Análise de Estratégias de Computação Verde em Grades Computacionais Oportunistas

Lesandro Ponciano[1], Jaindson Santana[1], Marcus Carvalho[1], Matheus Gaudencio[1], Francisco Brasileiro[1]

[1]Laboratório de Sistemas Distribuídos
Universidade Federal de Campina Grande (UFCG)
Av. Aprígio Veloso, 882 – Bloco CO – Campina Grande – PB – Brazil

{lesandrop, jaindson, marcus, matheusgr}@lsd.ufcg.edu.br,

fubica@dsc.ufcg.edu.br

Abstract. *Nowadays, opportunistic grids are getting more and more popular. The energy efficiency of those grids is also an increasing concern. This paper evaluates two green computing strategies to reduce energy consumption in opportunistic grid resources, namely: standby and hibernate. Both techniques are used when a resource is idle, i.e., available to the grid, but there is no work to be processed. We simulate an opportunistic grid that uses both strategies and also a scenario without a green computing strategy. As expected, both techniques increased the response time (makespan) of the jobs executed, but improved energy savings when compared with the scenario that does not use a green computing strategy. However, the standby approach resulted in greater savings and in a smaller impact on the application makespan, being a better strategy to be used in such grids.*

Resumo. *Grades oportunistas são sistemas computacionais que têm sido amplamente utilizados para execução de aplicações científicas. Nos últimos anos tem aumentado a preocupação com a eficiência energética dessas grades. Este trabalho avalia o impacto do uso de estratégias para diminuir o consumo de energia nessas grades. Duas estratégias são analisadas: standby e hibernate. Elas são utilizadas quando as máquinas estão ociosas e, portanto, disponíveis para a grade, mas não têm nenhuma tarefa para executar. Nossa avaliação utiliza um modelo simulado para avaliar o custo em termos de aumento do tempo de resposta das aplicações (makespan) e o benefício associado à redução no consumo de energia. Como esperado, ambas as estratégias impactam o makespan dos jobs executados, mas reduzem o gasto da infraestrutura com energia. Entretanto, a estratégia standby resultou em uma maior economia e em um menor impacto no tempo de resposta da aplicação, sendo, portanto, a estratégia mais apropriada para ser usada em grades oportunistas.*

1. Introdução

A evolução dos sistemas computacionais tem sido marcada pela busca por mais poder computacional a qualquer custo [Economist 2007]. No entanto, com o aumento do poder computacional, aumentou-se também o consumo de energia e, por consequência, a

emissão de dióxido de carbono (CO_2) no meio ambiente. Os problemas ambientais gerados pelo aumento da emissão de CO_2 e o custo financeiro ocasionado pelo consumo de energia têm impulsionado estudos que visam o desenvolvimento de mecanismos e tecnologias que façam uso eficiente da energia. Esses mecanismos e tecnologias são denominados computação verde ou *green computing* [Williams and Curtis 2008].

Computação verde tem influenciado diversas áreas da computação, como, por exemplo, o projeto de *hardware*, gerência de *datacenters* e grades de serviço [Meisner et al. 2009, Przybyla and Pegah 2007]. Este trabalho analisa a aplicação de estratégias de computação verde na gerência de recursos e escalonamento de tarefas em grades oportunistas. Os recursos pertencentes a este tipo de grade são utilizados de forma oportunista: se o computador não estiver sendo utilizado por um usuário, o poder computacional disponível, porém ocioso, pode ser utilizado pelas tarefas submetidas à grade.

Quando um computador está disponível para a grade, mas a grade não o utiliza, ele fica em estado ocioso (do inglês *idle*). Ao longo do tempo, a permanência dos computadores nesse estado caracteriza ciclos de ociosidade na grade e desperdício de energia. Esses ciclos de ociosidade são comuns em grandes computacionais [Iosup et al. 2006, Kondo et al. 2007]. O estado de ociosidade apresenta menor consumo de energia do que quando a máquina está executando alguma aplicação, porém esse consumo de energia é ainda significativo.

Duas estratégias para reduzir o consumo de energia nesses ciclos de ociosidade são: *standby* e *hibernate*. *Standby*, também conhecida por Suspensão para a Memória de Acesso Aleatório (RAM, do inglês *Random Access Memory*), consiste em manter a memória RAM ativa e reduzir a atividade do disco rígido e do processador. *Hibernate*, também conhecida por Suspensão para o Disco, consiste em salvar o estado da memória RAM no disco rígido, reduzir o uso de energia da RAM, do disco rígido e do processador. Essas estratégias são definidas pelo padrão de configuração avançada e interface de energia (ACPI, do inglês *Advanced Configuration and Power Interface* [Corporation et al. 2010]) - padrão aberto que unifica a configuração dos dispositivos e a interface de gerência de energia pelos sistemas operacionais.

Um fator importante nessas estratégias é que diversos dispositivos são desativados para reduzir o consumo de energia, mas o computador pode ser reativado eletronicamente através de um comando a algum dispositivo mantido em estado de espera, como: teclado, *modem*, LAN (conhecido como *Wake-on-LAN* - WOL) ou outro dispositivo Barramento Serial Universal (USB, do inglês *Universal Serial Bus*), que, ao ser acionado, coloca todo o computador novamente em atividade. Colocar e retirar um computador do estado *standby* é mais rápido que do estado *hibernate*, uma vez que não é preciso realizar a cópia dos dados da memória RAM para o disco rígido. Por outro lado, no estado *standby* tem-se maior consumo de energia do que no estado *hibernate*, porque a memória RAM permanece energizada.

Para utilização dessas estratégias em uma grade computacional oportunista, é necessário avaliar a redução do consumo de energia e o custo associado em termos do aumento no tempo de resposta dos *jobs* (também conhecido por *makespan* do *job*), ocasionado pelo tempo gasto para colocar e retirar o computador de tais estados. Isso porque,

quando uma nova tarefa é submetida à grade, ela precisará esperar o tempo necessário para que o computador seja colocado de volta em um estado completamente operacional. Colocar o computador no estado de baixo consumo de energia pode implicar na diminuição do custo de energia, mas o tempo de espera necessário para que ela se torne disponível pode impactar negativamente no desempenho da grade com o aumento do tempo de resposta das aplicações.

Este artigo busca avaliar como as estratégias *standby* e *hibernate* impactam na economia de energia e no tempo de resposta das tarefas em uma grade oportunista, identificando cenários em que essas estratégias minimizam o consumo de energia e avaliando o impacto no tempo de resposta. O artigo está organizado da seguinte forma; na Seção 2 é apresentado o estado da arte da aplicação de estratégias de redução do consumo de energia elétrica em sistemas computacionais; na Seção 3 é apresentado o modelo de simulação utilizado neste trabalho; em seguida, na Seção 4 , é descrito o estudo de caso no qual foi utilizada uma carga de trabalho de uma grade oportunista. Na Seção 5 são apresentados e analisados os resultados do estudo de caso. Por fim, na Seção 6 são descritas as conclusões e os trabalhos futuros.

2. Estado da Arte

Nos últimos anos, diversas abordagens têm sido propostas para analisar e reduzir o consumo de energia em sistemas computacionais, através de melhores projetos do *hardware* e do software desses sistemas, além do ambiente onde os mesmos estão inseridos. Em hardware, tem-se buscado desenvolver computadores mais econômicos em termos de consumo de energia. Já em software, um caminho é o desenvolvimento de softwares mais otimizados a fim de evitar processamento excessivo [Barroso and Hölzle 2007]. Em relação ao ambiente de implantação, toda a infraestrutura de suporte é considerada, incluindo principalmente o sistema de refrigeração das máquinas.

Esta preocupação com o consumo de energia não é nova. As indústrias de dispositivos móveis e *laptops* têm desenvolvido muitas pesquisas nesse sentido. No entanto, a sua motivação é o fato da energia ser algo bastante escasso (uso de baterias recarregáveis com suporte limitado, por exemplo). Além disso, as técnicas utilizam, entre outras coisas, padrões de comportamento do usuário que nem sempre se aplicam a todos os sistemas, o que invalidaria o uso dessas técnicas em outros cenários. Mesmo com a impossibilidade de reuso destas técnicas, tem-se observado uma melhora na relação entre o consumo de energia e utilização (carga de trabalho) dos computadores ao longo dos tempos. No entanto, esta relação ainda pode ser melhorada [Barroso and Hölzle 2007].

Um outro aspecto que pode ser explorado visando a diminuição do consumo de energia é a mudança do estado dos dispositivos dependendo da sua carga de uso. Alguns trabalhos já foram realizados com o objetivo de mensurar os gastos e abordar diferentes estratégias. Talebi et al. [Talebi and Way 2009] reporta práticas que podem ser usadas em salas de aula e laboratórios de pesquisa. São destacadas as práticas: *(i)* não utilização de protetor de tela que possui animações gráficas; *(ii)* colocar o monitor em modo *sleep*, no qual ele passa a consumir menos energia (reativação, por exemplo, com o movimentar do mouse); *(iii)* colocar o disco rígido em modo *sleep*, que implica na redução dos movimentos do disco rígido quando o computador está ocioso; *(iv) standby*, que consiste em manter os dados na memória RAM e reduzir a atividade dos outros componentes; *(v)*

hibernate, em que os dados são armazenados no disco rígido e os demais componentes são desligados, de modo que ao ser ligado novamente, os dados sejam recuperados do disco rígido e o computador retorne ao estado em que estava.

Um projeto da Escola de Educação da Universidade de Indiana [Indiana University 2009] visa implantar um mecanismo para colocar computadores *desktops* em modo *sleep* quando não estiverem em uso. A proposta é colocar os computadores neste estado após 2 horas e 15 minutos de inatividade contínua. Em um projeto piloto, obteve-se redução no consumo de energia em 48,3% para um *cluster* de 11 computadores *desktops* e em 30,9% na ala de escritório. Essa redução equivale a economia de até U$ 500.000,00 por ano para a universidade. Entretanto, o foco do trabalho é a redução do consumo utilizando o estado *sleep*, sem se preocupar com o tempo e o consumo de energia necessário para colocar e retirar os computadores desse estado. Em uma máquina oportunista onde esse tipo de operação de entrada e saída da máquina da grade pode ser frequente, não é claro que essa estratégia possa trazer economias similares.

Há também estudos que visam investigar estratégias de escalonamento ciente do consumo de energia. Essas estratégias visam minimizar o consumo de energia com o mínimo de impacto no desempenho das aplicações. Sharma e Aggarwal [Sharma and Aggarwal 2009] analisam um escalonamento ciente do consumo de energia em grades *desktop*. No entanto, eles analisam aplicações que fazem uso intensivo de memória, além disso, os *desktops* estão sempre disponíveis para a grade. De outro modo, Lammie, Brenner and Thain [Lammie et al. 2009] analisam estratégias de escalonamento de cargas de trabalho de grades computacionais em *clusters* multicores. O objetivo do escalonamento também é minimizar o consumo de energia e maximizar o desempenho, no entanto, apenas o uso de CPU foi considerado. São propostas três técnicas para atingir este objetivo: desativar as máquinas que se encontram subutilizadas; prover um escalonamento inteligente das tarefas de modo a minimizar o número de máquinas ativas; e fazer um dimensionamento dinâmico da frequência, de modo a ativar e a desativar as máquinas de acordo com a demanda. Ambos os estudos indicam redução significativa no consumo de energia e pouco impacto no desempenho.

Zong et al. [Zong et al. 2007] propõe um *framework* para simulação e avaliação da eficiência energética de algoritmos de escalonamento de tarefas que fazem uso intensivo de dados. Para avaliar o *framework*, utilizou-se uma política de escalonamento que consiste em escalonar tarefas para nodos que economizam mais energia (energeticamente mais eficientes). Assim, o escalonador dá prioridade a escalonar tarefas que fazem uso intensivo de dados a recursos mais eficientes para este tipo de tarefas, podendo retirar demais tipos de tarefas desses recursos, reservar recursos para alocar a tipos específicos de tarefas, e dá preferência por alocar tarefas, com dependências entre si, em um mesmo recurso a fim de evitar consumo de energia com *overhead* de comunicação. A grade utilizada considera total disponibilidade dos recursos e o escalonamento apenas de aplicações que fazem uso intensivo de dados, o que difere do ambiente e do tipo de tarefas submetidas a grades computacionais oportunistas.

O presente trabalho, diferente dos demais apresentados nesta seção, visa analisar estratégias de computação verde no contexto de grades computacionais oportunistas. Essas grades apresentam um fator não investigado pelos demais trabalhos em gra-

des computacionais que é aplicar essas estratégias em um cenário onde a disponibilidade das máquinas varia ao longo do tempo. A semelhança com os demais trabalhos está na utilização de estratégias de redução do consumo de energia durante os ciclos de ociosidade.

3. Modelagem de computação verde em Grades Oportunistas

Nossa análise de estratégias de computação verde em grades oportunistas considera tanto mecanismos para a redução no consumo de energia, como a colocação das máquinas em *standby* ou *hibernate* durante os ciclos de ociosidade, quanto mecanismos para minimizar o impacto no tempo de resposta das aplicações, como o escalonamento de tarefas para as máquinas de maior poder de processamento. Por ser uma grade oportunista, as máquinas podem ficar indisponíveis para a grade caso o usuário a esteja usando, por exemplo. Nosso modelo de grade considera a implementação de *checkpoint*, de modo que, se uma máquina tornar-se indisponível durante a execução de determinada tarefa, esta tarefa é interrompida e submetida novamente quando houver uma máquina disponível. Todo o processamento já realizado é aproveitado.

Ao longo do tempo, uma máquina pode estar em um de 4 estados possíveis (Figura 1): *(i)* Ocioso, caso não exista nenhuma estratégia de computação verde ativa; *(ii)* Grade, caso esteja executando alguma tarefa da grade; *(iii)* Usuário, caso o usuário a esteja utilizando, neste caso ela não está disponível para a grade; *(iv)* ou em um estado de computação verde, que pode ser uma das duas estratégias consideradas: *Standby* ou *Hibernate*. O estado ocioso apresenta maior consumo de energia que o estado de computação verde. Cada estratégia de computação verde apresenta um tempo de transição, que é o tempo que leva para ir e também para sair deste estado, e o custo de energia da máquina quando se encontra neste estado. No período de transição a máquina não pode ser utilizada por tarefas da grade, o que ocasiona um impacto no tempo de resposta das tarefas. Para realizar a transição a máquina realiza processamento. Este processamento representa um custo maior de energia que é equivalente ao custo de energia de uma máquina durante processamento normal. Um exemplo dos estados das máquinas na grade é apresentado na Figura 1. Para facilitar a compreensão, foram colocados nessa figura exemplos de valores típicos de consumo de energia e tempo de transição de estados. Note que o estado e as transições para o estado *Usuário* não são considerados no cálculo do custo da grade.

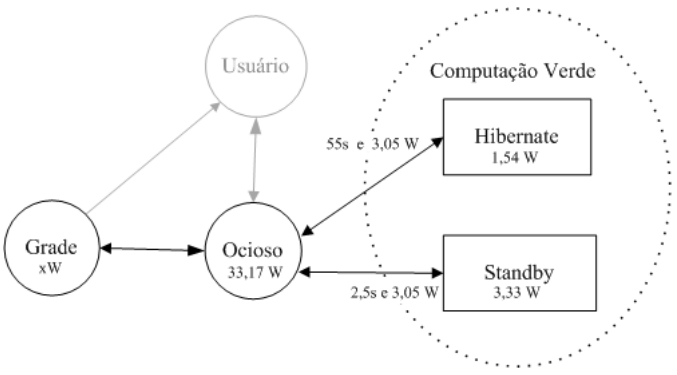


Figura 1. Estados das Máquinas

Para avaliar o consumo de energia da grade, somamos o consumo de energia de todas as máquinas durante o tempo do experimento. O consumo de energia de uma máquina é dado pela Equação 1, onde tem-se o tempo que a máquina permaneceu (t) e a potência em que operou (p) em cada uma das seguintes situações: (i) fazendo transição de estado (m); (ii) em estado de consumo de energia (e) ou (iii) executando uma tarefa na grade (g).

$$C = t_m \times p_m + t_e \times p_e + t_g \times p_g \quad (1)$$

Além do consumo de energia, também é importante medir o tempo de resposta dos *jobs*, que é o tempo decorrido entre a submissão e o término da execução de um *job*. Para avaliar o atraso no tempo de resposta com o uso das estratégias *standby* e *hibernate* em relação ao cenário em que nenhuma estratégia de computação verde é usada (ocioso), utilizou-se a medida *slowdown*. O *slowdown* é o tempo de resposta dos *jobs* com o uso de uma estratégia dividido pelo tempo de resposta dos *jobs* sem o uso de uma estratégia. Quando o *slowdown* é menor que 1, tem-se um ganho no tempo de resposta, de outro modo, quando o slowdown é maior que 1 ocorre um atraso.

4. Projeto do Estudo de Caso

Nossa avaliação do uso de estratégia de computação verde em grades oportunistas utiliza um modelo simulado para avaliar o custo em termos do aumento do tempo de resposta das aplicações e o benefício associado à redução no consumo de energia. Nesta seção, apresentamos o modelo simulado, os *traces* e os cenários avaliados. No nosso modelo, destacam-se os seguintes fatores: *(i)* variação na disponibilidade das máquinas ao longo do tempo; *(ii)* heterogeneidade das máquinas quanto ao poder de processamento e ao consumo de energia.

Para simular a demanda de *jobs* na grade utilizamos um *trace* da comunidade OurGrid[1]. Esse *trace* possui informações sobre o tempo de submissão e tempo de execução dos *jobs*. A escolha desse *trace* se deu porque atualmente essa grade encontra-se em produção e é constituída por *jobs* do tipo saco-de-tarefas (BoT, do inglês *bag-of-tasks*), que é o tipo de tarefa mais comum em grades oportunistas. A carga de trabalho possui 21.705 tarefas executadas entre 18 de dezembro de 2008 e 18 de novembro de 2009. Foram utilizados dados do tempo de submissão e do tempo de execução das tarefas. No *trace* há tarefas com diferentes tempos de execução, no entanto, existem algumas tarefas com tempo de execução muito grande, o que foge ao comportamento típico de tarefas submetidas a grades oportunistas. Para analisar melhor essas tarefas, calculou-se o 99-percentil e verificou-se que tarefas com mais de 41.367 segundos (27 tarefas) eram pouco frequentes e, portanto, foram eliminadas do *trace* utilizado como entrada para o simulador. O Exemplo 1(a) apresenta o modelo de *trace* de submissão de tarefas. O *trace* contém uma tarefa em cada linha. As tarefas são constituídas de um identificador que indica o código do *job* e o código da tarefa, um instante de submissão, e um valor estimado do tempo total de execução, respectivamente.

Como o ambiente de grade computacional avaliado é oportunista, as máquinas só estarão disponíveis para a grade quando estas estiverem ociosas, ou seja, quando não estiverem em uso pelos seus proprietários. Para representar a variação da disponibilidade

[1]Comunidade OurGrid, `http://www.ourgrid.org/`. [OurGrid Community 2010]

das máquinas ao longo do tempo, utilizou-se um *trace* de uma grade *desktop* que apresenta dados sobre o uso das máquinas, identificando os períodos em que as máquinas estão ociosas (em todos os *traces* utilizados neste trabalho, as informações referentes a tempo são medidas em segundos). O *trace* utilizado foi o DEUG [Kondo et al. 2007], que possui um período não contínuo de 1 mês, com informações sobre máquinas *desktop* da Universidade de París-Sud. Como as informações sobre disponibilidade nesse *trace* não formam um período contínuo, foi feito um tratamento dos dados para gerar um novo *trace* sem interrupções. O tratamento consistiu na concatenação dos dados em períodos sequenciais e ininterruptos, eliminando assim os períodos em que não se tem informações. Um exemplo do *trace* da disponibilidade das máquinas é apresentado no Exemplo 1(b). Em cada linha, tem-se uma máquina. Cada máquina é constituída, respectivamente, pelo instante de tempo em que ela se torna disponível para a grade e por quanto tempo ficou disponível.

Do *trace* DEUG também foram obtidas informações sobre o poder de processamento das máquina. Nesse *trace*, o poder de processamento é descrito pelo número de operações por segundo que a máquina pode executar. Há informações desse valor para as máquinas ao longo do tempo, e o valor varia em função da utilização da máquina no momento em que a medição foi realizada. Deste modo, utilizamos o maior desses valores para representar a capacidade de processamento de cada máquina (ver Exemplo 1(c)).

```
J1−1  1050208880   218
J2−1  1050208892   1107
J2−2  1050208892   723
```

(a) Demanda

```
V4−linux  1104904800   1636
V2−linux  1104904800   3611
V4−linux  1104914800   3666
```

(b) Disponibilidade

```
V2−linux  110700
V3−linux  177120
V4−linux  114810
```

(c) Processamento

Exemplo 1. Exemplos dos *traces* utilizados nas simulações

Para estimar a frequência de CPU das máquinas utilizamos a relação: 1,5 GHz equivalente a 110.700 operações por segundo (como apresentado em Kondo et al. [Kondo et al. 2007]). Cada segundo do tempo de execução das tarefas é convertido para 177.120 operações, o que equivale a um segundo de execução em um computador de 2.4 GHz; simplificação semelhante é apresentada em Lammie et al. [Lammie et al. 2009]. Na Figura 2 é apresentada a variação da frequência dos ciclos de CPU da amostra de máquinas utilizada nas simulações.

Como mencionado anteriormente, duas estratégias de computação verde foram avaliadas para este trabalho: *Standby* e *Hibernate*. Estas estratégias são comparadas com o estado ocioso, em que não são utilizadas estratégias de economia de energia. Para cada estratégia, são estabelecidos valores do consumo de energia e do tempo de transição entre os possíveis estados das máquinas. Esses valores são apresentados nos estados e nas transições na Figura 1. O consumo de energia varia de acordo com a frequência de CPU das máquinas, mas esse não é o único fator que impacta o consumo. Neste trabalho consideramos apenas aplicações que fazem uso intensivo de CPU e, assim, os valores de consumo de energia se basearam apenas neste fator. O valor de referência das máquinas foram obtidos com base na lista de máquinas avaliadas pela EnergyStar [Energy Star 2009]. Procuramos os valores das máquinas que possuíam uma mesma configuração geral do sis-

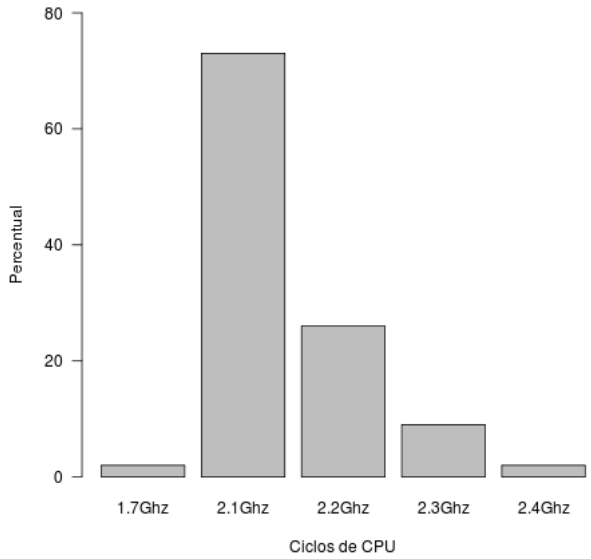


Figura 2. Frequência de CPU da amostra de máquinas utilizada nas simulações

tema, mas com diferentes frequências de CPU. Os tempos de transição de estados são os mesmos utilizados por Orgerie et al. [Orgerie et al. 2008].

O escalonador faz uso apenas das máquinas disponíveis e cada máquina pode executar apenas uma tarefa por vez. Quando uma máquina está disponível e não há atividade a ser realizada por ela, o simulador faz uso de uma estratégia de economia de energia. O recurso passa então por um período de transição. Nesse período, a potência considerada é de 200 W (não há uma medição exata deste valor, essa aproximação é apresentada por Orgerie et al. [Orgerie et al. 2008]) e após isso, passa para um estado em que o gasto de energia é dado pela estratégia utilizada. Assim, se uma determinada máquina está disponível e ociosa e for para o estado *hibernate*, ela ficará 55 segundos sem poder ser utilizada pela grade. Neste tempo, ela gastará a potência de 200 W, ou seja, um consumo de 3,05 Wh. Supondo que o recurso fique 1 hora nesse novo estado, ela terá gasto 1,54 Wh. Se uma tarefa for escalonada para esta máquina, ela precisará de 55 segundos (e novamente, 3,05 Wh) para poder se tornar utilizável. Só depois deste tempo de transição que a tarefa começará sua execução. O tempo de transição e consumo nos estados são apresentados na Tabela 1.

Tabela 1. Cenários Avaliados no Estudo de Caso

Estratégia	Tempo de transição	Custo de energia
Ocioso	0 s	33,17 Wh
Standby	2,5 s	3,33 Wh
Hibernate	55 s	1,54 Wh

O número de máquinas que compõem a grade foi fixado em 200. Este valor foi escolhido baseado em simulações, variando este valor até que fossem gerados cenários

tanto de baixa quanto de alta contenção, ou seja, alguns cenários em que as máquinas permanecem ociosas durante muito tempo e outros em que as máquinas permanecem ociosas durante pouco tempo. A contenção é inversamente proporcional à duração da janela de observação e diretamente proporcional ao tamanho das tarefas. Para avaliarmos o consumo em relação à contenção, relacionamos os dois parâmetros de contenção em uma nova medida que foi chamada de *densidade das tarefas* (d), que definimos na Equação 2, onde r_i é o tempo de execução de uma tarefa i e j é o tamanho da janela de simulação.

$$d = \frac{\sum_{i=0}^{n} r_i}{j} \tag{2}$$

Na Tabela 2 são apresentadas as características da carga de trabalho nos cenários de demanda avaliados. Cada cenário de demanda possui 3.000 tarefas. A demanda no cenário é medida pela densidade. Por essa tabela, nota-se que do cenário de baixa densidade para o cenário de alta densidade reduz-se o intervalo entre chegadas médio de *jobs* e aumenta-se o tempo de execução médio. Adicionou-se o desvio padrão dos dados para sinalizar a variabilidade dos dados em torno dessas médias. Além disso, montamos um quarto cenário que consiste na simulação de toda a carga de trabalho (considerados os tratamentos supracitados). Nesse cenário, tem-se 20.000 tarefas. Todos os *traces* de demanda e disponibilidade utilizados nas simulações realizadas neste trabalho estão disponíveis para *download* na página do projeto `http://redmine.lsd.ufcg.edu.br/projects/list_files/green-grid`.

Tabela 2. Cenários Avaliados no Estudo de Caso

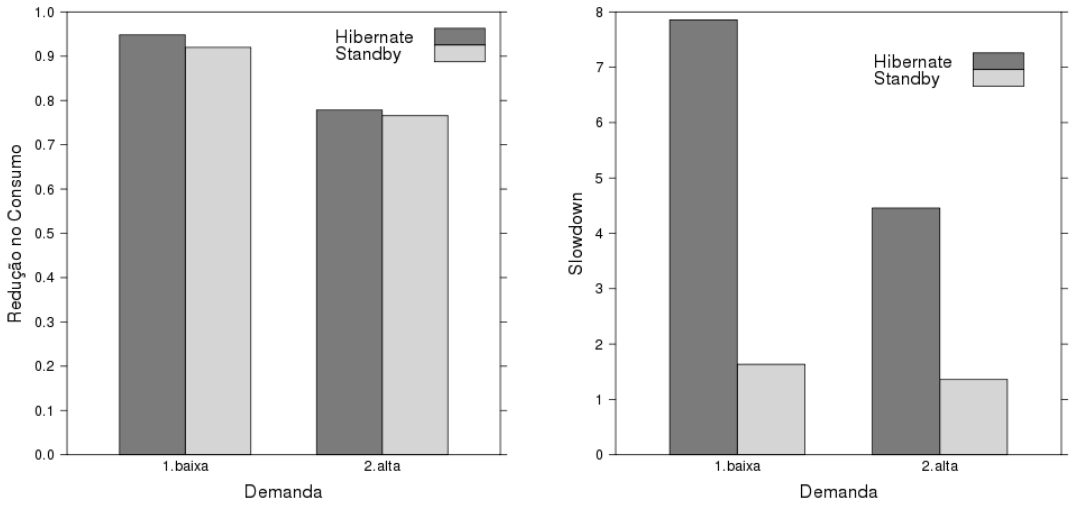
Cenário	Densidade	Intervalo entre Chegadas		Tempo de Execução	
		Média	Desvio Padrão	Média	Desvio Padrão
Baixa	0,42	49,27	323,31	20,95	40,48
Alta	86,89	15,39	222,84	572,53	1954,29

5. Apresentação e Análise dos Resultados

Os resultados são apresentados em três avaliações. A primeira avaliação consistiu na análise do consumo de energia e do *slowdown* em um cenário de total disponibilidade das máquinas, poder de processamento igual e com variação de alta e baixa demana de tarefas. A segunda avaliação considerou o ambiente típico de uma grade oportunista, com variação da disponibilidade das máquinas ao longo do tempo e máquinas com diferentes capacidades de processamento e ainda considerando cenários de alta e baixa demanda. Por fim, na terceira avaliação, utilizou-se toda a demanda de uma grade oportunista (todo o *trace*), com variação da disponibilidade das máquinas ao longo do tempo e máquinas com diferentes capacidades de processamento, no entanto, diferente da segunda avaliação, não houve separação quanto a cenários de alta e baixa demanda.

Na primeira avaliação, como as máquinas estão sempre disponíveis para grade e têm o mesmo poder de processamento, é mais fácil isolar o impacto da demanda de processamento nos resultados do *slowdown* e no consumo de energia. Os resultados dessa avaliação são apresentados na Figura 3. Por essa figura, pode-se notar que em cenário de baixa demanda tem-se alto *slowdown* da estratégia *hibernate* em relação ao estado ocioso.

Isso ocorre porque nesse cenário as tarefas têm em média 20,95 segundos, deste modo o acréscimo de 55 segundos para transição de estado aumenta muito o tempo de resposta das tarefas. De outro modo, com a estratégia *standby*, que tem custo de transição de 2,5 segundos, o acréscimo desse valor no tempo de resposta gera pouco atraso em relação ao estado ocioso.



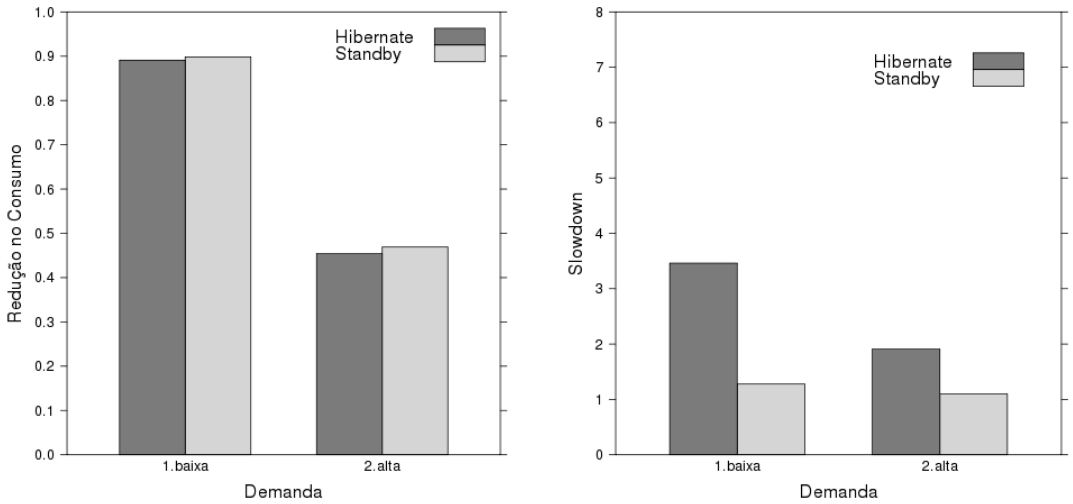
(a) Economia de Energia (b) Slowdown

Figura 3. Consumo de energia e *slowdown* com máquinas homogêneas e totalmente disponíveis

Outro fator importante nessa primeira avaliação é que a economia de energia reduz a medida em que se aumenta a densidade (Figura 3(a)). Isso porque, quando a demanda aumenta, reduz-se o tempo em que a máquina permanece no estado de computação verde o que ocasiona menor economia de energia. Como esperado, *hibernate* possibilita maior economia de energia. Essa economia, embora em níveis poucos significantes, reduz em relação ao *standby* a medida em que se aumenta a demanda. Isso ocorre devido ao aumento da demanda, que faz com que o tempo de permanência no estado de economia não supere o custo de transição para este estado.

Os resultados da segunda avaliação são apresentados na Figura 4. Essa figura mostra o *slowdown* e a redução de consumo em relação ao estado ocioso obtidos nesta avaliação. Nessa simulação, nota-se que a economia de energia reduz na medida em que aumenta a demanda da grade. Outro comportamento que merece destaque é o fato da estratégia *standby* apresentar maior redução no consumo de energia, em relação ao estado ocioso, do que a estratégia *hibernate*. Isso ocorre porque *standby* gasta menos tempo para fazer a transição de estados, gastando menos energia neste processo em relação à estratégia *hibernate*, que por demorar mais tempo consome mais energia. Outro fator é que, embora no estado *hibernate* tenha-se menor consumo de energia, o tempo de permanência em *hibernate* em média não é suficiente para cobrir esse alto consumo durante a transição, isso ocorre com mais intensidade à medida que se aumenta a demanda. Note que, na primeira avaliação, *hibernate* economizou mais energia que *standby*, o inverso do que ocorreu na segunda avaliação. Isso ocorre porque na primeira avaliação as máquinas,

que estão sempre disponíveis para a grade, têm mais tempo de ociosidade o que faz com que colocá-las em *hibernate* gere economia maior.



(a) Economia de Energia (b) Slowdown

Figura 4. Consumo de energia e *Slowdown* com máquinas heterogêneas com variabilidade na disponibilidade

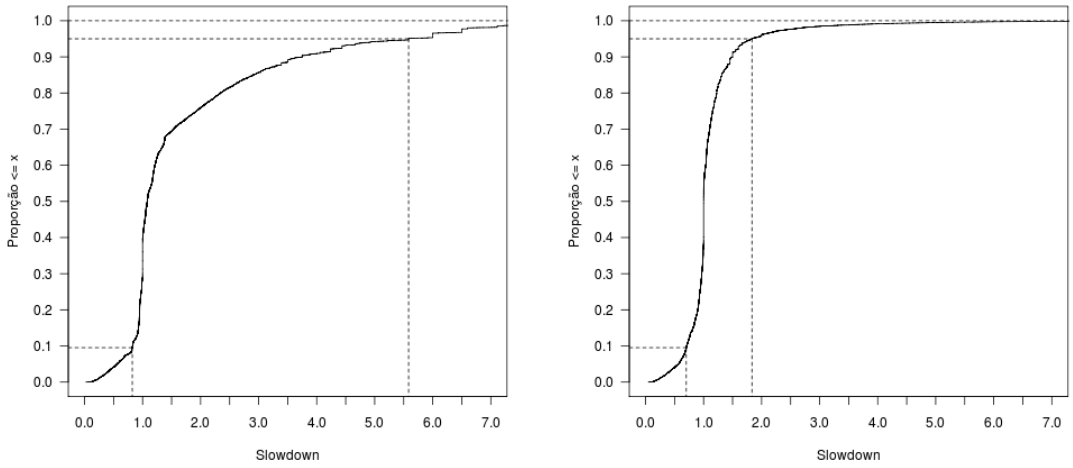
Os resultados da terceira avaliação são apresentados na Figura 5 e 6. A Figura 5 mostra o *slowdown* do tempo de resposta dos *jobs* nas estratégias *standby* e *hibernate* em relação ao estado ocioso, enquanto que a Figura 6 mostra o consumo de energia.

Podemos observar que a estratégia *standby* (Figura 5(b)) é melhor que a *hibernate* (Figura 5(a)) em relação ao impacto no tempo de resposta. De um modo geral, o *slowdown* do *hibernate* chega até 3,5 em 90% dos cenários, enquanto que no *standby* este valor não ultrapassa 1,5. Há situações em que nota-se ganho no tempo de resposta dos *jobs*. Este ganho ocorre porque, dados cenários com máquinas de diferentes poder de processamento, ao utilizar essas estratégias ocorre que alguns *jobs* têm que esperar um tempo para que as máquinas saiam do estado de computação verde e isto possibilita que estas tarefas sejam escalonadas para máquinas mais rápidas ou que fiquem mais tempo disponíveis do que as que foram escalonadas no cenário sem uso de estratégia. Esse ganho não ocorre quando se tem máquinas com mesmo poder de processamento e totalmente disponíveis (como na primeira avaliação).

A fim de identificar se a estratégia *standby* é melhor que *hibernate* para toda a carga de trabalho, verificamos o *slowdown* médio das tarefas gerado com o uso de cada estratégia. Nessa análise verificamos que *standby* tem *slowdown* médio de 1,14, com erro de $\pm$ 0,01 para um nível de confiança de 95%. De outro modo, a estratégia *hibernate* tem *slowdown* médio de 2,05, com erro de $\pm$ 0,03 para um nível de confiança de 95%. Assim, pode-se dizer que há evidências de que *standby* é melhor que *hibernate*, uma vez que *standby* gera menor *slowdown* médio e os intervalos de confiança não se cruzam.

Quanto ao consumo de energia, é possível observar que ocorreu economia de quase 50% com o uso das estratégias. Isso indica que as estratégias reduzem signifi-

cativamente o consumo de energia quando se considera toda a carga de trabalho (Figura 6).



(a) Hibernate (b) Standby

Figura 5. Função de Distribuição Acumulada (FDA) do *slowdown* para as estratégias computação verde

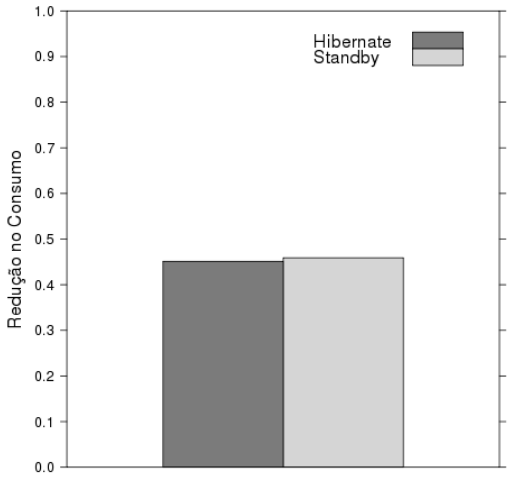


Figura 6. Consumo de Energia para toda a Workload de demanda do OurGrid

6. Conclusões e Trabalhos Futuros

Neste trabalho avaliamos o impacto que as estratégias *Standby* e *Hibernate* têm na redução do custo de energia de uma grade oportunista e no incremento do tempo de resposta dos *jobs* que executam em uma grade sujeita a essas estratégias de computação verde. Nossos resultados indicam que, em relação à economia de energia, as duas estratégias têm um comportamento semelhante, conseguindo economizar em média pelo

menos 48% da energia que é gasta em um cenário onde nenhuma estratégia de economia é usada. Por outro lado, em relação ao impacto no tempo de resposta das aplicações, a estratégia *Standby* é bem mais eficiente que a estratégia *Hibernate*, com um *slowdown* médio menor que 1,15, contra 2,05 da outra.

Como trabalhos futuros, nós iremos analisar outras estratégias de escalonamento, além de considerar não apenas aplicações intensivas em computação, mas também aplicações que fazem uso intensivo de dados. Nesse último caso, além de considerar o consumo de energia da CPU, também é necessário considerar o consumo de energia de outros componentes. As estratégias de computação verde desenvolvidas serão implantadas na grade computacional oportunista da Universidade Federal de Campina Grande (GridUFCG), que agregará mais de 2.000 *desktops* executando o *middleware* OurGrid [Cirne et al. 2006].

Referências

Barroso, L. A. and Hölzle, U. (2007). The case for energy-proportional computing. *Computer*, 40(12):33–37.

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.

Corporation, H.-P., Corporation, I., Corporation, M., Ltd., P. T., and Corporation, T. (2010). Advanced configuration and power interface specificatio. Disponível em: http://www.acpi.info/spec.htm. Acesso: Janeiro de 2010.

Economist, T. (2007). Going green. *The Economist*.

Energy Star (2009). Computer-desktops & integrated computers qp list. Disponível em: www.energystar.gov/ia/products/prod_lists/computers_prod_list.xls. Acesso: Setembro de 2009.

Indiana University (2009). Green computing project points to potential for energy savings. Disponível em: http://newsinfo.iu.edu/news/page/normal/11142.html. Acesso: Agosto de 2009.

Iosup, A., Dumitrescu, C., Epema, D., Li, H., and Wolters, L. (2006). How are real grids used? the analysis of four grid traces and its implications. In *GRID '06: Proceedings of the 7th IEEE/ACM International Conference on Grid Computing*, pages 262–269, Washington, DC, USA. IEEE Computer Society.

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(7):888–903.

Lammie, M., Thain, D., and Brenner, P. (2009). Scheduling Grid Workloads on Multicore Clusters to Minimize Energy and Maximize Performance. In *IEEE Grid Computing*.

Meisner, D., Gold, B. T., and Wenisch, T. F. (2009). Powernap: eliminating server idle power. In *ASPLOS '09: Proceeding of the 14th international conference on Architectural support for programming languages and operating systems*, pages 205–216, New York, NY, USA. ACM.

Orgerie, A. C., Lefèvre, L., and Gelas, J. P. (2008). Save watts in your grid: Green strategies for energy-aware framework in large scale distributed systems. *Parallel and Distributed Systems, International Conference on*, 0:171–178.

OurGrid Community (2010). Ourgrid status. Disponível em: `http://status.ourgrid.org/`.

Przybyla, D. and Pegah, M. (2007). Dealing with the veiled devil: eco-responsible computing strategy. In *SIGUCCS '07: Proceedings of the 35th annual ACM SIGUCCS conference on User services*, pages 296–301, New York, NY, USA. ACM.

Sharma, K. and Aggarwal, S. (2009). Energy aware scheduling on desktop grid environment with static performance prediction. In *SpringSim '09: Proceedings of the 2009 Spring Simulation Multiconference*, pages 1–8, San Diego, CA, USA. Society for Computer Simulation International.

Talebi, M. and Way, T. (2009). Methods, metrics and motivation for a green computer science program. *SIGCSE Bull.*, 41(1):362–366.

Williams, J. and Curtis, L. (2008). Green: The new computing coat of arms? *IT Professional*, 10(1):12–16.

Zong, Z., Qin, X., Ruan, X., Bellam, K., Yang, Y., and Manzanares, A. (2007). A simulation framework for energy efficient data grids. In *WSC '07: Proceedings of the 39th conference on Winter simulation*, pages 1417–1423, Piscataway, NJ, USA. IEEE Press.