No primeiro, situação em que $D < 1$, a nuvem se mostra como a plataforma de melhor desempenho. No segundo caso, para $D = 1$, as duas plataformas apresentam desempenhos semelhantes. Por fim, quando $D > 1$ o desempenho na grade supera o obtido na nuvem.

Quando a métrica D se situa na linha de indiferença, outro critério deve ser utilizado para o desempate. Neste caso, o critério mais racional a ser utilizado é o fator custo, cujas métricas estão descritas na próxima seção.

5.2.2 Métricas de Custo e de Valor

Custo Médio por Tarefa Utiliza-se um custo médio por tarefa (CMT), calculado pela fórmula da Equação 5.3, com o objetivo de comparar o custo quando se executa o mesmo *workload* em diferentes tipos de instâncias *spot*. Essa métrica também pode ser utilizada como critério de desempate quando se atinge o ponto de indiferença mencionado na seção anterior.

$$CMT = \frac{1}{\sum_{j=1}^{|W|} |J_j|} \cdot \sum_{j=1}^{|W|} \sum_{i=1}^{|J_j|} cost(J_j, i) \tag{5.3}$$

Valor da Grade Conforme adiantado no capítulo introdutório, estimar o valor fornecido por uma grade entre pares não é algo trivial. Em nossa abordagem, um mesmo *workload* é executado em duas plataformas de execução distintas, uma delas sendo uma grade entre pares e a outra uma infraestrutura de nuvem de instâncias. Enquanto na primeira, normalmente, o desempenho é o aspecto de maior interesse para o usuário, na última, além do desempenho fornecido, o custo é uma preocupação prevalecente. Portanto, esses dois aspectos, desempenho e custo, serão utilizados para a estimativa do valor da grade, com base, primeiramente, na ideia intuitiva de que quanto menor o *makespan* obtido em uma plataforma maior será o valor desta plataforma para o usuário, ou seja, o valor é inversamente proporcional ao *makespan*.

Como o objetivo é encontrar uma estimativa monetária para o valor da grade, pode-se recorrer ao aspecto custo na nuvem, visto que esse custo se refere à execução do mesmo *workload* executado na grade. A relação de proporcionalidade entre o valor da grade e o custo de execução na nuvem é, intuitivamente, direta.

Por último, resta ponderar o papel desempenhado pelo *makespan* obtido na nuvem para essa estimativa de valor da grade. Caso os *makespans* na nuvem e na grade fossem idênticos, bastaria mapear diretamente o valor da grade para o custo de execução na nuvem. Como não é esse o caso, é suficiente perceber que, para um *makespan* na grade fixo, quanto maior o *makespan* obtido na nuvem maior será o valor da grade ou, de forma genérica, o valor da grade depende da relação entre os *makespans* obtidos na grade e na nuvem.

Partindo de todas as considerações feitas anteriormente, para estimar o valor da grade V_g será utilizada a relação expressa na Equação 5.4, definida a seguir:

$$V_g = \frac{M_c}{M_g} \cdot C_c \tag{5.4}$$

Pela referida relação, M_c representa a soma dos *makespans* de todos os *jobs* pertencentes a um *workload* W quando executado, na nuvem, em instâncias de um dado tipo, enquanto C_c denota o custo total de tal execução. Ainda, M_g se refere à soma dos *makespans* dos mesmos *jobs* quando executados na infraestrutura de grade.

Custo Extra de Manutenção da Grade Opitz et al. fizeram um levantamento dos custos envolvidos na computação em grade, inclusive para grades oportunistas, agrupando-os em cinco categorias: *hardware*, *software*, eletricidade, pessoal e comunicação de dados [45].

Em uma grade como a comunidade OurGrid, os recursos computacionais não são adquiridos especificamente para a grade. Na verdade, os *desktops* já existem nas instituições, quer seja em laboratórios que servem como salas de aula ou nas mesas de trabalho de funcionários, de alunos de graduação, pós-graduação ou de pesquisadores. Ao ingressarem na grade, as máquinas serão utilizadas de forma oportunista e não intrusiva. Portanto, o custo de aquisição de novo *hardware* é inexistente. Os executáveis utilizados nas aplicações *BoT*, o *middleware* de grade e ambientes de execução (*R, Java, Gnuplot, Octave, etc.*) são, em sua maioria, *software* livre, de modo que não há custos com licenças.

As instituições participantes também já possuem uma conexão de *Internet* de banda larga,

utilizada originalmente para as necessidades da própria instituição. Isto permite, pelo menos para aplicações de computação intensiva, desconsiderar a necessidade de adquirir um canal dedicado unicamente para a grade.

Devido à abordagem de melhor-esforço e à característica descentralizada dos sistemas entre pares, um *peer* não incorre na obrigação de manter o seu *site* em regime 24/7. Na ocorrência de uma eventual falha de serviço que necessite de intervenção manual, o próprio administrador de sistemas ou algum usuário avançado, caso exista, se responsabiliza por encaminhar uma solução. Desta forma, para o escopo deste trabalho, não serão contabilizados custos adicionais com pessoal.

Deste modo, sobram os gastos adicionais com eletricidade, considerando o uso de *desktops* convencionais, do tipo *commodities*. Para uma análise que se aproxime minimamente da realidade, dois valores diferentes de consumo devem ser considerados: um para os períodos de completa ociosidade e outro para operação em carga máxima. A Tabela 5.3 apresenta valores referentes a vários sistemas quando submetidos à execução do SETI@home. Pelos dados, nota-se que o consumo de energia, mesmo em estado de completa ociosidade, não é desprezível.

Tabela 5.3: Consumo de Energia (em Watts) do SETI@home em Diferentes Sistemas [35]

Processador	***SETI@home*** - c^t	**Ocioso** - c^o	**Sobrecarga** - c^s
Athlon 64 2800+	115	77	38
Athlon XP 2500+	138	84	54
Athlon XP 3200+	154	87	67
Athlon 64 3400+	150	94	56
Athlon 64 FX-53	170	107	63
Pentium 4 2,40C GHz	144	86	58
Pentium 4 3,40C GHz	178	92	86
Pentium 4 EE 3,40 GHz	180	93	87
Pentium 4 2,80 GHz	179	116	63
Pentium 4 3,40E GHz	198	120	78

Como no tipo de grade em questão os recursos são utilizados de forma oportunista, devem ser considerados dois cenários: quando o recurso está sendo utilizado apenas pela grade

(nos períodos noturnos, por exemplo) e quando se utiliza o recurso nos pequenos intervalos em que o usuário local deixa temporariamente de utilizar sua máquina. No primeiro caso, todo o consumo deve ser creditado à grade, enquanto no segundo apenas a diferença entre os consumos em carga total e em ociosidade devem ser considerados (supondo que o sistema não seria desligado nesses intervalos). Assim, para realizar a estimativa de custo extra introduzido pela grade, basta multiplicar o consumo estimado pelo preço de 1 *kWh* (*kilowatt-hora*).

Após todas as considerações, chega-se ao modelo simplificado $CDPM(S, m)$ para o cálculo do custo diário por máquina, descrito pela Equação 5.5, em que $cons^o(S, m)$, $cons^s(S, m)$ e $cons^t(S, m)$ referem-se, respectivamente, aos consumos quando em ociosidade, em sobrecarga e total (vide Tabela 5.3) da m-ésima máquina do *site* S, k é o preço de 1 *kWh*, r é a taxa de ociosidade das máquinas em horário normal de trabalho, representado por h horas por dia, e u é a utilização do sistema, isto é, o percentual de tempo em que a máquina é efetivamente utilizada para processar tarefas da grade.

$$\begin{aligned} CDPM(S,m) = \frac{k}{1000} \cdot \{ \quad & h \cdot r \quad \cdot u \cdot cons^s(S,m) + \\ & (24-h) \cdot \left[u \cdot cons^t(S,m) + (1-u) \cdot cons^o(S,m) \right] \} \end{aligned} \tag{5.5}$$

Partindo do modelo de custo diário por máquina, a fórmula para estimar o custo de manutenção da grade é trivial: basta somar os custos de todas as máquinas individuais presentes na grade e multiplicar pela quantidade de dias em operação. Denotando por d um intervalo de tempo em número de dias, a Equação 5.6 apresenta a referida fórmula.

$$C_g = d \cdot \sum_{s=1}^{|G|} \sum_{m=1}^{|S_s|} CDPM(S_s, m) \tag{5.6}$$

Para o caso de máquinas e sites homogêneos, considerando np o número de *peers* e nm o número de máquinas por *peer*, o modelo anterior pode ser escrito simplesmente como:

$$C_g = d \cdot np \cdot nm \cdot CDPM \tag{5.7}$$

Em que $CDPM$ é o custo diário de cada máquina homogênea, para qual c^s, c^t e c^o denotam, respectivamente, os consumos de sobrecarga, total e de ociosidade da máquina homogênea, permitindo a escrita da Equação 5.5 da seguinte forma:

$$CDPM = \frac{k}{1000} \cdot \left\{ h \cdot r \cdot u \cdot c^s + (24 - h) \cdot \left[u \cdot c^t + (1 - u) \cdot c^o \right] \right\} \tag{5.8}$$

Na prática, os sistemas computacionais, quer sejam os *desktops* individuais ou a grade em si, tendem a utilizar alguma estratégia de economia de energia, atualmente sob a denominação de *computação verde* (do inglês *green computing*), para reduzir o desperdício de energia quando o sistema está em ociosidade.

Ponciano et al. investigaram o impacto gerado pelo uso de estratégias de computação verde em grades entre pares e concluíram que o uso do estado de sobreaviso (*standby*) é a melhor estratégia a ser adotada [51] [53]. Considerando as particularidades da grade entre pares, foi identificado que o uso de *standby* gera um impacto desprezível no *makespan* dos *jobs* e permite uma economia substancial de energia nos períodos de ociosidade.

A estratégia de *standby* consiste em salvar todo o estado do sistema na memória principal e desligar os componentes que tipicamente são responsáveis pela maior parte do consumo presente no estado de ociosidade c^o, que são o processador e o disco rígido. A memória principal se mantém energizada e, na ocorrência de um evento de atividade, como o retorno do usuário local ou a chegada uma tarefa para execução, os componentes voltam a ser energizados e o estado do sistema é restaurado. A transição de estado ocioso-*standby* e de *standby*-ativo demora, em média, apenas 2,5 segundos e o estado é responsável por um consumo de energia equivalente a 3,33 *Watts* [51].

Para uma grade que utiliza uma estratégia de economia de energia, a equação para o cálculo do custo diário por máquina deve ser reescrita de forma a considerar também o consumo de energia no estado de *standby*. Antes de efetivamente entrar em *standby*, é necessário que decorra um tempo de inatividade, configurado pelo usuário ou administrador e normalmente definido em 5, 10 ou 15 minutos. Por simplicidade, será considerado um tempo de inatividade de 0 segundos, isto é, a passagem para o modo de economia de energia se daria de forma imediata. Ressaltando que os cálculos realizados são apenas uma estimativa, imagina-se que tal simplificação não compromete a validade do resultado final.

Desta forma, denotando por c^v o consumo de energia usando a estratégia de *standby*, a Equação 5.5 ficaria:

$$CDPM^v = \frac{k}{1000} \cdot \left\{ h \cdot r \cdot u \cdot (c^t - c^v) + (24 - h) \cdot \left[u \cdot c^t + (1 - u) \cdot c^v \right] \right\} \tag{5.9}$$

Capítulo 6

Resultados e Discussão

Para cada combinação da variação dos fatores ($|G|$ e tipo de instância *spot*) foram realizadas 10 execuções simuladas, de acordo com a configuração estatística apresentada na Seção 5, e os valores contidos nos gráficos estão em um intervalo de confiança de 95%.

Considerando as características dos dois sistemas sob estudo, e na inexistência do limite $L = 100$ na nuvem, e ainda supondo, apenas por ilustração, que as capacidades computacionais das instâncias na nuvem e na grade fossem idênticas (e com apenas um processador), o desempenho obtido na nuvem seria sempre superior, visto que na nuvem as máquinas utilizadas se comportam como dedicadas, enquanto na grade os recursos são utilizados de forma oportunista e, consequentemente, as execuções estão susceptíveis a preempções. No entanto, como o limite $L = 100$ existe, a capacidade máxima de paralelização que um único usuário consegue obter é de 100 tarefas simultâneas. Caso o número de tarefas na aplicação do usuário seja maior que 100, inevitavelmente haverá espera em fila e, consequentemente, um aumento no *makespan* da aplicação.

Na grade entre pares, por sua vez, a quantidade de recursos disponíveis depende primordialmente da quantidade de *peers* na grade, permitindo que, em alguns momentos, sejam obtidos níveis de paralelização superiores ao possível na nuvem. Para uma aplicação *BoT* na grade, o que vai definir a quantidade de recursos para si será, além da volatilidade dos recursos, a contenção do sistema, isto é, quantas outras aplicações também estarão disputando recursos entre o início e o término daquela primeira.

Como os diferentes tipos de instâncias na nuvem possuem capacidades computacionais diferentes (seja em velocidade por núcleo ou quantidade de núcleos por instância), além de

preços variados, torna-se necessário identificar instâncias que sirvam como parâmetro para a comparação com a grade.

As próximas seções identificam quais as instâncias mais adequadas para a comparação com a grade (considerando o *trade-off* entre *makespan* e custo de execução) e calculam o valor estimado para grades de diferentes tamanhos, de acordo com os critérios definidos no Capítulo 5.

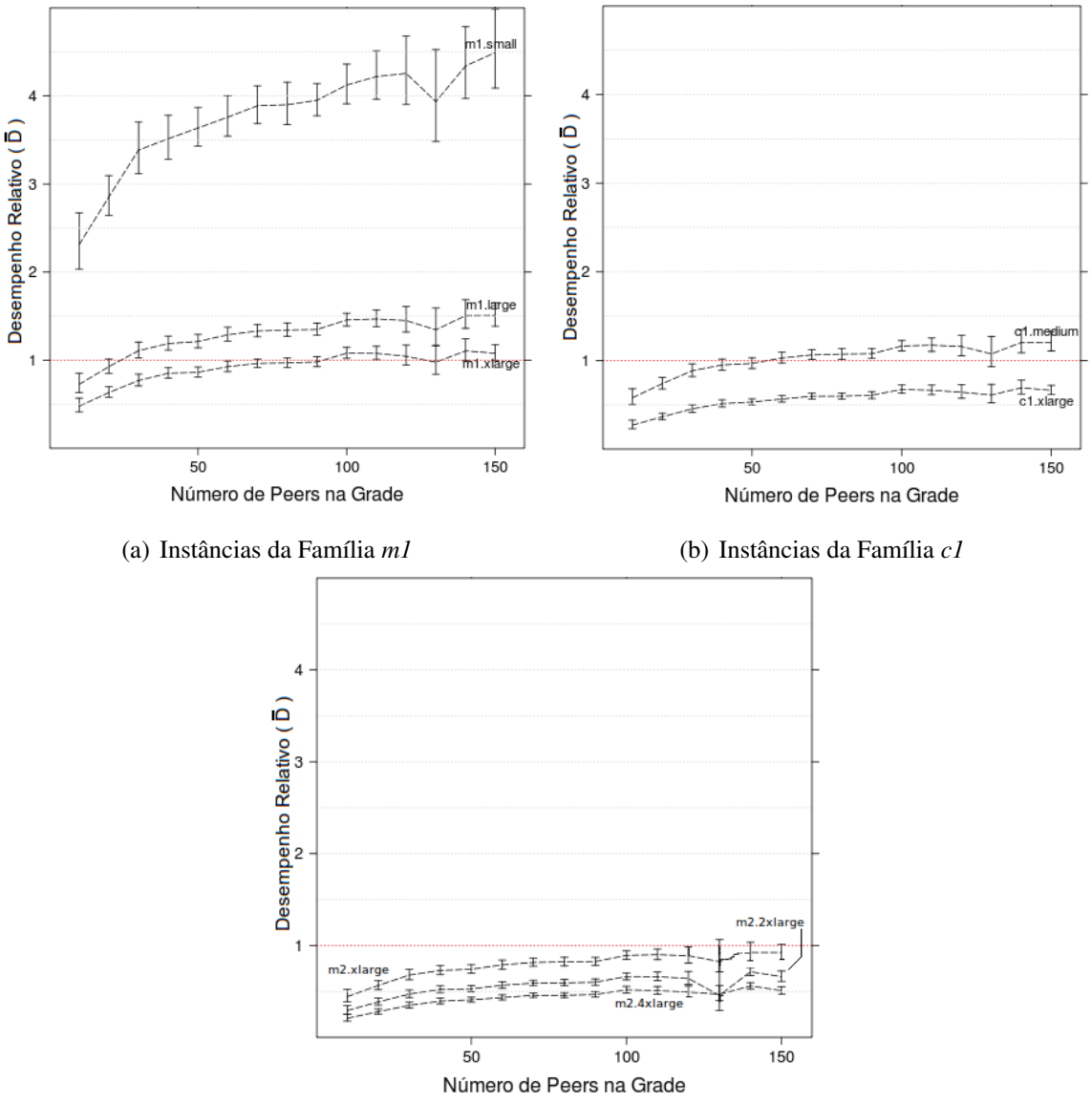
6.1 Instâncias Adequadas para a Comparação

A Figura 6.1 apresenta os resultados da execução de um mesmo *workload* tanto em uma grade entre pares quanto nas instâncias das diferentes famílias na nuvem *spot*, enquanto a Figura 6.2 seleciona três destas instâncias com bom desempenho para uma melhor visualização. Primeiramente, como esperado, nota-se que as instâncias *spot* com maior capacidade computacional apresentam um desempenho superior ao obtido na grade (posicionam-se abaixo da linha $y = 1$).

Pelo gráfico, observa-se que, ao passo que a quantidade de *peers* na grade aumenta, o desempenho comparativo da grade em relação a todas as instâncias *spot* melhora consideravelmente até chegar aproximadamente à quantidade de 100 *peers*, com destaque para o crescimento acentuado na faixa entre 10 e 40 *sites*. A partir daí, a métrica tende a uma estabilização, observando-se um aumento dos intervalos de confiança, o que sugere uma maior aleatoriedade nos resultados que formam os referidos intervalos.

O provável motivo de tal comportamento está na saturação da grade para a característica do *workload*, isto é, a partir de uma certa quantidade de *peers* a adição de mais *peers* não gera uma maior satisfação para os *peers* já existentes, visto que os novos *peers* também irão demandar recursos e a quantidade de máquinas que cada *peer* adiciona à grade (apenas 30) não é suficiente para compensar a maior disputa gerada entre todos os pares. É importante observar que, conforme descrito no Capítulo 4, quando surgem novos *jobs* de um determinado *peer* todos os seus recursos locais serão alocados para as tarefas referentes a estes *jobs* e, caso haja tarefas de *peers* remotos executando, estas serão preemptadas, o que potencialmente aumenta os *makespans* de seus respectivos *jobs*.

Ao se entrar no mérito da comparação dos desempenhos, observa-se um distanciamento



(a) Instâncias da Família *m1*

(b) Instâncias da Família *c1*

(c) Instâncias da Família *m2*

Figura 6.1: Comparação de Desempenho entre a Grade entre Pares e as Instâncias *Spot*

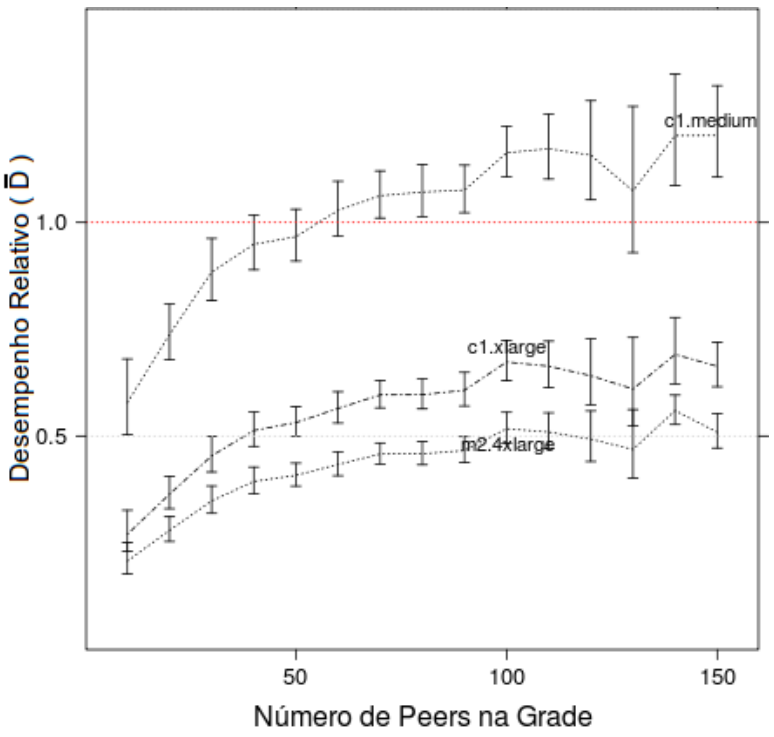


Figura 6.2: Comparação com as Instâncias *c1.medium*, *c1.xlarge* e *m2.4xlarge*

considerável da instância *m1.small* em relação à linha de indiferença ($y = 1$). A explicação para o desempenho desfavorável apresentado pela instância *m1.small* está na baixa capacidade computacional de tais instâncias quando comparadas com as máquinas *desktops* típicas da grade (Figura 5.1), visto que contam com apenas 1 *ECU*, conforme pode ser visto na Tabela 2.1. Além disto, estas contam com apenas 1 núcleo, o que tende a aumentar a espera em fila pela liberação de instâncias.

Este resultado nos indica que, ao contrário das abordagens seguidas por Kondo et al. [39] e Andrzejak et al. [9], se for para avaliação de desempenho comparativo, diante das outras opções existentes, a instância *m1.small* não deve ser considerada para a execução de aplicações *BoT*. Nos referidos trabalhos, a instância *m1.small* é utilizada como referência para avaliar o custo/benefício da computação na nuvem em relação à computação voluntária e também para a tomada de decisão relacionada ao escalonamento de aplicações de computação intensiva na nuvem. Mais adiante se conclui que também que a *m1.small* não é uma instância adequada quando se considera o fator custo.

No outro extremo, o bom resultado apresentado pela instância *m2.4xlarge* se explica, obviamente, por sua maior capacidade computacional (3.25 *ECU*) e, de forma mais importante,

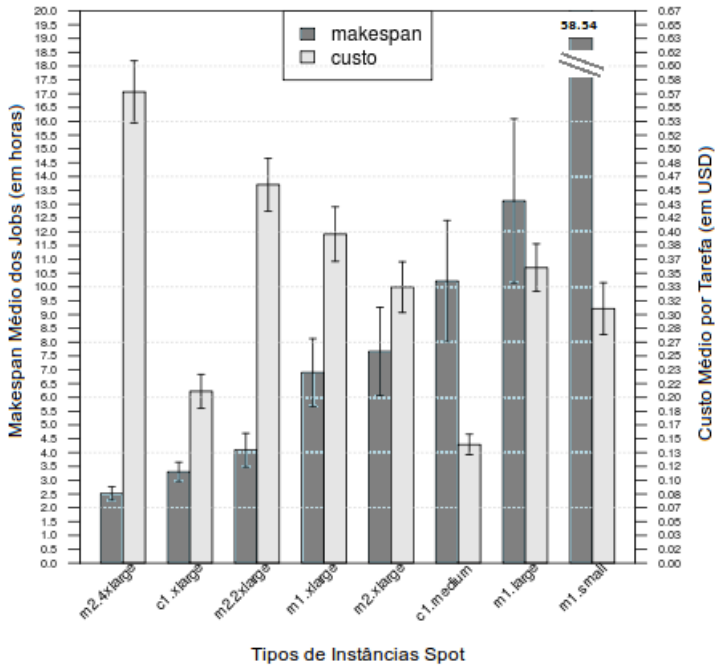
pelos 8 núcleos disponíveis por instância, o que diminui o efeito negativo do limite L de 100 máquinas, visto que, conforme descrito no Capítulo 5, é alocada uma tarefa por núcleo.

Ao considerar o fator custo, é apresentada na Figura 6.3(a) a contrapartida financeira que o usuário precisa realizar para obter o desempenho apresentado na Figura 6.2. De imediato, observa-se que a instância *m1.small* não é uma boa opção, visto que, embora apresente um custo por tarefa não muito alto, apresenta um desempenho sofrível. Já a instância *m2.4xlarge*, como visto anteriormente, apresenta o melhor desempenho, no entanto é a solução mais cara. Quanto ao fator custo, destacam-se as instâncias *c1.medium* e *c1.xlarge*, respectivamente com o menor e o segundo menor custo de execução por tarefa.

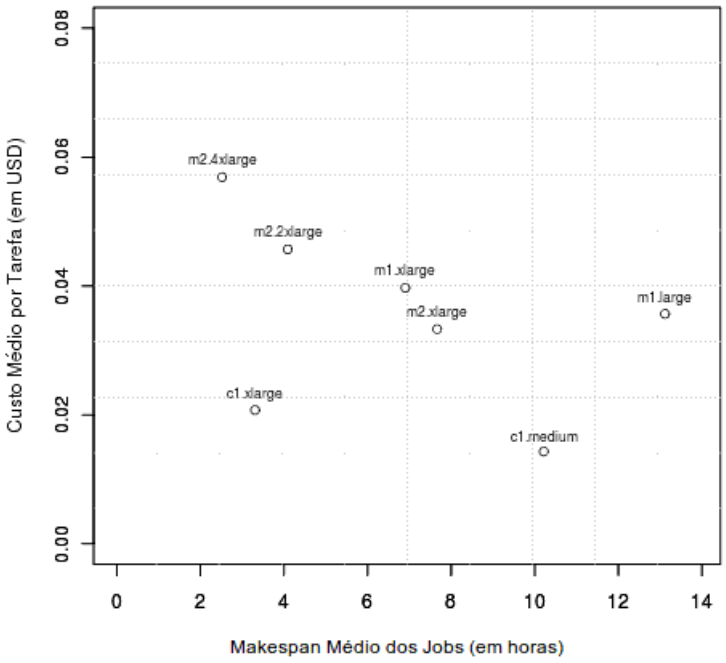
Para uma análise um pouco mais criteriosa de qual, de fato, é a instância que oferece o melhor custo/benefício, é necessário considerar o *trade-off* existente entre o *makespan* e o custo associado. A Figura 6.3(b) reflete visualmente como as instâncias se comportam neste aspecto (não considerando preferência por um fator em particular). Idealmente, o usuário deseja pagar o mínimo pelo melhor desempenho possível para a sua aplicação. Desta forma, quão mais à esquerda e mais para baixo do gráfico estiver a instância, melhor. Visualmente se nota a instância *c1.xlarge* isolada como a opção que oferece o melhor custo/benefício para o usuário (a instância *m1.small* não é exibida pois se posiciona aproximadamente no ponto cartesiano $(58, 0.3)$.

Embora tudo indique a *c1.xlarge* como a instância adequada, não se pode relevar o fato de a *c1.medium* ser quase um terço mais barata que a *c1.xlarge*. Como discutido no capítulo introdutório, a receita financeira resultante da execução de aplicações de *e-science* é comumente indefinida, de forma que, muitas vezes, a preocupação primeira é com o custo, e não com o melhor desempenho em si. Ainda assim, o desempenho oferecido precisa ser bom o suficiente para não inviabilizar o trabalho, isto é, o *makespan* obtido não pode ser excessivamente longo. Desta forma, deve-se utilizar alguma ponderação entre os fatores *makespan* e custo para realizar a escolha da instância adequada para a comparação.

Seja pc a importância que o usuário confere ao custo da execução. pc pode assumir valores entre 0 e 1. Quão mais preocupado com o custo estiver o usuário, maior será o peso que este confere a pc. Seja pm a importância que o usuário confere ao *makespan* obtido na execução e se aplique a restrição $pm + pc = 1$, de forma que $pm = 1 - pc$. Considera-se MIN_MKSP como o menor *makespan* obtido dentre todas as instâncias e, de forma



(a) Comparação de Desempenho e Custo por Tarefa na Nuvem



(b) *Trade-off Makespan* Vs Custo na Execução na Nuvem

Figura 6.3: Visualização do *Trade-off Makespan*/Custo da Execução na Nuvem *Spot*

similar, MIN_COST o menor custo obtido. Então, para uma dada instância $inst$, define-se a função $sat(inst, pc)$ como a satisfação obtida, diante das outras opções, ao se executar o *workload* na instância $inst$ tendo pc como a ponderação para o custo. Também, $mksp(inst)$ e $cost(inst)$ representam, respectivamente, o *makespan* e o custo referentes à instância $inst$.

$$sat(inst, pc) = \frac{1}{\{\frac{cost(inst)-MIN\_COST}{MIN\_COST} + 1\} \cdot pc + \{\frac{mksp(inst)-MIN\_MKSP}{MIN\_MKSP} + 1\} \cdot pm} \quad (6.1)$$

A Figura 6.4 ilustra como as diferentes instâncias se comportam diante de uma varredura de parâmetros para o fator pc. Mais uma vez, destaca-se a instância *c1.xlarge* apresentando, além de tudo, uma notável resistência diante da variação de pc. De acordo com a métrica $sat(inst, pc)$, apenas em dois cenários extremos alguma outra instância a supera: **1.** Quando a preocupação com o custo é inferior a 0.1, em que a superação se dá pela *m2.4xlarge* e **2.** Quando a preocupação com o custo é superior a 0.9, em que a *c1.medium* se destaca.

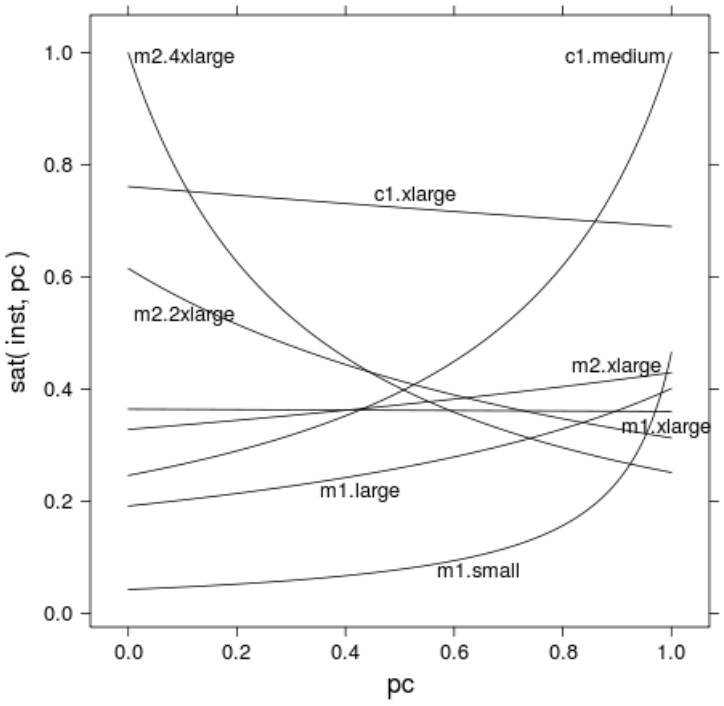


Figura 6.4: Makespan e Custo para as Instâncias *Spot*

Pela discussão anterior, percebe-se que a instância *c1.xlarge* representa a opção mais adequada para servir de referência para estimar o valor que uma grade entre pares fornece a

seus usuários de aplicações BoT típicas de *e-science*, visto que apresenta a melhor relação entre custo e benefício. Além desta, também será considerada a instância *c1.medium* para ilustrar o valor da grade no cenário extremo em que há prevalência de preocupação com o custo. Embora a instância *m2.4xlarge* tenha se destacado no outro cenário extremo (preocupação máxima com o *makespan*), esta não será considerada para as estimativas da próxima seção, visto que este extremo não corresponde à característica de melhor-esforço da grade entre pares.

6.2 Estimando o Valor de uma Grade entre Pares

De início, será considerado um cenário em particular para servir de base para as instanciações das equações referentes às métricas de custo e de valor definidas na Seção 5.2.2. Ao final, serão apresentados os números referentes a todos os cenários e discutidos os aspectos mais relevantes.

Para o cenário de uma grade com 50 *peers*, cada um dispondo de 30 máquinas, com a instância *c1.xlarge* como parâmetro de comparação, têm-se valores médios para $M_c = 789.019,6s$, $C_c = 4.207,953\ USD$ e $M_g = 1.383.318,9s$. Com estes parâmetros, a Equação 5.4 (pg. 43) fornece:

$$\begin{aligned} V_g &= \frac{789.019,6}{1.383.318,9} \times 4.207,953 \\ &= 2.400,14\ USD \end{aligned} \tag{6.2}$$

Uma vez obtido um valor para a grade, para verificar a sua efetividade como plataforma de baixo custo, deve-se considerar, também, o seu custo de manutenção.

É importante observar que no cenário em que os usuários não utilizam a grade, visto que executam as aplicações na nuvem, os domínios administrativos aos quais eles pertencem não incorreriam no custo extra de manter seus *desktops* ligados para servirem às tarefas provenientes da grade, quer sejam oriundas do próprio domínio ou de *peers* remotos. Como custos extras, serão considerados os custos com energia para compartilhar a infraestrutura com os demais *peers* da grade.

Então, para ser, de fato, uma solução efetiva, deve-se ter:

$$V_g > C_g \tag{6.3}$$

6.2.1 Estimativa de Custos da Grade entre Pares

Para a estimativa de custo da grade, será utilizado o modelo simplificado para máquinas homogêneas referentes a uma grade que utiliza estratégia de economia de energia, definido na Seção 5.2.2. Embora no ambiente de grade simulado as máquinas sejam heterogêneas, o uso do modelo simplificado com máquinas que pertencem a casos extremos de eficiência energética permite a obtenção de casos limites, englobando quaisquer variações para máquinas heterogêneas.

Supondo que os recursos são utilizados durante um total de 10 horas diárias por seus usuários locais ($h = 10$, equivalente ao horário comercial, de 8 às 18hs) e sabendo que o *workload* cobre um período de 7 dias ($d = 7$), tem-se um total de $7 \times 10 = 70$ horas em que se deve creditar apenas parcialmente o custo de energia à grade. No tempo restante (98 horas), deve-se considerar o custo total. Como não foi modelado o efeito do fim de semana no estudo descrito na Seção 5, considera-se todos os dias como dias normais de trabalho. Considera-se, também, que a ociosidade das máquinas, mesmo em horário comercial, seja de cerca de 70% ($r = 0,7$), conforme identificado por Kondo et al. [41].

O custo por *kWh* varia consideravelmente de região para região e também de acordo com o tipo do consumidor (se serviço público, residencial, comercial ou industrial). Por exemplo, no Brasil a tarifa praticada para consumidores residenciais varia mais de 100% a depender da operadora e da região [10]. Para evitar problemas com conversão de moeda, visto que o custo por tarefa é estimado em dólares americanos (*USD*), será considerado o custo médio por kWh cobrado de instituição educacionais nos Estados Unidos [39], o que equivale a 5 centavos de dólar ($k = 0,05$). Abordagem semelhante foi utilizada por Kondo et al. [39], de onde foi retirado o custo por *kWh* utilizado.

Consultando a Tabela 5.3 (pg. 44), pode-se perceber dois extremos de consumo de energia, com uma máquina mais eficiente ($c^t_{min} = 115$) do ponto de vista energético e uma extremamente ineficiente ($c^t_{max} = 198$). Desta forma, podem ser encontrados os dois valores limites para a estimativa de custo citados no início desta seção, C_g^{min} e C_g^{max}. Para o

estado de economia de energia, será adotado o valor de 3,33 *Watts* para ambas as máquinas ($c^v = 3,33$). Ainda, para o cenário com 50 *peers*, a taxa média de utilização das máquinas pela grade é de 48,44% ($u = 0,4844$).

De posse dos parâmetros e instanciando a Equação 5.9 (pg. 47) para os dois extremos se tem:

$$\begin{aligned} CDPM^v_{min} &= \frac{0,05}{1000} \times \{10 \times 0,7 \times 0,4844 \times (115 - 3,33) + \\ &\qquad (24 - 10) \times [0,4844 \times 115 + (1 - 0,4844) \times 3,33]\} \\ &= 0.0591286\ USD \end{aligned} \tag{6.4}$$

$$\begin{aligned} CDPM^v_{max} &= \frac{0,05}{1000} \times \{10 \times 0,7 \times 0,4844 \times (78 - 3,33) + \\ &\qquad (24 - 10) \times [0,4844 \times 198 + (1 - 0,4844) \times 3,33]\} \\ &= 0.1013441\ USD \end{aligned} \tag{6.5}$$

Em resumo, o custo diário extra por máquina que pode ser atribuído à grade varia entre $0.0591286\ USD$ e $0.1013441\ USD$. Reconsiderando a constatação da Inequação 6.3, para o corrente caso de 50 *sites* ($np = 50$), cada um com 30 máquinas ($nm = 30$), e instanciando a Equação 5.7 (pg. 45):

$$\begin{aligned} C_g^{min} &= 7 \times 50 \times 30 \times 0.0591286 \\ &= 620,85\ USD \end{aligned} \tag{6.6}$$

$$\begin{aligned} C_g^{max} &= 7 \times 50 \times 30 \times 0.1013441 \\ &= 1.064,11\ USD \end{aligned} \tag{6.7}$$

Como se obteve $V_g = 2.400,14$ na Equação 6.2, que é maior que C_g no pior caso (todas as máquinas sendo ineficientes do ponto de vista energético), conclui-se, considerando todas as simplificações e aproximações realizadas, que a grade é, de fato, uma solução de baixo

custo para a execução de aplicações *BoT*. A Tabela 6.1 resume as considerações para todos os tamanhos de grades utilizados neste trabalho. Os valores estão ordenados de acordo com a coluna Eff^{V_g} no pior caso, que representa a efetividade da grade como solução de baixo custo, cujos valores são calculados da seguinte forma:

$$Eff^{V_g} = \frac{V_g - C_g}{V_g}$$

A métrica percentual Eff^{V_g} indica como e quanto o valor da grade se comporta em relação ao custo de manutenção. Um valor de 100% indicaria um cenário de execução sem custo, enquanto um valor negativo indica um custo superior ao valor obtido. Esta métrica captura a mesma ideia de *margem de lucro líquido*, utilizada na contabilidade financeira. A Figura 6.5 ilustra esta comparação entre o custo e o valor fornecido pela grade.

Ainda com relação à comparação com a instância *c1.xlarge*, todos os cenários apresentaram valores positivos para Eff^{V_g}, embora para $np \leq 30$ essa efetividade seja sempre inferior a 50%. Para todos os cenários com $np \geq 60$, Eff^{V_g} aparenta se estabilizar aproximadamente entre 60% e 70%, reflexo da tendência apresentada na Figura 6.2 (pg. 51).

Para o caso de usuários que se preocupam primariamente com o custo, o que talvez reflita melhor o perfil dos usuários de aplicações *BoT* típicas de *e-science*, a Tabela 6.2 e a Figura 6.6 apresentam os resultados da estimativa de valor da grade quando comparada com a instância *c1.medium*. Neste caso, observa-se que as estimativas para V_g são bem superiores quando comparadas com as respectivas contrapartes na Tabela 6.1. Além disto, a efetividade é superior a 50% para todos os casos e, já a partir de $np = 30$, situa-se sempre acima de 75%. Com a instância *c1.medium* se pode perceber melhor o efeito negativo do limite $L = 100$ na nuvem, visto que esta conta com apenas 2 núcleos, então propicia um nível máximo de paralelização de 200 tarefas simultâneas por usuário (na *c1.xlarge* este limite é de 800 tarefas).

De acordo com os resultados apresentados, observa-se a efetividade da grade entre pares como solução de baixo custo para a execução de aplicações *BoT*. É importante salientar que, na prática, as instâncias não pertencentes à família *standard (m1)* são mais difíceis de serem obtidas no modelo *Spot Instances* (a oferta é menor) e, conforme se pode perceber na Figura 2.2 (pg. 13), são as mais susceptíveis à oscilação do *spot-price*. Diante desta consideração, deve-se interpretar o resultado obtido na nuvem *spot* mais como algo nominal

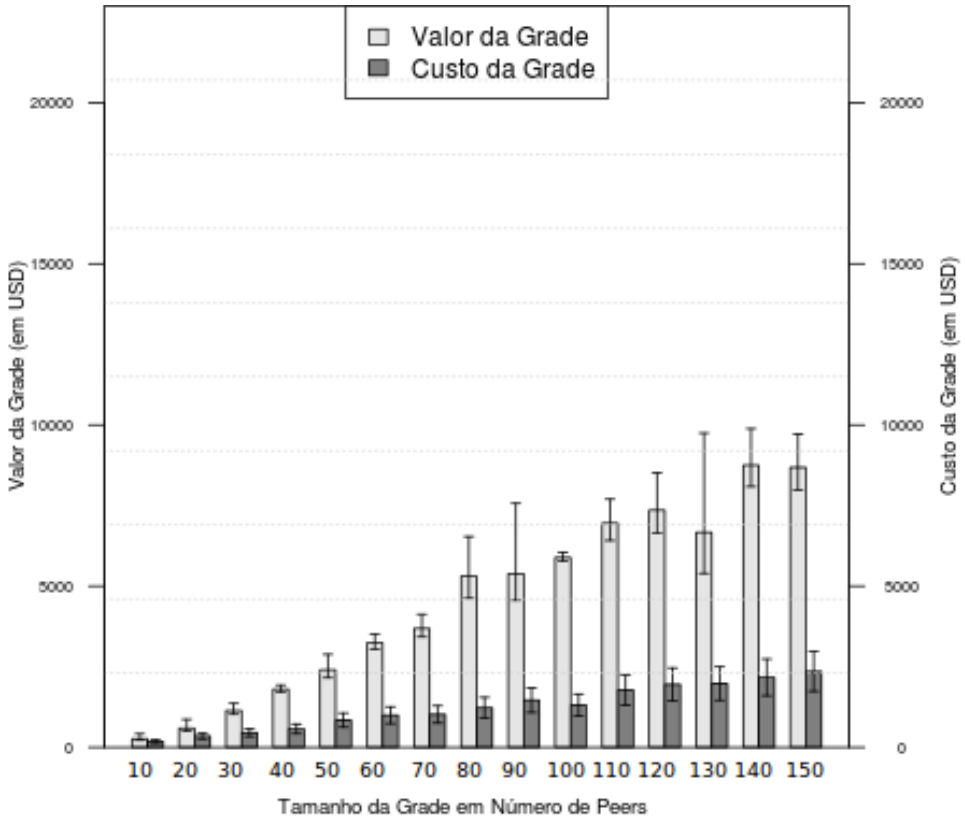


Figura 6.5: Valor da Grade em Comparação com a Instância *c1.xlarge*

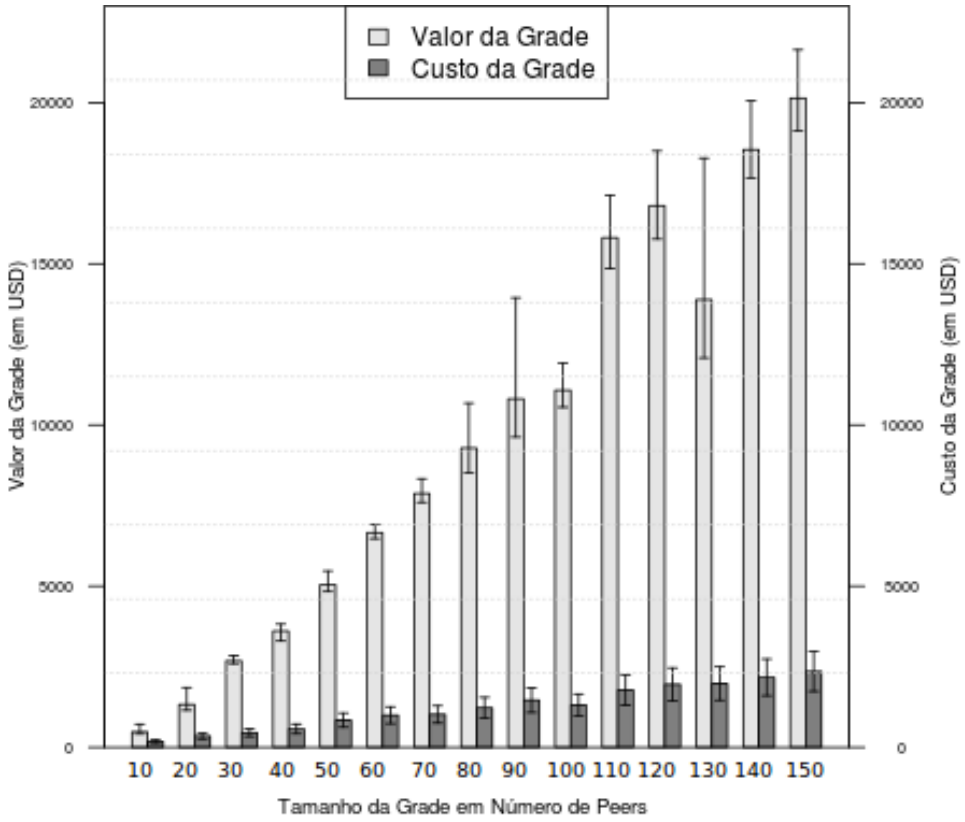


Figura 6.6: Valor da Grade em Comparação com a Instância *c1.medium*

Tabela 6.1: Valor da Grade em Comparação com a Instância *c1.xlarge*[1]

np	**u** (%)	$CDPM^v$ (USD)		C_g (USD)		V_g (USD)	Eff^{V_g} (%)	
		min	max	min	max		max	min
10	52,09	0,06341	0,1088	133,16	228,49	261,94	49,16	12,77
20	50,55	0,06160	0,1057	258,73	443,75	590,30	56,17	24,83
30	43,18	0,05296	0,09059	333,64	570,71	1.139,84	70,73	49,93
50	48,44	0,05913	0,1013	620,84	1.064,09	2.400,14	74,13	55,67
40	40,95	0,05034	0,08602	422,86	722,60	1.802,10	76,54	59,90
60	47,72	0,05828	0,09987	734,36	1.258,35	3.243,62	77,36	61,21
130	43,58	0,05343	0,09142	1.458,75	2.495,68	6.668,36	78,12	62,57
70	41,72	0,05125	0,0876	753,33	1.287,78	3.688,04	79,57	65,08
90	46,68	0,05707	0,09776	1.078,63	1.847,59	5.385,75	79,97	65,69
150	45,02	0,05512	0,09436	1.736,32	2.972,29	8.681,76	80,00	65,76
120	46,67	0,05705	0,09773	1.437,73	2.462,67	7.357,43	80,46	66,53
110	46,53	0,05689	0,09744	1.314,14	2.250,88	6.956,24	81,11	67,64
140	42,83	0,05256	0,08989	1.545,16	2.642,68	8.299,74	81,38	68,16
80	43,96	0,05388	0,0922	905,17	1.548,84	5.310,39	82,95	70,83
100	37,39	0,04618	0,07877	969,73	1.654,11	5.887,30	83,53	71,90

[1] Valores ordenados de acordo com a coluna Eff^{V_g} no pior caso

do que, pelo menos atualmente, passível de ser obtido na prática.

Além disto, a suposição feita na Seção 5.1.2, que os lances dos usuários são sempre superiores ao *spot-price*, não tem como se sustentar nesta situação prática, visto que, a depender dos interesses de negócio da *Amazon*, o *spot-price* pode ser posto em um patamar artificialmente alto com o único propósito de forçar a preempção das instâncias alocadas. Tal procedimento é perfeitamente compreensível, visto que o modelo *spot* se baseia na capacidade excedente e é provido para o usuário sem garantias. Na verdade, com o objetivo de estimar o valor da grade no pior caso, o modelo de nuvem simulado se utiliza da característica dedicada presente nos modelos *on-demand* e *reserved* e nos menores preços e no maior limite do modelo *spot*.

Diante do exposto, é bastante razoável o desempenho da grade diante de *c1.xlarge*, a melhor instância (considerando o *trade-off makespan*/custo) e notável o desempenho quando

Tabela 6.2: Valor da Grade em Comparação com a Instância *c1.medium*[1]

np	**u** (%)	$CDPM^v$ (USD)		C_g (USD)		V_g (USD)	Eff^{V_g} (%)	
		min	max	min	max		max	min
10	52,09	0,06341	0,1088	133,16	228,49	477,60	72,12	52,16
20	50,55	0,06160	0,1057	258,73	443,75	1.326,95	80,50	66,56
30	43,18	0,05296	0,09059	333,64	570,71	2.683,51	87,57	78,73
50	48,44	0,05913	0,1013	620,84	1.064,09	5.045,56	87,70	78,91
40	40,95	0,05034	0,08602	422,86	722,60	3.606,50	88,28	79,96
60	47,72	0,05828	0,09987	734,36	1.258,35	6.660,78	88,97	81,11
130	43,58	0,05343	0,09142	1.458,75	2.495,68	13.890,03	89,50	82,03
90	46,68	0,05707	0,09776	1.078,63	1.847,59	10.812,06	90,02	82,91
80	43,96	0,05388	0,0922	905,17	1.548,84	9.284,05	90,25	83,32
70	41,72	0,05125	0,0876	753,33	1.287,78	7.872,68	90,43	83,64
140	42,83	0,05256	0,08989	1.545,16	2.642,68	17.476,84	91,16	84,88
100	37,39	0,04618	0,07877	969,73	1.654,11	11.078,81	91,25	85,07
150	45,02	0,05512	0,09436	1.736,32	2.972,29	20.141,44	91,38	85,24
120	46,67	0,05705	0,09773	1.437,73	2.462,67	16.804,89	91,44	85,35
110	46,53	0,05689	0,09744	1.314,14	2.250,88	15.810,31	91,69	85,76

[1] Valores ordenados de acordo com a coluna Eff^{V_g} no pior caso

comparado com a instância *c1.medium*, a menos custosa, mas que oferece bom desempenho para a execução do *workload*.

Por fim, para realizar uma estimativa de valor da grade de forma relativa, a Tabela 6.3 e a Figura 6.7 exibem o valor por hora-máquina proporcionado pela grade, $VPHM$, calculado da seguinte forma:

$$VPHM = \frac{V_g - C_g}{(\,d \cdot 24\,) \cdot (\,np \cdot nm\,)}$$

Tal forma de valoração se assemelha à forma de tarifação comumente praticada na provisão de IaaS, que é o valor cobrado por hora de uso de uma instância. A Figura 6.7 apresenta o resultado da valoração relativa com a métrica $VPHM$. No gráfico, a barra de erros em cada ponto representa a eficiência energética das máquinas, isto é, para o limite inferior se tem máquinas que fornecem um $CDPM^v_{max}$ enquanto o limite superior se refere a máquinas

Tabela 6.3: Valor por Hora-Máquina Proporcionado Pela Grade

np	**Instância de Comparação**			
	c1.xlarge		*c1.medium*	
	max (USD)	min (USD)	max (USD)	min (USD)
10	0,002555	0,0006635	0,006834	0,004943
20	0,003289	0,001454	0,010597	0,008762
30	0,005332	0,003764	0,015541	0,013974
40	0,006841	0,005355	0,015792	0,014305
50	0,007061	0,005302	0,017558	0,015799
130	0,007951	0,006369	0,018973	0,017391
60	0,008298	0,006565	0,019598	0,017865
70	0,008318	0,006803	0,020180	0,018665
100	0,009757	0,008399	0,020058	0,018700
80	0,010926	0,009329	0,020781	0,019185
90	0,009495	0,007800	0,021458	0,019763
140	0,009573	0,008017	0,022579	0,021023
150	0,009187	0,007552	0,024345	0,022711
120	0,009788	0,008093	0,025409	0,023714
110	0,010177	0,008487	0,026147	0,024458

mais eficientes (fornecem um $CDPM^{v}_{min}$). Pode ser percebido que, a partir do ponto de saturação, quando comparado com a instância menos custosa, uma hora de uma máquina da grade vale aproximadamente $0.02\ USD$.

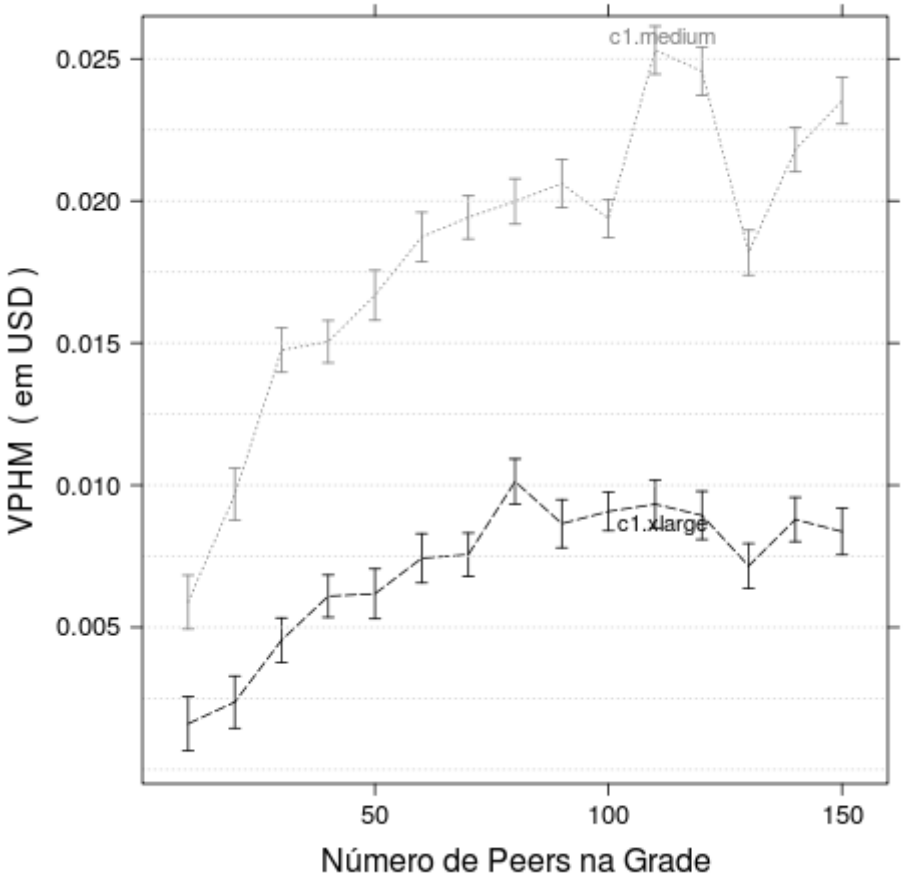


Figura 6.7: Valor por Hora-Máquina Proporcionado Pela Grade

Capítulo 7

Conclusão e Trabalhos Futuros

Grades entre Pares e *Infraestrutura como Serviço* (IaaS), um dos produtos oferecidos pela *Computação na Nuvem*, são duas opções atraentes para a execução de aplicações massivamente paralelas de computação intensiva, denominadas *Bag-of-Tasks* (*BoT*).

No presente trabalho, as duas plataformas foram simuladas com o objetivo de comparação de desempenho e posterior atribuição de valor à grade com base nos custos de execução na IaaS. Cada sistema foi estimulado com um mesmo *workload* típico de aplicações *BoT* e, com base na análise da relação custo/benefício, foram identificadas as melhores opções de execução na nuvem, sendo uma para o caso geral e a outra para usuários cujo principal interesse reside na redução de custos. Ao final, ambas foram consideradas para as estimativas de valor para a grade. Também foram considerados os custos extras incorridos aos participantes da grade, objetivando identificar se a plataforma de grade se mantém efetivamente como solução de baixo custo diante da concorrência dos baixos preços e melhor qualidade de serviço oferecida pelos provedores de IaaS.

Os resultados obtidos confirmam a efetividade da grade como solução de baixo custo e, devido a limitações de escalabilidade dos atuais provedores de IaaS, considerando a relação custo/benefício, consegue obter uma valoração competitiva mesmo diante da melhor opção atualmente existente na nuvem. Ainda, para usuários preocupados majoritariamente com o custo, mesmo com poucos participantes, a grade se apresenta como opção superior à nuvem. Para cenários de grades pequenas e usuários cuja principal preocupação está no desempenho, e não no custo, a grade não se mostra como opção efetiva e a nuvem, pelo menos nominalmente, apresenta-se como a opção mais adequada.

Como trabalho futuro, pode-se melhorar as estimativas de custo de manutenção da grade entre pares ao adicionar outros fatores que exercem influencia no custo, a exemplo do impacto do aumento de utilização na vida útil dos equipamentos e custos com pessoal. Outro aspecto necessário para um trabalho futuro é a simulação de um cenário mais realista para a nuvem, que considere restrições orçamentárias para a execução, além de incluir os outros custos inerentes à nuvem, a exemplo de custo de comunicação e de armazenamento. Espera-se que, em cenários mais realistas, os valores estimados para a grade sejam ainda maiores que os obtidos no presente trabalho.

Também visando a uma maior proximidade com a realidade, é importante analisar os mesmos cenários deste trabalho com uma estratégia de escalonamento na grade que use replicação de tarefas, nos moldes da *WQR* (*WorkQueue with Replication*) [54]. O intuito seria verificar se o ganho potencial de desempenho obtido às custas de um maior desperdício de recursos compensa o aumento do custo da grade, proporcionado pelo maior consumo de energia.

Uma vez que neste trabalho todas as análises consideraram o valor para a grade como um todo, é importante verificar como os valores obtidos se distribuem entre os diferentes *peers*, de forma a identificar em quais situações a participação na grade é sempre positiva para o *peer* e, na outra ponta, em quais situações o uso da nuvem seria sempre vantajosa. Um caminho para tanto é tentar relacionar o *workload* do *peer*, a quantidade de recursos que este possui para contribuir com a grade e o limite máximo de instâncias que podem ser mantidas simultaneamente na nuvem.

Bibliografia

[1] BOINC Papers, http://boinc.berkeley.edu/trac/wiki/BoincPapers. 2010.

[2] Folding@home Papers, http://folding.stanford.edu/English/Papers. 2010.

[3] SETI@home, http://setiathome.berkeley.edu. 2010.

[4] What is e-science?, http://www.escience-grid.org.uk/what-e-science.html. 2010.

[5] Amazon.com. Amazon, http://www.amazon.com, 2010.

[6] David P. Anderson and Gilles Fedak. The Computational and Storage Potential of Volunteer Computing. In *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*, CCGRID '06, pages 73–80, Washington, DC, USA, 2006. IEEE Computer Society.

[7] Nazareno Andrade, Francisco Vilar Brasileiro, Walfredo Cirne, and Miranda Mowbray. Automatic Grid Assembly by Promoting Collaboration in Peer-to-Peer Grids. *Journal of Parallel and Distributed Computing*, 67:957–966, 2007.

[8] Artur Andrzejak, Derrick Kondo, and David P Anderson. Ensuring Collective Availability in Volatile Resource Pools Via Forecasting. *IFIP - International Federation For Information Processing*, 2008.

[9] Artur Andrzejak, Derrick Kondo, and Sangho Yi. Decision Model for Cloud Computing under SLA Constraints. *Modeling, Analysis, and Simulation of Computer Systems, Int. Symp. on*, 0, 2010.

[10] ANEEL. http://www.aneel.gov.br/area.cfm?idarea=493. 2010.

[11] Cosimo Anglano, Massimo Canonico, Marco Guazzone, Marco Botta, Sergio Rabellino, Simone Arena, and Guglielmo Girardi. Peer-to-Peer Desktop Grids in the Real World: The ShareGrid Project. *Cluster Computing and the Grid, IEEE International Symposium on*, 0:609–614, 2008.

[12] Eliane Araújo, Walfredo Cirne, Gustavo Wagner, Nigini Oliveira, Enio P. Souza, Carlos O. Galvão, and Eduardo Sávio Martins. The SegHidro Experience: Using the Grid to Empower a Hydro-Meteorological Scientific Network. *e-Science and Grid Computing, International Conference on*, 0, 2005.

[13] Michael Armbrust, Armando Fox, and Rean Griffith. Above the Clouds: A Berkeley View of Cloud Computing, Tech. Rep. EECS-2009-28, EECS Dept, Univ. of California, Berkeley. 2009.

[14] Amazon AWS. Amazon Web Services (AWS), http://aws.amazon.com, 2010.

[15] B. A. Aylward and E. B. Barbier. Valuing Environmental Functions in Developing Countries. *Biodiv. Cons. 1, 34*, 1992.

[16] J.-P. Barde and D. W. Pearce. Valuing the Environment: Six Case Studies. 1991.

[17] W Bolosky, J Douceur, D Ely, and M Theimer. Feasibility of a Serverless Distributed file System Deployed on an Existing Set of Desktop PCs. *In Proceedings of SIGMETRICS*, 2000.

[18] J Brevik, D Nurmi, and R Wolski. Automatic Methods for Predicting Machine Availability in Desktop Grid and Peer-to-peer Systems. *in IEEE International Symposium on Cluster Computing and the Grid (CCGrid'04), Apr*, 2004.

[19] Marcus Williams Aquino de Carvalho. Predição da Qualidade de Serviço em Grades Computacionais P2P. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, Março 2011.

[20] Andrew Chien, Brad Calder, Stephen Elbert, and Karan Bhatia. Entropia: architecture and performance of an enterprise desktop grid system. *J. Parallel Distrib. Comput.*, 63:597–610, May 2003.

[21] Walfredo Cirne, Francisco Brasileiro, Nazareno Andrade, Lauro Costa, Alisson Andrade, Reynaldo Novaes, and Miranda Mowbray. Labs of the World, Unite!!! *Journal of Grid Computing*, 4(3):225–246, 2006.

[22] Rostand Costa, Francisco Vilar Brasileiro, Guido Lemos de Souza Filho, and Dênio Mariz Sousa. Sobre a Amplitude da Elasticidade dos Atuais Provedores de Computacao na Nuvem. In *Anais do XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC2011)*, Campo Grande, MS, Brasil, Maio 2011. Sociedade Brasileira de Computação (SBC).

[23] R Costanza, S. C. Farber, and J Maxwell. Valuation and Management of Wetlands Ecosystems. *Ecol. Econ. 1*, pages 335–361, 1989.

[24] R S De Groot, M A Wilson, and R M J Boumans. A typology for the classification, description and valuation of ecosystem functions, goods and services. *Ecological Economics*, 41(3):393–408, 2002.

[25] Alexandra Dehnhardt. The replacement value of flood plains as nutrient sinks: a case study of the river Elbe. *World Congress of Environmental and Resource Economists*, 2002.

[26] J. A. Dixon and P. B. Sherman. Economics of Protected Areas. 1990.

[27] Amazon EC2. Amazon Elastic Compute Cloud (Amazon EC2), http://aws.amazon.com/ec2, 2010.

[28] Amazon EC2FAQ. Amazon EC2 FAQs, http://aws.amazon.com/ec2/faqs/, 2010.

[29] Amazon ECU. Amazon EC2 Computing Unit (ECU), http://aws.amazon.com/ec2/instance-types/, 2010.

[30] Scott Eliason. *Maximum Likelihood Estimation - Logic and Practice*, volume 96 of *Quantitative Applications in the Social Sciences*. Sage Publications, 1993.

[31] D. H. J. Epema, M. Livny, R. van Dantzig, X. Evers, and J. Pruyne. A Worldwide Flock of Condors: Load Sharing among Workstation Clusters. *Future Gener. Comput. Syst.*, 12, May 1996.

[32] Scott Fields. Hunting for Wasted Computing Power, 1993 Research Sampler, University of Wisconsin-Madison, http://www.cs.wisc.edu/condor/doc/WiscIdea.html, 1993.

[33] Ian Foster. What is the Grid? - a three point checklist. *GRIDtoday*, 1(6), julho 2002.

[34] S. Garfinkel. Commodity Grid Computing with Amazons S3 and EC2. *Login*, 32(1), 2007.

[35] T. Hubner. Power Consumption of Current Processors, http://www.computerbase.de/artikel/prozessoren/2004/bericht-energieverbrauch-aktueller-prozessoren/. May 2004.

[36] Alexandru Iosup and Dick Epema. Grid Computing Workloads: Bags of Tasks, Workflows, Pilots, and Others. *IEEE Internet Computing*, 99, 2010.

[37] Alexandru Iosup, Mathieu Jan, Ozan Sonmez, and Dick Epema. The Characteristics and Performance of Groups of Jobs in Grids. In *In Euro-Par, volume 4641 of LNCS*. Springer-Verlag, 2007.

[38] Ivan I. Ivanov. Utility Computing: Reality and Beyond. In Joaquim Filipe and Mohammad S. Obaidat, editors, *E-business and Telecommunications*, volume 23 of *Communications in Computer and Information Science*, pages 16–29. Springer Berlin Heidelberg, 2009. 10.1007/978-3-540-88653-2_2.

[39] D. Kondo, B. Javadi, P. Malecot, F. Cappello, and D.P. Anderson. Cost-Benefit Analysis of Cloud Computing Versus Desktop Grids. In *Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–12, May 2009.

[40] D. Kondo, M. Taufer, C.L. Brooks, H. Casanova, and A.A. Chien. Characterizing and Evaluating Desktop Grids: An Empirical Study. In *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, abril 2004.

[41] Derrick Kondo, Gilles Fedak, Franck Cappello, Andrew A. Chien, and Henri Casanova. Characterizing Resource Availability in Enterprise Desktop Grids. *Future Gener. Comput. Syst.*, 23:888–903, August 2007.

[42] Alexander Lenk, Markus Klems, Jens Nimis, Stefan Tai, Thomas Sandholm, Hewlett-packard Laboratories, and Palo Alto. What is Inside the Cloud? An Architectural Map of the Cloud Landscape. *Management*, pages 23–31, 2009.

[43] Frank J Massey. The Kolmogorov Smirnov Test for Goodness of Fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951.

[44] University of Wisconsin Milwaukee. Einstein@home, http://www.einsteinathome.org/.

[45] Alek Opitz, Hartmut König, and Sebastian Szamlewska. What Does Grid Computing Cost? *Journal of Grid Computing*, 6:385–397, 2008. 10.1007/s10723-008-9098-8.

[46] Ana Maria Oprescu and Thilo Kielmann. Bag-of-Tasks Scheduling under Time and Budget Constraints. *IEEE Int. Conf. and Workshops on Cloud Comp. Tech. and Science (CloudCom 2010)*, 2010.

[47] Ourgrid.org. OurGrid, http://www.ourgrid.org/, 2004.

[48] Ourgrid.org. OurSim - OurGrid Simulator, http://redmine.lsd.ufcg.edu.br/projects/show/oursim, 2010.

[49] D. Pearce. Economic Values and the Natural World. 1993.

[50] David Pearce and Ece O zdemiroglu et al. Economic Valuation with Stated Preference Techniques - Summary Guide. *Department for Transport*, 2002.

[51] Lesandro Ponciano and Francisco Brasileiro. On the impact of energy-saving strategies in opportunistic grids. In *Energy Efficient Grids, Clouds and Clusters Workshop, proceedings of the 11th ACM-IEEE International Conference on Grid Computing (Grid 2010),*, pages 282 – 289, Brussels, Belgium, 2010. ACM-IEEE.

[52] Cesar A. F. De Rose, Tiago C. Ferreto, Marcelo B. de Farias, Vladimir G. Dias, Walfredo Cirne, Milena P. M. Oliveira, Katia Saikoski, and Maria Luiza Danieleski. Gerpavgrid: using the grid to maintain the city road system. In *Computer Architecture and High Performance Computing, 2006. SBAC-PAD '06. 18TH International Symposium on*, pages 73 –80, 2006.

[53] Lesandro Ponciano dos Santos. Avaliação do Impacto de Estratégias de Economia de Energia em Grades Computacionais Entre-Pares. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, Março 2011.

[54] Daniel Paranhos da Silva, Walfredo Cirne, and Francisco Vilar Brasileiro. Trading Cycles for Information: Using Replication to Schedule Bag-of-Tasks Applications on Computational Grids. In Harald Kosch, László Böszörményi, and Hermann Hellwagner, editors, *Euro-Par 2003 Parallel Processing*, volume 2790 of *Lecture Notes in Computer Science*, pages 169–180. Springer Berlin / Heidelberg, 2003. 10.1007/978-3-540-45209-6_26.

[55] Yogesh Simmhan, Catharine van Ingen, Girish Subramanian, and Jie Li. Bridging the gap between desktop and the cloud for escience applications. *Cloud Computing, IEEE International Conference on*, 0:474–481, 2010.

[56] Amazon EC2 SLA. Amazon EC2 Service Level Agreement, http://aws.amazon.com/ec2-sla/, 2010.

[57] Amazon EC2 Spot. Amazon Spot Instances, http://aws.amazon.com/ec2/spot-instances/, 2010.

[58] Spot-History. Amazon Spot Instances' Spot Prices History, http://cloudexchange.org/, 2010.

[59] University of Stanford. Folding@home, http://folding.stanford.edu/.

[60] Jeffrey E. Thomas and Brad M. Wilson. The Indemnity Principle: Evolution from a Financial to a Functional Paradigm. *Journal of Risk Management and Insurance, Vol. 10, No. 30, 2005*, 2005.

[61] David de Vaus, Mathew Gray, and David Stanton. Measuring the Value of Unpaid Household, Caring and Voluntary Works of older Australians, 2003.

[62] Lei Wang, Jianfeng Zhan, Weisong Shi, Yi Liang, and Lin Yuan. In Cloud, do MTC or HTC Service Providers Benefit from the Economies of Scale? In *Proceedings of the*

2nd Workshop on Many-Task Computing on Grids and Supercomputers, MTAGS '09. ACM, 2009.

[63] Alan Wilter, Carla Osthoff, Cristiane Oliveira, Diego E.B. Gomes, Eduardo Hill, Laurent E. Dardenne, Patrícia M. Barros, Pedro A.A.G.L. Loureiro, Reynaldo Novaes, and Pedro G. Pascutti. The BioPAUÁ Project: A Portal for Molecular Dynamics Using Grid Environment. 3594:214–217, 2005. 10.1007/11532323_26.

[64] XtremWeb. XtremWeb, http://www.xtremweb.net, 2004.

[65] Sangho Yi, Derrick Kondo, and Artur Andrzejak. Reducing Costs of Spot Instances via Checkpointing in the Amazon Elastic Compute Cloud. In *Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing*, CLOUD '10. IEEE Computer Society, 2010.