Estimando o Valor de uma Grade Entre Pares para a Execução de Aplicações do Tipo Saco de Tarefas

Edigley Fraga, Francisco Brasileiro, Dalton Serey

Universidade Federal de Campina Grande
Departamento de Sistemas e Computação
Laboratório de Sistemas Distribuídos
Av. Aprígio Veloso, s/n, Bloco CO, Bodocongó
58.429-900, Campina Grande, PB

{edigley,fubica,dalton}@dsc.ufcg.edu.br

Abstract. *Desktop grids have been used as a low cost solution to execute highly parallel applications that require low QoS from the execution infrastructure (as known as Bag-of-Tasks or BoT). More recently Infrastructure as a Service (IaaS) Cloud Computing has emerged as an attractive alternative for the execution of BoT that offers both low cost and improved QoS control. In this work, we compare the performance and the value yielded by both types of infrastructure for the execution of typical e-sience BoT. Our results indicate that in typical scenarios in which there is not high contention, desktop grids can yield services of the same quality level as IaaS providers at much lower costs. In fact, to beat grids in such scenarios, IaaS prices should be reduced about 50% to 70%.*

Resumo. *Grades de desktops têm sido usadas como solução de baixo custo para a execução de aplicações massivamente paralelas (BoT) desde os anos 90. Recentemente, a computação na nuvem, por meio da provisão de Infraestrutura como Serviço (IaaS), emergiu como alternativa atraente para a execução de aplicações BoT. Neste trabalho, comparamos, via simulação, os desempenhos obtidos nas duas infraestruturas ao serem utilizadas para a execução de aplicações BoT típicas de e-science. Os resultados indicam que, em cenários de baixa e média contenção, grades eferecem, a um custo inferior, serviços com desempenho similar ao obtido na IaaS. De fato, para superar as grades, os preços praticados em IaaS deveriam sofrer uma redução entre 50% e 70%.*

1. Introdução

O acesso a grandes quantidades de recursos computacionais tem impactado a forma como a ciência é conduzida em diversos campos de conhecimento. O uso intenso da computação para gerar e processar dados científicos, denominado de *e-science*, é possível porque boa parte das aplicações pode ser facilmente executada de forma paralela em diversos recursos computacionais, acelerando processos iterativos de geração e análise de dados.

Aplicações dessa natureza são conhecidas como sacos de tarefas (ou *BoT* - do inglês *bag-of-tasks*) [Iosup and Epema 2011] em função de serem constituídas por uma grande quantidade de tarefas que podem ser executadas de forma independente, facilitando consideravelmente as atividades de escalonamento e tolerância a falhas. A característica mais importante dessas aplicações é a não-exigência de uma rígida garantia de

qualidade de serviço por parte da plataforma de execução. Em teoria, qualquer equipamento dotado de um microprocessador é capaz de executar tarefas de uma aplicação *BoT*.

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* [UCB 2011a]. Tais infraestruturas se baseiam no aproveitamento oportunista de recursos, utilizando ciclos ociosos de máquinas *desktops* não dedicadas. Um dos primeiros representantes de grades dessa natureza foi o projeto Condor [Fields 1993]. Inspirados pelo sucesso das primeiras grades de *desktops*, plataformas de computação voluntária conseguiram agregar milhões de *desktops* de usuários domésticos espalhados por todos os continentes [UCB 2011b]. Mais recentemente, grades entre pares também têm sido implantadas e utilizadas para a execução de uma variedade de aplicações *BoT* [Araujo et al. 2005, Anglano et al. 2008].

Em meados da década passada, a computação na nuvem, na forma de provisão de Infraestrutura como Serviço (*IaaS*), emergiu como uma alternativa para a execução de aplicações *BoT*. Uma característica notável presente no paradigma da computação na nuvem é a elasticidade: capacidade de ser virtualmente e quase que instantaneamente expandida ou reduzida, sob demanda e controle direto do cliente [Armbrust et al. 2009]. Aliada ao baixo custo, tal propriedade o torna atraente para a execução de aplicações BoT.

Até o presente momento tem sido difícil estimar o valor que uma grade de *desktops* oferece aos usuários de aplicações *BoT*. Em particular, a tarefa é especialmente difícil se quisermos usar métricas de negócio (valor monetário), visto que a receita financeira resultante da execução de aplicações de *e-science* é comumente indefinida. Neste trabalho, para atingir este objetivo, usamos 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 a referência para estimar o valor proporcionado pela grade de *desktops*.

A metodologia é baseada na simulação das duas plataformas de execução, sendo ambas submetidas a um mesmo *workload* (carga de aplicações *BoT*). Para cada plataforma simulada é realizada a medição do *makespan* (tempo entre a submissão e a conclusão) de cada aplicação executada. Para estimar o custo da execução, usamos o método de precificação e os preços praticados pelo *Amazon Web Services*, um dos principais provedores de *IaaS* do mercado. Também estimamos os limites do custo operacional extra em que incorrem os mantenedores de uma grade de *desktops*, de modo a verificar em que situações essa plataforma é mais vantajosa que a contratação de um provedor de *IaaS*.

Os resultados mostram que, para a maior parte das configurações avaliadas, a grade de *desktops* se apresenta como a alternativa mais vantajosa. Em especial, quando a contenção não é muito alta, os preços praticados pelos provedores de IaaS teriam que cair entre 50% e 70% para que a grade de *desktops* deixasse de ser a melhor alternativa.

2. Contextualização

Grade Entre Pares A participação em comunidades de grades entre pares permite às instituições e aos laboratórios participantes reduzir o chamado Custo Total de Propriedade. 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 in-

trusiva, sem prejudicar os usuários locais. Além disto, por ser par da comunidade, o laboratório pode dispor dos ciclos ociosos dos demais pares. Isto implica a disponibilidade de um número de recursos computacionais muito maior do que seria possível obter individualmente. Em troca, o laboratório também disponibiliza seus ciclos ociosos para os demais pares da comunidade. Exemplos de *middlewares* bem sucedidos no suporte a grades oportunistas em larga escala incluem o projeto Condor [Epema et al. 1996] e o OurGrid[1] [Cirne et al. 2006], descrito brevemente a seguir.

Na terminologia OurGrid, denomina-se de *comunidade* o conjunto de laboratórios participantes – também chamados de *sites* ou *peers*, quando se quer enfatizar a natureza P2P (do inglês *peer-to-peer*) da plataforma. O OurGrid é baseado em uma política de livre-entrada, em que 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. Para desencorajar um comportamento egoísta por parte dos *peers*, a plataforma OurGrid prioriza o atendimento aos usuários de acordo com o histórico de doações. Esse mecanismo de incentivo é chamado de Rede-de-Favores (*NoF*, do inglês *Network-of-Favors*) [Andrade et al. 2007] e age de maneira autônoma, baseando-se unicamente na reciprocidade entre os pares.

Amazon Elastic Compute Cloud O *Amazon Elastic Compute Cloud* (*Amazon EC2*) [EC2 2010] 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, flexibilidade (tipos variados de instâncias e sistemas operacionais), confiabilidade e segurança. Pelo modelo de negócio adotado, os usuários são cobrados apenas pelos recursos que efetivamente consomem, de forma semelhante a um mercado de computação utilitária [Ivanov 2009]. Atualmente, são praticados 3 modelos de precificação no *Amazon EC2*, denominados *on-demand*, *reserved* e *spot*. No primeiro, 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 dos recursos computacionais que não foram negociados nos outros dois modelos. Por oferecer os menores preços sem comprometimento antecipado, o modelo *spot* foi o adotado para a comparação com a grade.

Ao solicitar o serviço *spot*, o usuário especifica o tipo e a quantidade de instâncias que deseja alocar, bem como o preço máximo (*lance*), por hora, que está disposto a pagar. A *Amazon* mantém continuamente o valor mínimo (*spot-price*) de cada tipo de instância *spot*. Assim, se o lance for maior ou igual ao *spot-price*, a solicitação será atendida. As instâncias alocadas ficarão disponíveis para o usuário até que este termine seu uso ou até que o *spot-price* supere o lance (o que acontecer primeiro). É importante esclarecer que o preço cobrado pelas instâncias alocadas é o *spot-price* e não o valor do lance. Logo, o valor pago será sempre menor ou igual ao valor do lance. O valor *spot-price* é definido pelo *Amazon EC2* e flutua periodicamente como função da oferta e da demanda de instâncias *spot*. Assim como nos outros dois modelos, horas parciais são cobradas como horas inteiras, a menos que ocorra uma preempção como decorrência do *spot-price* ter superado o valor do lance. Neste caso, não haverá cobrança por fração de hora de uso.

[1] OurGrid, http://www.ourgrid.org

Tabela 1. Tipos de Instâncias *Amazon EC2*

Tipo[1]	Mem[2]	CPU[3]	Núcleos	CPU/Núcleo[3]	Preço ($)[4]		
					Spot	Reserved	OD[5]
$m1.small$	1.7	1	1	1	0.030	0.030	0.085
$m1.large$	7.5	4	2	2	0.124	0.120	0.340
$m1.xlarge$	15	8	4	2	0.250	0.240	0.680
$c1.medium$	1.7	5	2	2,5	0.059	0.060	0.170
$c1.xlarge$	7	20	8	2,5	0.240	0.240	0.680
$m2.xlarge$	17.1	6.5	2	3,25	0.170	0.170	0.500
$m2.2xlarge$	34.2	13	4	3,25	0.435	0.340	1.000
$m2.4xlarge$	68.4	26	8	3,25	0.822	0.680	2.000

[1] Prefixos: *m1 (Stantard), m2 (High-Memory)* e *c1 (High-CPU) Instances* – [2] Em Gigabytes

– [3] Em número de *ECU - EC2 Computing Unit* (aprox. 1,0 $\sim$ 1,2 GHz) – [4] Em Dólar Americano (*USD*), referente a dezembro de 2010 – [5] On-Demand

Algumas restrições, contudo, existem. Cada usuário está limitado a 20 instâncias dos modelos *on-demand* e *reserved* e a um limite de 100 instâncias para o modelo *spot*. Para usar um número maior de instâncias, o usuário deve solicitar formalmente, explicando seu caso de uso, e o pedido poderá, ou não, ser aprovado [2].

Há seis famílias de instâncias: *Standard, Micro, High-Memory, High-CPU, Cluster Compute e Cluster GPU*. Neste trabalho, focamos nosso interesse em três famílias: *Standard* (*m1*), *High-Memory* (*m2*), e *High-CPU* (*c1*). Instâncias *m2* são indicadas para serviços com alto consumo de memória e instâncias *c1* são indicadas para computação de alto desempenho. Detalhes de cada família são apresentados na Tabela 1. Na tabela, o acrônimo ECU (*EC2 Computing Unit*) representa uma unidade virtual de computação definida pelo *EC2* que equivale a aproximadamente um processador de 1,0$\sim$1,2 GHz[3].

3. Modelo de Sistema

Grade Entre Pares: Formalmente, definimos uma grade entre pares G como um conjunto de *sites* S_i: $G = \{S_1, S_2, ..., S_n\},\ n > 1$. Por sua vez, um *site* consiste em um conjunto de máquinas M_i: $S = \{M_1, M_2, ..., M_n\},\ n > 0$. 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* utilizado (*BoT*), a modelagem considerando apenas recursos computacionais já é satisfatória. Já as máquinas são modeladas como n-tuplas de processadores P_i: $M = (P_1, P_2, ..., P_n),\ n > 0$. Por fim, um processador é uma 1-tupla com o seu elemento unitário representando a capacidade computacional ν do processador: $P = (\nu),\ \nu \in \mathbb{N}^+$.

Os conjuntos de recursos dos *sites* são disjuntos ($\forall_{i,j}, i \neq j, S_i \cap S_j = \emptyset$). Como os recursos são utilizados de forma oportunista, as máquinas efetivamente disponíveis para a grade ao longo tempo representam um subconjunto do conjunto de máquinas dos *sites*.

Nuvem de Instâncias *Spot* A plataforma de computação na nuvem C, ou nuvem de instâncias, é modelada por um conjunto de máquinas M e um limite $L > 0$ que indica o número de instâncias que podem ser alocadas por um

[2] http://aws.amazon.com/ec2/faqs/

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

usuário. Um *spot-price* Π é modelado por um par instante de tempo e preço: $\Pi = (t, \pi),\ t \in \mathbb{N},\ \pi \in \mathbb{R}^+$. A oscilação O do *spot-price* é expressa como uma série de *spot-prices*: $O = (\Pi_1, \Pi_2, ..., \Pi_n),\ n \gg 0\ \wedge\ \forall_{i,j}, i < j, \Pi_{i_1} < \Pi_{j_1}$. Há, por fim, a necessidade de considerar os usuários do sistema, dado que o limite de elasticidade L é aplicado por usuário. O conjunto de usuários é $\Upsilon = (\upsilon_1, \upsilon_2, ..., \upsilon_n),\ n > 0$.

3.1. Modelo de *Workload*

Para formalizar o *workload* de aplicações *BoT*, adaptou-se a notação introduzida em [Iosup et al. 2007]. Nessa notação, um workload W é uma sequência de *jobs* [4], ordenada pelo tempo de submissão: $W = \{J_1, J_2, ..., J_n\},\ n > 0$. Um *job*, por sua vez, consiste em um conjunto não vazio de tarefas: $J = \{T_1, T_2, ..., T_n\},\ n > 0$, representando, cada uma, uma demanda computacional de τ unidades de tempo: $T = (\tau),\ \tau \in \mathbb{N}^+$.

Utilizaremos, ainda, a seguinte notação para nos referirmos a propriedades de um job J: $user(J)$, $peer(J)$, $makespan^G(J)$, $makespan^C(J)$ e $cost(J, i)$. As funções se referem, respectivamente, ao usuário que submeteu o *job*, ao *peer* ao qual o usuário pertence, aos *makespans* na grade e na nuvem e ao custo de execução, na nuvem, da i-ésima tarefa do *job* J. Com base nas funções definidas acima, definimos as seguintes macros para facilitar a referência a subconjuntos do workload W com base nos usuários: $W^u = \{J \mid J \in W \wedge user(J) = u\},\ |W^u| \geq 0$; e *peers* de origem: $W^s = \{J \mid J \in W \wedge peer(J) = s\},\ |W^s| \geq 0$.

3.2. Modelo de Simulação

Para avaliar os modelos anteriormente apresentados, optou-se pelo uso de simulação baseada em eventos, implementada no OurSim [5], um simulador de grades entre pares implementado na linguagem Java e que também oferece suporte à simulação de grades oportunistas. Além disto, o OurSim também provê alguns ganchos, o que o tornou adequado para simular o modelo simplificado da nuvem de instâncias *spot*.

Conforme apresentado na Seção 3, nas simulações não foram considerados fatores como rede (tráfego, latência, etc.) nem armazenamento (memória ou disco), visto que o foco são aplicações de computação intensiva. Cada simulação ocorre desde a submissão do primeiro *job* até o término do último, ínterim em que são gerados alternadamente intervalos de indisponibilidade e disponibilidade para cada máquina da grade.

Grade Entre Pares: 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 um componente *Rede-de-Favores*. 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 prioriza a alocação em máquinas locais. Caso os recursos locais não sejam suficientes, as demais tarefas serão submetidas 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. Vale ressaltar que, nas simulações executa-

[4] O termo *job* é utilizado como sinônimo de aplicação *BoT*

[5] OurSim - OurGrid Simulator, http://redmine.lsd.ufcg.edu.br/projects/show/oursim

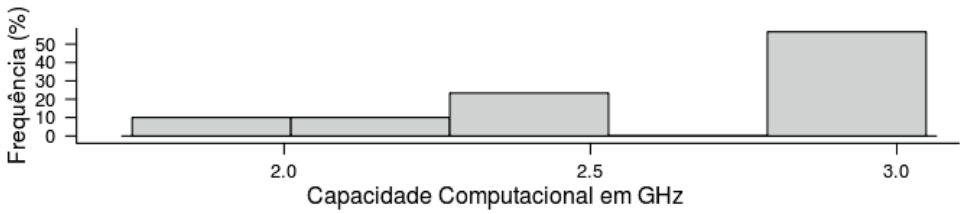

Figura 1. Velocidade das Máquinas de um Site Ourgrid (50 máquinas)

das, não foram considerados mecanismos de replicação ou de *checkpointing*. Diante de um evento de preempção, o *JobScheduler* apenas ressubmete a tarefa preemptada.

Nuvem de Instâncias *Spot*: O principal componente é o escalonador de instâncias *spot* (*SpotScheduler*), 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 pertencentes ao mesmo usuário sejam liberadas. No caso de instâncias com vários núcleos, é alocada uma tarefa por núcleo. Também são contabilizadas as horas de alocação de cada instância, de forma a permitir o cálculo do custo total de execução.

4. Experimentos e Métricas para Análise

4.1. Parâmetros de Simulação

Configuração dos Sistemas: Com o objetivo de analisar cenários envolvendo diferentes níveis de contenção na grade, a estratégia utilizada é de, para um mesmo *workload*, variar o número de máquinas presentes na grade como um todo. Para tanto, foram consideradas duas dimensões para aumentar o número de máquinas na grade: aumentar o número de máquinas que cada *peer* possui ou aumentar o número de *peers* na grade. Foi realizada uma varredura de parâmetros na faixa de 5 a 50 em cada dimensão, com passo de 5 unidades. Devido à restrição de espaço, são apresentados apenas os cenários em que $|G|$ assume os valores em $\{10, 25, 50\}$ e $|S|$ em $\{10, 25, 35, 50\}$. Nas simulações, por simplicidade, com relação ao número de máquinas e também quanto à distribuição estatística da capacidade computacional das máquinas, os *peers* são considerados homogêneos.

Para definir a distribuição estatística da capacidade computacional das máquinas, foi realizada uma análise em um *site* representativo da comunidade OurGrid. Esse *site* agrega cerca de 50 máquinas do tipo *desktop*, utilizadas por alunos de graduação e pós-graduação, além de funcionários do laboratório. A Figura 1 apresenta a distribuição da capacidade computacional das máquinas. Com esses dados, foi calculada uma função empírica de distribuição acumulada e a capacidade computacional de cada máquina simulada foi definida com base nesta função.

Para o ambiente de nuvem de instâncias *spot*, foi considerado um limite $L = 100$, o mesmo praticado pelo serviço *Amazon EC2*. Para a oscilação dos *spot-prices* Π_i, foram utilizados os históricos de preço informados pelo *AWS*, o que constrói toda a série temporal O, sendo uma para cada tipo de instância apresentada na Tabela 1. Cada histórico tem duração de pouco mais de 1 ano (desde dezembro de 2009, quando o serviço foi lançado) e a cada execução é escolhido aleatoriamente um instante do *trace* como instante inicial. Como usuários da nuvem Υ, aos quais é aplicado o limit L, são extraídos todos os usuários que submetem algum *job* para o sistema ($J \in W \Rightarrow user(J) \in \Upsilon$). Como o interesse do presente trabalho na infraestrutura de nuvem é verificar quanto o

Tabela 2. Distribuições do *Workload* e da Volatilidade das Máquinas[1]

(a) Modelo de Iosup

Usuário	IAT[2]	Ciclo Diário	Tam. do Job	ART[3]	RTV[4]
Z(1.31,368)	W(4.25,7.86)	W(1.79,24.16)	W(1.76,2.11)	N(2.73,6.1)	W(2.05,12.25)

(b) Tempo de Execução

Ibercivis
L(7.33,0.74)

(c) Volatidade das Máquinas

Disponibilidade	Indisponibilidade
L(7.95, 2.12)	L(7.24, 1.04)

[1] N, L, W, Z referem-se às distribuições Normal, Lognormal, Weibull e Zipf. O segundo parâmetro da Zipf é o número de usuários – [2] Tempo entre chegadas (IAT: Inter-Arrival Time) – [3] Tempo médio de execução (ART: average task runtime) – [4] Variância do ART (RTV: Runtime Variability)

limite L afeta o desempenho de aplicações *BoT* quando comparado com o desempenho na grade, foi considerado que os lances presentes em B são altos o suficiente para superar o *spot-price* no momento de submissão e até que a última tarefa do *job* seja concluída. Foram realizadas simulações para todas as instâncias presentes na Tabela 1, porém, em cada execução, o escalonamento é feito para apenas um tipo de instância, de forma que não são permitidas interações entre instâncias diferentes para a execução do *workload*.

Padrão de Disponibilidade das Máquinas: Foi realizada uma análise do padrão de disponibilidade das máquinas da comunidade OurGrid, referente a um intervalo de 4 meses, originados de 4 *sites* (englobando em torno de 150 máquinas). O objetivo foi encontrar uma função de distribuição de probabilidades 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 (*MLE*). 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*) para um nível de significância $\alpha = 0.05$. As distribuições escolhidas estão na Tabela 2(c).

***Workload* dos Usuários:** Foi realizada uma adaptação no modelo para *BoT* proposto em [Iosup et al. 2008b] e apresentado na Tabela 2(a). São consideradas as características inter-*job* (usuário de origem e padrão de chegada) e as intra-*job* (quantidade de tarefas e a distribuição estatística dos tempos de execução). Os aspectos inter-*jobs* são seguidos integralmente, no entanto, foi necessário realizar um tratamento adicional nos aspectos intra-*job*, de forma a adaptar os *jobs* gerados às características de grades oportunistas. Como o modelo original teve como base *traces* de grades de serviço (que utilizam recursos dedicados) as características intra-*jobs* diferenciam-se bastante de aplicações *BoT* típicas de grades oportunistas (*jobs* grandes e tarefas pequenas [Kondo et al. 2007]).

Ao contrário de *workloads* de grades de serviço, que possuem repositórios de *traces* e que já foram estudados e caracterizados de forma abrangente [Iosup et al. 2008a], grades oportunistas ainda carecem de uma caracterização de seus *workloads* típicos e da disponibilização de uma quantidade razoável de *traces* reais, de modo a se obter tal caracterização. Para contornar essa carência, partiu-se da seguinte suposição: os *jobs* de grade de serviço podem ser "quebrados" de forma que o tempo de execução de suas tarefas alcancem um grão similar àqueles de *jobs* propícios de serem executados em grades oportunistas. Como consequência, obtêm-se tarefas menores e mais tarefas por *job*.

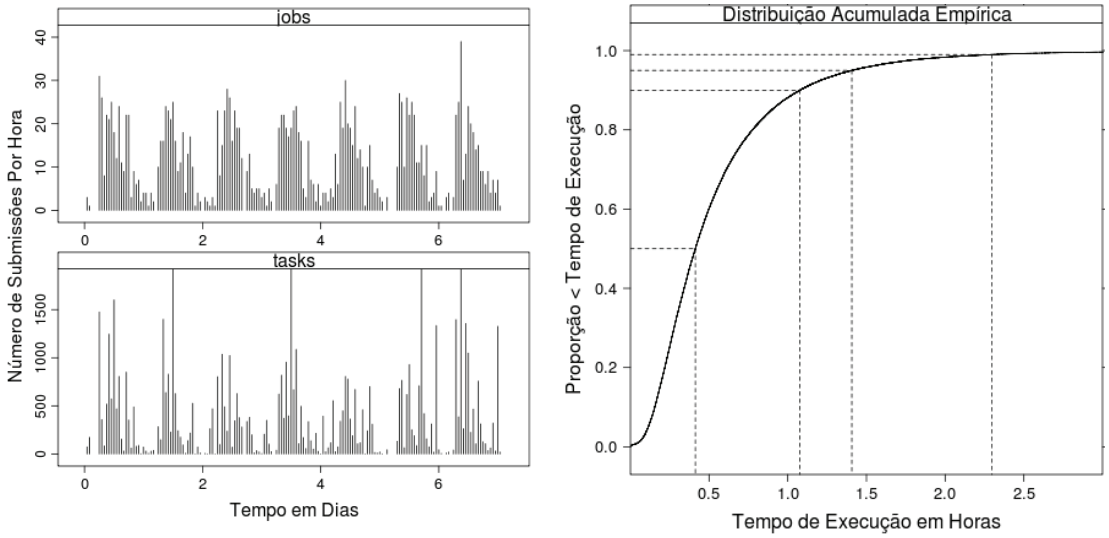

Figura 2. Visão geral do *workload* (a) e do tempo de execução das tarefas (b)

O segundo problema a ser atacado é como "quebrar" o *job* original. Para tanto, foi utilizado um *trace* da grade oportunista *ibercivis*[6], cuja distribuição estimada para o tempo de execução das tarefas se encontra na Tabela 2(b) e foi obtida seguindo a mesma metodologia utilizada para o caso do padrão de disponibilidade das máquinas. Então, para cada tempo de execução gerado a partir do modelo de Iosup, realiza-se a divisão de acordo com valores amostrados da distribuição da Tabela 2(b).

A Figura 2(a) ilustra um exemplo de *workload* gerado, sendo possível perceber o padrão do ciclo diário de submissão. A Figura 2(b) mostra como os tempos de execução ficam estatisticamente distribuídos, como resultado das alterações descritas anteriormente. Ademais, todos os *workloads* gerados são referentes a um período de 7 dias.

4.2. Métricas de Comparação

Em cada simulação, são registrados os *makespans* de cada *job*, tanto para a execução na grade quanto na nuvem. No caso da nuvem, também é computado o custo de execução de cada *job*. 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: $D^s = \frac{\sum_{j=1}^{|W^s|} makespan^C(W_j^s)}{\sum_{j=1}^{|W^s|} makespan^G(W_j^s)}$

Também é calculada a média para essa métrica ($\overline{D}$), de tal forma a representar o bem-estar geral que a grade oferece a seus participantes: $\overline{D} = \frac{1}{|S|} \sum_{s=1}^{|S|} D^s$.

Ambas as métricas (D^s e $\overline{D}$), tratadas simplesmente como D, flutuam em torno de uma linha de indiferença ($y = 1$). A métrica D deve ser interpretada com base nesta linha,

[6]Plataforma de Computação Cidadã, http://www.ibercivis.es/

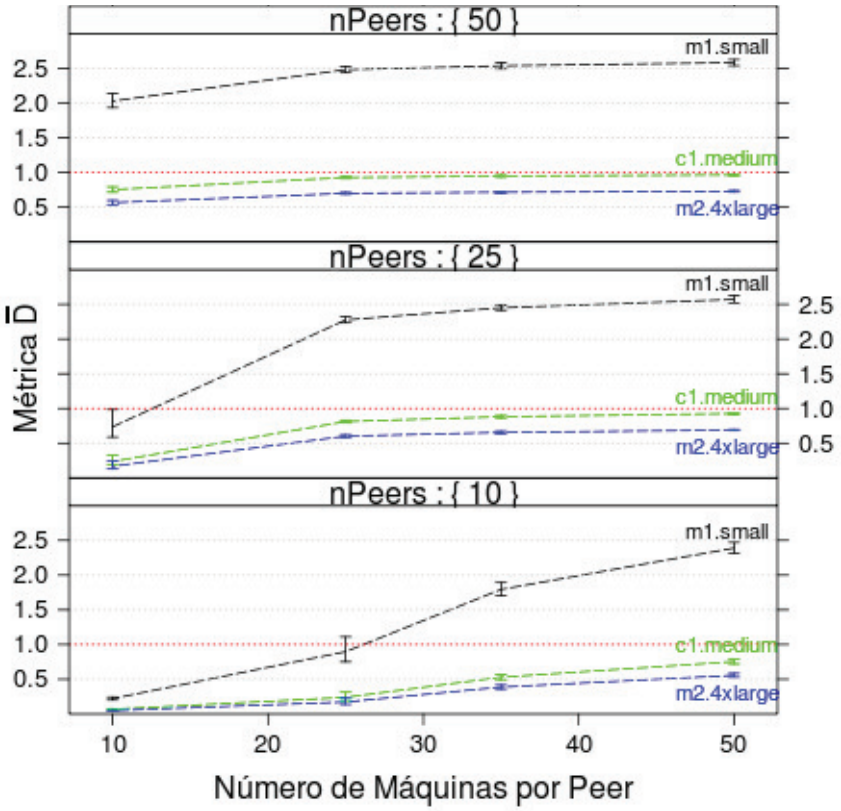

Figura 3. Desempenho Grade Vs. Instâncias *spot*

resultando em três casos que guiam a decisão a respeito de qual plataforma fornece, no geral, o melhor desempenho: C, se $D < 1$; G, se $D > 1$. Quando $D = 1$, outro critério deve ser utilizado para o desempate. Também se utiliza o custo médio por tarefa (ACT), permitindo a comparação de custo de execução em diferentes tipos de instâncias *spot*: $ACT = \frac{1}{\sum_{j=1}^{|W|} |W_j|} \sum_{j=1}^{|W|} \sum_{t=1}^{|W_j|} cost(W_j, t)$

5. Resultados e Discussão

Para cada combinação da variação dos fatores ($|G|$, $|S|$ e tipo de instância *spot*), foram realizadas 30 simulações, de acordo com a configuração estatística da Seção 4, e os valores nos gráficos estão em um intervalo de confiança de 95%, com erro máximo de 5%.

A Figura 3 apresenta os resultados da execução de um mesmo *workload* tanto em uma grade entre pares quanto na nuvem de instâncias *spot*. Os pontos do gráfico são referentes à comparação com três tipos de instâncias *spot* (*m1.small, c1.medium e m2.4xlarge*). Estas instâncias são representativas, pois ilustram os dois extremos e também um valor intermediário para a comparação. Como esperado, ao passo em que o tamanho da grade aumenta, e consequentemente diminui a contenção do sistema, a métrica $\overline{D}$ se estabiliza acima da linha de indiferença para a instância *m1.small* e um pouco abaixo para a instância *m2.4xlarge*. Por sua vez, a instância *c1.medium* apresenta desempenho muito parecido com o da grade para os cenários em que a contenção não é muito alta. O péssimo resultado apresentado pela instância *m1.small* é explicado pela baixa capacidade computacional de tais instâncias quando comparadas com os *desktops* típicos da grade, visto que contam com apenas 1 *ECU* (vide Tabela 1 e Figura 1). No outro

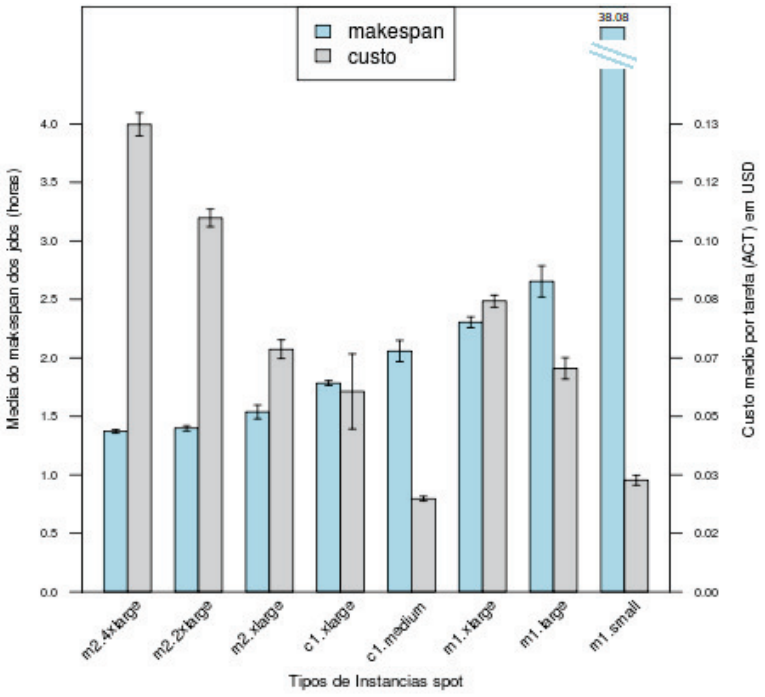

Figura 4. Makespan e Custo para as instância *spot*

extremo, o bom resultado apresentado pela instância *m2.4xlarge* se explica por sua maior capacidade de computação (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.

É apresentada, na Figura 4, a contrapartida financeira que o usuário precisa realizar para obter o desempenho presente na Figura 3. De imediato, observa-se que a instância *m1.small* não é uma boa opção, visto que, embora apresente um custo por tarefa competitivo, apresenta um desempenho sofrível. Já a instância *m2.4xlarge*, como visto anteriomente, apresenta o melhor desempenho, no entanto, é a solução mais cara. Mais uma vez, destaca-se a instância *c1.medium*, pois apresenta um desempenho aproximadamente 40% inferior em relação à instância *m2.4xlarge*, mas, em compensação, obteve o menor custo por tarefa, 5 vezes inferior ao obtido na *m2.4xlarge*. As instâncias *m2.xlarge* e *c1.xlarge* apresentam bom desempenho, mas custam, aproximadamente, 150% e 90% a mais que a *c1.medium*, por um desempenho apenas 25% superior. Portanto, a instância *c1.medium* é a melhor opção para servir de referência para estimar o valor de uma grade P2P visto que: *1.* os desempenhos apresentados são parecidos ($\overline{D}$ próxima à linha de indiferença) e *2.* representa a solução mais barata (não superestima o valor da grade).

Para estimar o valor da grade (V_g), será utilizada a Relação 1, para a qual M_g e M_c são a soma dos *makespans* dos *jobs* quando executados na grade e na nuvem, respectivamente, e C_c o custo total de execução de W na nuvem. A relação se baseia na ideia intuitiva de que quão menor o *makespan* obtido em uma plataforma, maior será o valor desta para o usuário, ou seja, o valor é inversamente proporcional ao *makespan*. Com o objetivo de estimar monetariamente o valor da grade, recorre-se ao aspecto custo de execução na nuvem, visto que esse custo se refere à execução do mesmo *workload*. 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*

Tabela 3. Consumo de energia (em Watts) do SETI@Home

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

c^t Consumo em carga total do ***SETI@Home*** – c^o Consumo do sistema quando ocioso – c^s Sobrecarga de consumo ($c^t - c^o$)

obtido na nuvem para essa estimativa de valor da grade. Caso os *makespans* na nuvem e na grade fossem idênticos, bastaria-nos mapear diretamente o valor da grade para o custo de execução na nuvem. Como não é esse o caso, é suficiente perceber que o valor da grade depende da relação entre os *makespans* obtidos na grade e na nuvem.

$$V_g = \frac{M_c}{M_g} \cdot C_c \tag{1}$$

Para o cenário com 50 *peers*, cada um dispondo de 25 máquinas, tem-se:

$$V_g = \frac{7.824.652}{7.375.565} \times 1.600 = 1.697,42\ USD$$

Para verificar a efetividade do valor da grade, deve-se considerar, também, o seu custo de manutenção. Para ser, de fato, uma solução efetiva, deve-se ter:

$$V_g > C_g \tag{2}$$

Nas grades oportunistas, em que os recursos não são adquiridos tendo a grade em mente, a ideia de custo não se dá de forma direta, tais como em outras plataformas [Opitz et al. 2008]. Na prática, a principal preocupação em relação ao custo extra reside no consumo de energia, visto que, se não fosse a grade, a máquina seria, após o uso, solenemente desligada pelo usuário local. Deste modo, para estimar o custo da grade, serão considerados os gastos extras com eletricidade. Para uma análise que se aproxime da realidade, dois referenciais de consumo devem ser considerados: um para os períodos de completa ociosidade e outro para operação em carga máxima. A Tabela 3 apresenta valores referentes a vários sistemas submetidos à execução do SETI@home.

Devem ser consideradas duas situações para o crédito de consumo à grade: ***(i)*** - quando o recurso está sendo utilizado apenas pela grade (p. ex, nos períodos noturnos) e ***(ii)*** - 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 creditados (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*. Para evitar problemas com conversão de moeda, será considerado o custo médio de 5 centavos de dólar, por *kWh*, cobrado de instituição educacionais nos Estados Unidos [Kondo et al. 2009].

Após todas as considerações, chega-se ao modelo simplificado C_g para o cálculo do custo extra com energia durante d dias, para o caso de máquinas e sites homogêneos.

$$C_g = d \cdot np \cdot nm \cdot \tfrac{k}{1000} \cdot \underbrace{\{\overbrace{h \cdot r \cdot u \cdot c^s}^{\textit{Devido à situação (ii)}} + \overbrace{(24-h) \cdot [u \cdot c^t + (1-u) \cdot c^o]}^{\textit{Devido à situação (i)}}\}}_{\text{Consumo diário por máquina (em Watts)}}$$

No modelo, np é o número de *peers* e nm o número de máquinas por *peer*; c^s, c^t e c^o denotam, respectivamente, os consumos de sobrecarga, total e de ociosidade da máquina homogênea (vide Tabela 3), k é o preço de 1 *kWh*, r é a taxa de ociosidade das máquinas em horário normal de trabalho (h horas por dia) e u é a utilização do sistema.

Consultando a Tabela 3, pode-se perceber dois extremos de consumo de energia, com uma máquina mais eficiente ($c^t_{min} = 115$, $c^s_{min} = 38$ e $c^o_{min} = 77$) do ponto de vista energético e uma extremamente ineficiente ($c^t_{max} = 198$, $c^s_{max} = 78$ e $c^o_{max} = 120$). Dessa forma, aplicando o modelo C_g podemos encontrar dois valores limites, C_g^{min} e C_g^{max}, para a estimativa de custo. Assumindo que os recursos são utilizados durante um total de 10 horas diárias por seus usuários locais ($h = 10$), considerando que a taxa de utilização para $np = 50$ e $nm = 25$ é de 17,5% ($u = 0,175$), que a ociosidade das máquinas em horário comercial é de cerca de 70% ($r = 0,7$) [Kondo et al. 2007], $k = 0,05\ USD$ e sabendo que o *workload* cobre um período de 7 dias ($d = 7$), instanciando C_g para os dois extremos, temos: $C_g^{min} = 532,7219\ USD$ e $C_g^{max} = 860,4094\ USD$.

Como se obteve $V_g = 1.697,42$, maior que C_g^{max}, conclui-se, considerando todas as simplificações e aproximações, que a grade em questão é uma solução de baixo custo para a execução de aplicações *BoT*. A Tabela 4 resume as considerações para todas as instâncias de grades analisadas. Os valores estão ordenados de acordo com a última coluna, que representa a efetividade da grade como solução de baixo custo no pior caso.

A métrica percentual V_g^{eff} ($V_g^{eff} = \frac{V_g - C_g}{V_g}$) indica quanto e como 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. Vale destacar dois casos extremos que resultam em um baixo valor efetivo: **1.** quando $np = 10$ e $nm = 10$, devido a um desempenho ruim (*makespans* muito longos em função da altíssima contenção), embora tenha um baixo custo de manutenção. **2.** quando $np = 50$ e $nm = 50$, devido a uma sub-utilização das máquinas, que permanecem ociosas (desperdiçando energia) a maior parte do tempo, embora apresente um alto valor para V_g.

V_g^{eff} também pode ser utilizada para indicar em quanto porcento o preço praticado pela *Amazon* deveria ser reduzido para que a grade não fosse a melhor opção ou, de outra forma, quão mais potente deveria ser a instância *c1.medium*. Em particular, para as últimas 6 configurações, observa-se que a margem vai, aproximadamente, de 50% a 70%.

6. Trabalhos Relacionados

Em [Kondo et al. 2009] os autores analisam o custo-benefício da computação na nuvem e de grades de *desktops* baseadas em computação voluntária, destacando os *trade-offs* de desempenho entre as duas plataformas. O trabalho apresentado em [Oprescu and Kielmann 2010] envolve o escalonamento com restrição de orçamento para aplicações *BoT* em múltiplos provedores de *IaaS* com diferentes desempenho de *CPU* e custo, de forma a minimizar o tempo de término enquanto respeita o valor máximo a ser gasto. O foco principal em [Yi et al. 2010] é a investigação de como *checkpointing* pode ser utilizado para reduzir custos de execução de aplicações na nuvem *spot* e, ainda assim, manter alta confiabilidade. Por sua vez, em [Andrzejak et al. 2010] é proposto um modelo

Tabela 4. Considerações de Custo das Duas Opções de Execução

np	nm	C_g (USD)		V_g (USD)	V_g^{eff} (%)	
		min	max		max	min
10	10	65,58	115,96	14,84	-341,91	-681,40
25	10	146,55	254,21	268,99	45,52	5,49
50	50	1.001,20	1.588,96	1.768,67	43,39	10,16
10	25	150,67	262,66	334,10	54,90	21,38
50	35	718,44	1.148,39	1.745,46	58,84	34,21
50	25	532,72	860,41	1.697,42	68,62	49,31
25	50	521,20	836,76	1.688,85	69,14	50,45
10	35	188,75	322,18	758,40	75,11	57,52
25	35	380,97	618,84	1.584,13	75,95	60,94
25	25	287,66	473,92	1.394,90	79,38	66,02
10	50	245,07	409,81	1.243,01	80,28	67,03
50	10	250,24	420,41	1.287,87	80,57	67,36

probabilístico para otimização de custo, desempenho e confiabilidade para a execução de aplicações *BoT* sobre a nuvem *spot*. De acordo com o levantamento bibliográfico realizado, não há nenhum trabalho que considere os efeitos do limite real na elasticidade da nuvem e a consequente implicação no aumento do *makespan* de aplicações *BoT*, fator que coloca as grades oportunistas como opções adequadas para a execução de tais aplicações.

7. Conclusão e Trabalhos Futuros

Os resultados de simulação mostram que, devido ao limite real na elasticidade oferecida pelo provedor de IaaS, mesmo para grades relativamente pequenas e para os cenários em que a contenção de recursos não é muito alta, uma grade entre pares se mostra uma alternativa mais vantajosa, considerando as métricas custo e *makespan*. A essência do resultado apresentado está, com relação ao custo, na característica oportunista da grade, e, quanto ao desempenho, na impossibilidade dos atuais provedores de IaaS de oferecer elasticidade ilimitada para todos os usuários que simultaneamente solicitam seus serviços.

Como trabalho futuro, pretende-se melhorar as estimativas de custo de manutenção de grade entre pares, ao adicionar outros fatores que influenciam no custo, a exemplo do impacto do aumento de utilização na vida útil dos equipamentos. Também, pretende-se utilizar técnicas mais sofisticadas para o escalonamento de tarefas na nuvem, de forma a permitir uma redução no custo de execução.

Referências

Andrade, N., Brasileiro, F. V., Cirne, W., and Mowbray, M. (2007). Automatic Grid Assembly by Promoting Collaboration in Peer-to-Peer Grids. *Jour. of Par. and Dist. Computing*, 67:957–966.

Andrzejak, A., Kondo, D., and Yi, S. (2010). Decision Model for Cloud Computing under SLA Constraints. In *Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on*, pages 257 –266.

Anglano, C., Canonico, M., Guazzone, M., Botta, M., Rabellino, S., Arena, S., and Girardi, G. (2008). Peer-to-Peer Desktop Grids in the Real World: The ShareGrid Project. *Cluster Computing and the Grid, IEEE International Symposium on*, 0:609–614.

Araujo, E., Cirne, W., Wagner, G., Oliveira, N., Souza, E. P., Galvao, C. O., and Martins, E. S. (2005). The SegHidro Experience: Using the Grid to Empower a Hydro-Meteorological Scientific Network. In *First Int. Conf. on e-Science and Grid Computing*. IEEE Computer Society.

Armbrust, M., Fox, A., and Griffith, R. (2009). Above the Clouds: A Berkeley View of Cloud Computing, Tech. Rep. EECS-2009-28, EECS Dept, Univ. of California, Berkeley.

Cirne, W., Brasileiro, F., Andrade, N., Costa, L., Andrade, A., Novaes, R., and Mowbray, M. (2006). Labs of the World, Unite!!! *Journal of Grid Computing*, 4(3):225–246.

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

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

Fields, S. (1993). Hunting for Wasted Computing Power, 1993 Research Sampler, University of Wisconsin-Madison, http://www.cs.wisc.edu/condor/doc/WiscIdea.html.

Iosup, A. and Epema, D. (2011). Grid Computing Workloads: Bags of Tasks, Workflows, Pilots, and Others. *IEEE Internet Computing*, 15:19–26.

Iosup, A., Li, H., Jan, M., Anoep, S., Dumitrescu, C., Wolters, L., and Epema, D. (2008a). The Grid Workloads Archive. *Future Generation Computer Systems*, 24(7):672–686.

Iosup, A., Sonmez, O., Anoep, S., and Epema, D. (2008b). The Performance of Bags-of-Tasks in Large-scale Distributed Systems. In *Proceedings of the 17th Int. Symp. on High Performance Distributed Computing*, HPDC '08, pages 97–108, New York, NY, USA. ACM.

Iosup, R., Jan, M., Sonmez, O., and Epema, D. (2007). The Characteristics and Performance of Groups of Jobs in Grids. In *In Euro-Par, volume 4641 of LNCS*. Springer-Verlag.

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

Kondo, D., Fedak, G., Cappello, F., Chien, A. A., and Casanova, H. (2007). Characterizing Resource Availability in Enterprise Desktop Grids. *Future Gener. Comput. Syst.*, 23:888–903.

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

Opitz, A., König, H., and Szamlewska, S. (2008). What Does Grid Computing Cost? *Journal of Grid Computing*, 6:385–397. 10.1007/s10723-008-9098-8.

Oprescu, A. M. and Kielmann, T. (2010). Bag-of-Tasks Scheduling under Time and Budget Constraints. *IEEE Int. Conf. and Workshops on Cloud Comp. Tech. and Science (CloudCom 2010)*.

UCB (2011a). BOINC Statistics, http://www.allprojectstats.com/po.php?projekt=0.

UCB (2011b). SETI@Home, University of Berkeley, California, http://setiathome.berkeley.edu.

Yi, S., Kondo, D., and Andrzejak, A. (2010). 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.