Economia de Energia usando Computação Voluntária Virtualizada

Fábio Diniz Rossi[1], **Patric Lincoln Ramires Izolan**[1], **Garibaldi da Silveira Júnior**[1], **Jader Renan da Silva Almeida**[1]

[1]Campus de Alegrete
Instituto Federal de Educação, Ciência e Tecnologia Farroupilha
RS 377 – KM 27 – Alegrete – RS

fabio.rossi@iffarroupilha.edu.br

Abstract. *Traditionally, workloads submitted to grid computing environments are bag-of-tasks that can be processed in a distributed way, and using idle hosts connected to a grid environment. Thus, in idle moments, these hosts are connected and the tasks are executed. This is a model designed to not interfere with the rate of use of such hosts. But even running, these hosts often do not utilize all its processing power. This paper proposes an approach that uses underutilized hosts through virtualization, adjusting the use of grid applications to the leftover resources from daily usage of hosts. This approach can reduce the runtime of the applications while saving energy.*

Resumo. *Tradicionalmente, as cargas de trabalho submetidas aos ambientes de computação em grade são do tipo saco-de-tarefas, que podem ser processadas de forma distribuída devido a falta de dependência entre os dados. Esse modelo utiliza hosts ociosos conectados a um ambiente de grade, portanto em momentos em que esses* hosts *não estão em uso, as tarefas podem ser executadas. Porém, com o poder computacional atual, mesmo em uso, os computadores atuais não utilizam todo o seu poder de processamento. Esse artigo propõe a adoção de virtualização visando utilizar fatias ociosas dentro de um novo em execução, disponibilizando essa fatia ao processamento de aplicações de grade. Essa arquitetura pode reduzir o tempo de execução das aplicações de grade, ao mesmo tempo que permite economia de energia.*

1. Introdução

Geralmente, a computação voluntária [Nov et al. 2010] consiste em um ambiente computacional distribuído usualmente associado a projetos científicos em larga escala, que podem se utilizar do poder de processamento de computadores ociosos ao redor do mundo, visando processar grandes quantidades de informação. Para que isso seja possível, os dados são divididos e enviados para diferentes *hosts* de uma grade, podendo assim ser processados de forma simultânea, reduzindo custos e tempo de pesquisa. Esses *hosts* suportam um *software* cliente que conecta esse *host* a uma grade computacional distribuída e heterogênea, criando um ambiente altamente escalável.

Com o aumento no poder de processamento dos computadores pessoais [Herlihy 2007], existe subutilização dos recursos. Além disso, as aplicações atuais não são paralelas o suficiente para tirar proveito desses recursos em sua totalidade, como de

processadores multi-núcleo. Devido a isso, embora existam aplicações que fazem uso intensivo dos recursos [Hwang et al. 2003], essa não é a realidade da maioria das aplicações atuais. Portanto, embora um computador execute uma aplicação, provavelmente exista uma fatia de recursos ociosos e que estão disponíveis para uso. Esse cenário é interessante para utilização da computação voluntária, que executa de forma concomitante à utilização habitual, sem interferir sobre esta.

A tecnologia que permite usar fatias de recursos disponíveis, aumentando ou reduzindo a alocação dos recursos dependendo da necessidade, é a virtualização [Uhlig et al. 2005]. Essa tecnologia permite executar sistemas operacionais independentes, concorrentemente, sobre os mesmos recursos. A vantagem dessa arquitetura comparada com arquiteturas tradicionais de computação voluntária consiste na não interrupção dos serviços da grade. Atualmente, existem diversos trabalhos que estudam os *trade-offs* entre desempenho e consumo de energia, como em [Siam et al. 2010], [Abts et al. 2010] e [Marzolla 2012]. Esta é uma preocupação crescente, uma vez que foram alcançados os limites físicos de silício [Bohr 1998]. Portanto, nosso trabalho combina algumas tecnologias que podem reduzir esse *trade-off*. A grande questão respondida por este artigo, além de medir o desempenho de um ambiente de computação voluntária virtualizado, é verificar o quanto de economia de energia esta solução promove. Além disso, este artigo mostra que as relações entre desempenho (tempo de execução das aplicações de rede) e economia de energia, expondo este *trade-off*.

Este artigo apresenta dois problemas clássicos nas soluções de computação voluntária. O primeiro refere-se ao aumento no consumo de energia. O processador ocioso geralmente tem menor consumo de energia quando comparado ao estado de ativo. Além disso, os participantes podem deixar o nó de rede conectado por 24 horas, e desativar a economia de recursos tais como os estados de suspensão. Além disso, se a refrigeração adequada não está em vigor, esta carga constante sobre o nó voluntário pode causar superaquecimento. O segundo problema é a redução no desempenho do nó. Se a aplicação tenta executar computação voluntária enquanto o computador estiver em uso, as aplicações do usuário serão afetadas. Isto ocorre devido ao aumento da contenção do processador, memória cache, disco, E/S, e rede.

Desta forma, os principais objetivos deste trabalho são:

- verificar se a camada de virtualização isola o cliente voluntário virtualizado dos aplicativos de usuário;
- testar diferentes configurações de taxas de utilização entre as aplicações do usuário e aplicações voluntárias virtualizadas;
- propor uma nova abordagem, permitindo flexibilidade de aplicações voluntárias;
- visualizar os limites do trade-off entre desempenho e consumo de energia.

O trabalho está dividido da seguinte forma: na Seção II são descritos os conceitos de computação voluntária e virtualização; a Seção III apresenta os trabalhos relacionados; a Seção IV descreve o ambiente e seus recursos; a Seção V mostra a análise e; finalmente, a Seção VI apresenta nossas conclusões e trabalhos futuros.

2. Referencial Teórico

Esta seção apresenta conceitos gerais sobre computação voluntária e virtualização - conjunto de tecnologias estudadas neste trabalho.

2.1. Computação Voluntária

A computação voluntária [Nov et al. 2010] tem como objetivo oferecer à projetos de larga-escala, a capacidade ociosa de computadores pessoais distribuídos geograficamente, visando processar grandes quantidades de informação, reduzindo assim o tempo de processamento de um determinado experimento. Qualquer pessoa pode ser um voluntário. É necessário instalar o *software* cliente no computador e ser inscrito em algum projeto. Este software é responsável pela comunicação entre o computador pessoal e o servidor de borda da grade, recebendo tarefas eventualmente, quando estiver ocioso.

Projetos que levariam milhares de anos para processar a informação, tem seu tempo de execução reduzido consideravelmente através da utilização da computação voluntária, devido a divisão problema em milhões de pequenas unidades que podem ser processados de forma paralela e distribuída. O servidor central envia pequenos pacotes de dados para todos os computadores cadastrados na grade. Embora alguns desses projetos visem causas coletivas, que exigem computadores robustos de grandes centros de pesquisa, a maioria deles estão satisfeitos com os uso de computadores pessoais. Assim, estes dados são processados por computadores pessoais, e os resultados são enviados através da Internet para um servidor central. Tais dados são recebidos pelo servidor, analisados, e o processo de envio de novas tarefas é iniciada novamente, até que toda a carga de trabalho tenha sido finalizada.

2.2. BOINC

Neste artigo, utilizamos o *software* cliente de grade chamado BOINC [Anderson 2004]. A motivação para o desenvolvimento do BOINC partiu da proposta do projeto SETI (Search for Extraterrestrial Intelligence), abandonado pela NASA em meados da década de 70. Em 1990, a proposta do SETI foi retomada pelo instituto SETI League, que em conjunto com a Universidade de Berkeley lançou o programa científico chamado SERENDIP, que impulsionou a computação distribuída através de computadores pessoais com o lançamento do SETI@Home. Em 1992, pesquisadores da UC Berkeley desenvolveram o BOINC, diferentemente do SETI, com o objetivo de aceitar qualquer tipo de tarefa para processamento. O sucesso deste projeto deixou claro que a computação distribuída podia ser usado para muitos outros projetos científicos que exigem capacidade de processamento. O BOINC é altamente personalizável e permite escolher quais projetos serão ajudados e a quantidade de recursos que estarão disponíveis. BOINC faz atualizações automáticas das aplicações, além de *download* automático de novas tarefas a ser processadas.

2.3. Virtualização

Nos últimos anos, a capacidade de processamento nos computadores aumentou consideravelmente. Existem situações em que as aplicações podem ser executadas de forma mais eficaz, melhorando o desempenho do processador. Uma proposta que permite esse ganho consiste na utilização de virtualização [Uhlig et al. 2005]. A virtualização permite dissociar o *hardware* do sistema operacional, trazendo ferramentas novas e úteis, além de permitir que o administrador do sistema controle o uso do processador, memória, armazenamento e outros recursos do sistema operacional sobre os sistemas hóspedes, chamados de máquinas virtuais (VMs). Assim, cada VM recebe a quantidade necessária de recursos. Este controle elimina o risco de alguma VM utilize toda a memória disponível ou

toda a carga do processador. Este tipo de flexibilidade muda o conceito tradicional de uso dos recursos e planejamento de capacidade. Em ambientes virtualizados, é possível lidar com os recursos computacionais, como processador, memória cache e armazenamento, que podem ser facilmente realocados para atender a demanda de aplicações, quando necessário. A virtualização não é um conceito novo, pois na década de 60 já foi aplicada à *mainframes*. Atualmente, com o avanço poder de processamento dos computadores pessoais, diferentes modelos de gerenciadores de VMs (VMMs) têm sido desenvolvidos e seu uso tornou-se generalizado.

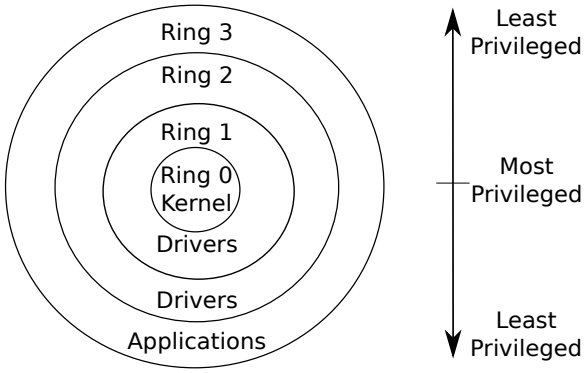

Figura 1. Estrutura de Anéis

Dessa maneira, a virtualização permite executar vários sistemas operacionais na mesma máquina física. Isto é possível com a utilização de programas específicos que cria máquinas virtuais, emulando os componentes físicos de um nó. A capacidade de isolamento da virtualização é um dos principais fatores para apoiar este trabalho. Esse isolamento é feito através de uma técnica chamada de anel de compressão (Figura 1. Processadores x86 padrão apresentam quatro níveis de prioridade para a execução de código, numeradas entre 0-3. Código no nível 0 podem executar qualquer instrução sobre a CPU, enquanto em nível 3 (a menos restrito) mantém as aplicações em geral e que não podem executar diretamente sobre a CPU. Estes níveis de prioridade ganharam o nome de anéis por causa da maneira como eles foram ilustrados no manual de programação do *chip* 80386. Isto também se relaciona com o isolamento do desempenho, o que significa que uma VM pode ser isolada da fatia de recursos alocados para o sistema operacional (SO) base ou outras VMs hospedadas na mesma máquina física. O VMM que executa dentro do SO base, quando uma VM é criada, move o kernel virtual para executar no nível 1, ao invés de nível 0. O kernel virtual pensa que está em execução no nível 0, mas está atualmente em execução no nível 1, e isso permite que o VMM possa monitorar a execução da VM e gerenciar o acesso à memória e periféricos, e, eventualmente, emular instruções de software que só poderia realizar chamadas de sistema no verdadeiro nível 0

2.4. VirtualBox

Neste trabalho utilizamos um monitor de máquinas virtuais (VMM) chamado VirtualBox. VirtualBox é um *software* que permite suporte à VMs, desenvolvido pela Sun Microsystems, e que foi posteriormente adquirida pela Oracle. Este VMM permite criar ambientes para instalação de diferentes SOs sobre um SO hospedeiro, compartilhando o mesmo hardware físico.

VirtualBox tem um desenho extremamente modular com interfaces de programação internas e um design cliente-servidor bem definido. O SO hospedeiro mantém a comunicação entre o VMM e as VMs. VirtualBox também oferece ferramentas de desenvolvimento de software de fonte aberta. As definições de configuração de máquinas virtuais são armazenados inteiramente em XML e são independentes das máquinas locais. Portanto, as definições podem ser facilmente transferidos para outros computadores.

3. Trabalhos Relacionados

Existem diversos estudos utilizando computação voluntária em conjunto com a tecnologia de virtualização, como o [Ben Belgacem et al. 2012], [Marosi et al. 2013], [Zhang et al. 2011] e [Fernández De Vega et al. 2013]. O artigo [Ferreira et al. 2011] mostra a tecnologia de virtualização como *sandbox* para aplicações de segurança no BOINC. Além disso, o artigo apresenta o desenvolvimento de um *middleware* chamado libboincexec, permitindo ao BOINC executar de forma otimizada em diversos VMMs. Em outro estudo [Lombrana González et al. 2010], são apresentadas algumas limitações ao usar o BOINC com algumas aplicações específicas, devido ao mal gerenciamento de dependências. Os autores implementaram uma solução que detecta se o ambiente virtual fornece todas as dependências, ou realiza o *download* destas através da Internet, com a finalidade de que a aplicação seja executada com êxito.

Algumas soluções para ambientes virtualizados que suportam BOINC estão em desenvolvimento, como em [McGilvary et al. 2013]. O trabalho usa a virtualização visando portar o BOINC em qualquer ambiente de computação, e capacitá-lo a migrar aplicações de uma VM para outra, aproveitando-se de recursos como elasticidade, capacidade intrínseca em ambientes virtualizados. O trabalho [Cavalcanti et al. 2006] usa VMs para atender a necessidade de segurança no compartilhamento de arquivos. Para isso, o estudo utilizou Xen para oferecer a segurança de um sistema de arquivos distribuído para o ambiente OurGrid. Embora os resultados mostraram que não houve nenhum ganho em métricas de desempenho, a utilização de sistema de arquivos sobre virtualização mostrou bons resultados quanto à transferência de arquivos entre máquinas locais. O artigo [Cunsolo et al. 2009] apresenta o conceito de rede como um serviço. Este novo paradigma é comparado com o estado da arte, e discutida como uma proposta viável a ser implementada.

No melhor de nosso conhecimento, nenhum trabalho anterior usou a tecnologia de virtualização objetivando utilizar as fatias de recursos ociosos em nós em execução, permitindo uma computação voluntária em um ambiente não-ocioso. Além disso, nenhum outro estudo avaliou o consumo de energia relacionado com a utilização de virtualização como suporte à computação voluntária em computadores *online*.

4. Ambiente de Teste

Uma arquitetura de computação voluntária tradicional é apresentada na Figura **??**. Nesta, existem quatro computadores pessoais conectados a um servidor, formando uma grade computacional. A máquina 1 apresenta a execução de um processo em espaço de usuário (o raio representa um fluxo de execução), o que significa que esta máquina não está ociosa neste momento para aplicações de grade. As máquinas 2, 3, 4 não apresentam qualquer fluxo no espaço do usuário, permitindo que a grade envie e mantenha fluxos de execução sobre seus recursos (o fluxo de execução no BOINC).

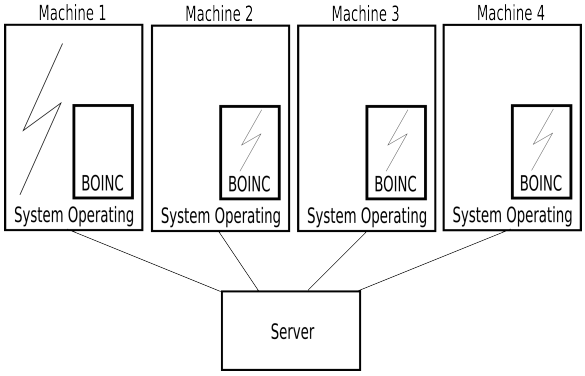

Figura 2. Ambiente Tradicional de Grade

A principal questão discutida neste artigo é que, mesmo se as máquinas estão sendo usados pelo usuário, seus recursos estão sendo subutilizadas devido ao grande poder de processamento e latência das memórias das máquinas atuais. Por isso, propõe-se uma nova arquitetura que pode ser vista na Figura **??**, que usa a tecnologia de virtualização para manter sempre a execução da grade ativa, mesmo se o usuário está usando a máquina. A virtualização tem a capacidade de adaptar-se à parcela de recursos que outros aplicativos não estão usando, o que significa que não existirá interferência sobre o desempenho das aplicações do usuário.

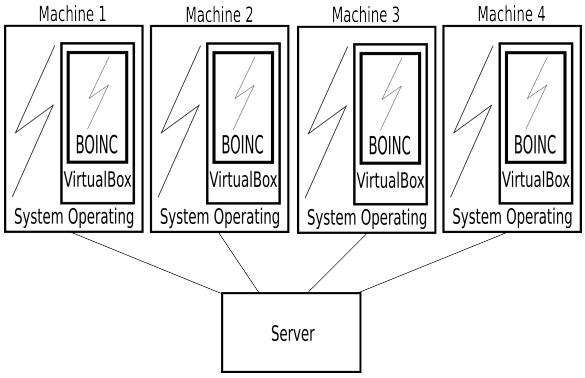

Figura 3. Virtualized Grid Environment

As avaliações foram realizadas em uma arquitetura cliente-servidor, com vários clientes acessando a aplicação de grade virtualizada em uma máquina servidor, através de uma rede Gigabit Ethernet. Os servidores usados nos experimentos são compostas de quatro processadores com dois núcleos Intel Xeon E5520 cada (16 núcleos lógicos no total), 2.27GHz, 16 Gb de RAM. A medição do consumo de energia foi realizada com a utilização de um multímetro que está ligado entre a fonte de alimentação e cada máquina da grade. Este dispositivo (multímetro digital EZ-735) tem uma conexão USB que permite a leitura externa.

Os testes foram realizados utilizando um conjunto de testes padrão do BOINC, levando em média, 24 horas para realizar todo o processamento. Para um teste mais realístico, as máquinas revezaram períodos de atividade e inatividade por parte do usuário, sendo 8 horas de atividade e 16 horas de inatividade. O teste escolhido foi calcular a conjectura de Collatz (também conhecido como $3n+1$). A conjectura tem como regra que

qualquer número natural, quando aplicado a esta conjectura, no final será sempre reduzido a 1. Em nossos experimentos, a totalidade do teste levou 40 horas de processamento da grade.

Este cenário é muito realista, se for comparado com um ambiente de produção, onde a máquina é utilizada durante o período de tempo de um trabalhador comum, e a aplicação da grade usa o tempo ocioso fora destas 8 horas. Com este cenário de teste usando o ambiente virtualizado, pudemos desenvolver vários exemplos com diferentes configurações,como limitação da taxa de utilização da VM entre 10% e 90% (em relação às fatias dos recursos discutidas anteriormente).

5. Avaliações

Esta seção apresenta as avaliações das métricas propostas relativas ao tempo de execução e ao consumo de energia. A primeira avaliação mostra a sobrecarga do ambiente virtualizado sobre a aplicação de grade.

A Figura 4 mostra a diferença no rendimento entre o ambiente nativo do Linux e do ambiente virtualizado, quando usamos o BOINC sobre VMs. Como a aplicação recebe uma carga do tipo saco-de-tarefas, e após processar, retorna a resposta para o servidor, é possível medir a taxa de transferência de ambos os modos: quando os pacotes são enviados; e quando os pacotes são recebidos pelo servidor. Como toda a comunicação é interpretada pelo VMM, existe um atraso de até 25% nas transferências. Essa sobrecarga não é significativo quando comparada com os outros benefícios que a virtualização traz esta nova abordagem.

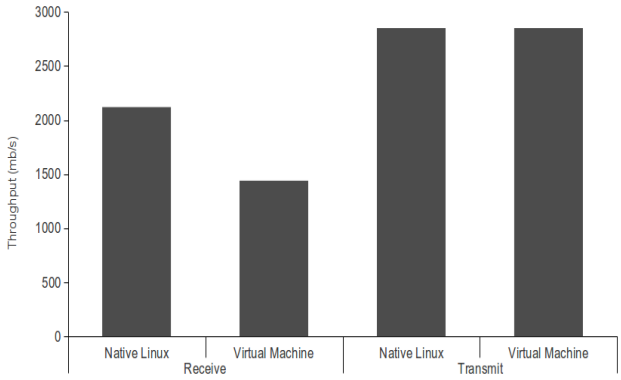

Figura 4. Throughput

Outro fator importante a ser avaliado consiste no tempo de execução das aplicações. A Figura 5 mostra os resultados da utilização do ambiente virtual juntamente com o BOINC em termos de tempo de execução. Em todos os casos, existe uma melhoria no tempo de execução. O tempo de execução no pior caso, é bastante próximo ao teste de base que levou 40 horas. O tempo de execução mostrou maior impacto nos testes em que existe maior utilização da VM e, consequentemente, maior largura de banda disponível para a VM mais próxima. A capacidade que torna isso possível é o isolamento entre VMs (apresentada anteriormente devido aos níveis de acesso em forma de anel do SO), que não excedem os limites estabelecidos como taxas de uso nesse artigo. Devido a isso, não existiu interferência sobre o desempenho das outras aplicações do usuário.

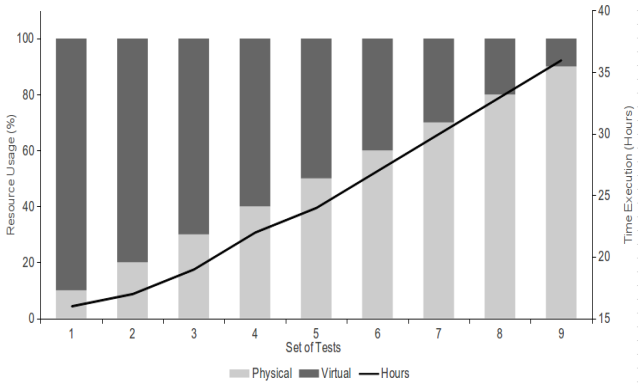

Figura 5. Tempo de Execução

Quanto ao consumo de energia avaliado nessa proposta, é possível visualizar que podemos alcançar um certo limite de economia de energia. Para comparar o consumo de energia da presente proposta, utilizamos como valor de base, um teste com a mesma carga, mas em um ambiente nativo sem virtualização, como pode ser visto na Tabela **??**.

Tabela 1. Consumo de Energia em Ambiente de Teste Real

Real-Virtual	Tempo de Execução	Consumo de Energia
100-0	40	7040

Na Tabela 2, os valores estão relacionados às taxas de utilização dos recursos entre as aplicações do usuário e da VM+BOINC. Além disso, é mostrado o tempo de execução em horas para cada taxa de utilização, e o consumo de energia médio para cada máquina em watts.

Tabela 2. Consumo de Energia em Ambiente de Teste Real-Virtual

Real-Virtual	Tempo de Execução	Consumo de Energia
10-90	16	4640
20-80	17	4930
30-70	19	5510
40-60	22	6380
50-50	24	6960
60-40	27	7830
70-30	30	8700
80-20	33	9570
90-10	36	10440

Até um limite de cerca de 50% de uso entre as aplicações do usuário e ambiente virtualizado com BOINC, existe economia de energia. Isto é associado com menor tempo de execução da aplicação de grade, que apesar de ter um maior consumo de energia du-

rante a execução, o tempo de execução foi tão menor que no final, consumiu menos energia. Acima deste valor, embora o tempo de execução ainda mostre bons resultados quando comparado com os testes do ambiente nativo, o consumo de energia aumenta significativamente. Talvez isto não seja uma limitação decisiva para a adoção da proposta, dado que nos computadores que estão atualmente disponíveis para grades computacionais, tais como computadores pessoais, o uso de recursos da maioria dos usuários geralmente não ultrapassa 50%.

6. Conclusões e Trabalhos Futuros

A virtualização é a técnica de execução de um ou mais servidores virtuais em um servidor físico. Isto permite uma maior densidade de utilização dos recursos (hardware, armazenamento, etc), permitindo a manutenção de isolamento e segurança. Com base na característica de isolamento, este trabalho propõe o uso da tecnologia de virtualização como suporte à computação voluntária, permitindo a essa arquitetura de processamento distribuído executar de forma concomitante com outras aplicações do usuário, não apenas em momentos de ociosidade.

Os resultados mostraram que existem vantagens nesta abordagem, em que não existe nenhuma influência sobre as aplicações do usuário, ao passo que existe melhoria no desempenho das aplicações de grade, devido a uma melhor utilização dos recursos disponíveis pela virtualização. Além disso, os resultados mostraram que nossa proposta pode economizar energia até uma taxa de utilização de 50% dos recursos, o que valida nossa proposta se levarmos em consideração a baixa taxa e utilização dos recursos pelos usuários comuns. Existem VMs com maior desempenho, mas com menos isolamento. Assim como existem VMs com menos desempenho, mas com mais isolamento. Como trabalho futuro, propõe-se uma avaliação com tais tipos diferentes de VM e *containers*.

Referências

Abts, D., Marty, M. R., Wells, P. M., Klausler, P., and Liu, H. (2010). Energy proportional datacenter networks. *SIGARCH Comput. Archit. News*, 38(3):338–347.

Anderson, D. P. (2004). Boinc: A system for public-resource computing and storage. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, GRID '04, pages 4–10, Washington, DC, USA. IEEE Computer Society.

Ben Belgacem, M., Abdennadher, N., and Niinimaki, M. (2012). Virtual ez grid: A volunteer computing infrastructure for scientific medical applications. *Int. J. Handheld Comput. Res.*, 3(1):74–85.

Bohr, M. (1998). Silicon trends and limits for advanced microprocessors. *Commun. ACM*, 41(3):80–87.

Cavalcanti, E., Assis, L., Gaudencio, M., Cirne, W., and Brasileiro, F. (2006). Sandboxing for a free-to-join grid with support for secure site-wide storage area. In *Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing*, VTDC '06, pages 11–, Washington, DC, USA. IEEE Computer Society.

Cunsolo, V. D., Distefano, S., Puliafito, A., and Scarpa, M. (2009). Cloud@home: bridging the gap between volunteer and cloud computing. In *Proceedings of the 5th international conference on Emerging intelligent computing technology and applications*, ICIC'09, pages 423–432, Berlin, Heidelberg. Springer-Verlag.

Fernández De Vega, F., Olague, G., Trujillo, L., and Lombraña González, D. (2013). Customizable execution environments for evolutionary computation using boinc + virtualization. 12(2):163–177.

Ferreira, D., Araujo, F., and Domingues, P. (2011). libboincexec: A generic virtualization approach for the boinc middleware. In *Proceedings of the 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum*, IPDPSW '11, pages 1903–1908, Washington, DC, USA. IEEE Computer Society.

Herlihy, M. (2007). The multicore revolution: the challenges for theory. In *Proceedings of the 27th international conference on Foundations of software technology and theoretical computer science*, FSTTCS'07, pages 1–8, Berlin, Heidelberg. Springer-Verlag.

Hwang, S., Jeong, K., Im, E., Woo, C., Hahn, K.-S., Kim, M., and Lee, S. (2003). An analysis of idle cpu cycles at university computer labs. In *Proceedings of the 2003 international conference on Computational science and its applications: PartI*, ICCSA'03, pages 733–741, Berlin, Heidelberg. Springer-Verlag.

Lombrana González, D., Jiménez Laredo, J., Fernández de Vega, F., and Merelo Guervós, J. (2010). Characterizing fault-tolerance of genetic algorithms in desktop grid systems. In Cowling, P. and Merz, P., editors, *Evolutionary Computation in Combinatorial Optimization*, volume 6022 of *Lecture Notes in Computer Science*, pages 131–142. Springer Berlin Heidelberg.

Marosi, A., KováCs, J., and Kacsuk, P. (2013). Towards a volunteer cloud system. *Future Gener. Comput. Syst.*, 29(6):1442–1451.

Marzolla, M. (2012). Optimizing the energy consumption of large-scale applications. In *Proceedings of the 8th International ACM SIGSOFT Conference on Quality of Software Architectures*, QoSA '12, pages 123–132, New York, NY, USA. ACM.

McGilvary, G. A., Barker, A., Lloyd, A. D., and Atkinson, M. P. (2013). V-boinc: The virtualization of boinc. *CoRR*, abs/1306.0846.

Nov, O., Anderson, D., and Arazy, O. (2010). Volunteer computing: a model of the factors determining contribution to community-based scientific research. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 741–750, New York, NY, USA. ACM.

Siam, M. Z., Krunz, M., Cui, S., and Muqattash, A. (2010). Energy-efficient protocols for wireless networks with adaptive mimo capabilities. *Wirel. Netw.*, 16(1):199–212.

Uhlig, R., Neiger, G., Rodgers, D., Santoni, A. L., Martins, F. C. M., Anderson, A. V., Bennett, S. M., Kagi, A., Leung, F. H., and Smith, L. (2005). Intel virtualization technology. *Computer*, 38(5):48–56.

Zhang, Y., Li, Y., and Zheng, W. (2011). Using user-level virtualization in desktop grid clients for application delivery and sandboxing. In *Proceedings of the 2011 Fourth International Symposium on Parallel Architectures, Algorithms and Programming*, PAAP '11, pages 289–293, Washington, DC, USA. IEEE Computer Society.