Proposta de *Workflow* para Alocação de Máquinas Virtuais Utilizando Características de Processamento

Paulo Antônio Leal Rego[1,2], **Emanuel Ferreira Coutinho**[1,2,3], **José Neuman de Souza**[1,2]

[1]Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat)
[2]Mestrado e Doutorado em Ciência da Computação (MDCC)
[3]Instituto UFC Virtual
Universidade Federal do Ceará
Fortaleza – CE – Brasil

pauloalr@lia.ufc.br, emanuel@virtual.ufc.br, neuman@ufc.br

Abstract. *This paper describes a proposal for a workflow to handle the allocation of computing resources, specifically the processing power in heterogeneous environments, where there is a wide variety of processors on the machines of the infrastructure. The contribution aims to standardize the representation of the processing capacity in terms of what we call Processing Unit (PU), combined with the limiting of the CPU usage to provide performance isolation and maintaining the processing capability independent of the Physical Machine (PM) in what the Virtual Machine (VM) has been allocated. This work presents experiments with a focus on defining and managing the PU, which aim to assess the initial activities of the proposed workflow.*

Resumo. *Esse artigo descreve uma proposta de workflow para tratar alocação de recursos computacionais, especificamente poder de processamento, em ambientes heterogêneos, onde existe uma grande variedade de processadores nas máquinas da infraestrutura. A contribuição visa padronizar a representação da capacidade de processamento, em termos do que chamamos de Unidade de Processamento (UP), aliado à limitação do uso da CPU para prover isolamento de desempenho e manter a capacidade de processamento independente da Máquina Física (MF) em que a Máquina Virtual (MV) foi alocada. O trabalho apresenta experimentos com foco na definição e gerenciamento da UP, que visam avaliar as atividades iniciais do workflow proposto.*

1. Introdução

A exigência crescente de recursos computacionais em áreas como computação científica abriu o caminho para o desenvolvimento da computação paralela e distribuída. Depois do sucesso generalizado dos clusters e a adoção mais lenta das Grades Computacionais, o modelo de Computação em Nuvem é cada vez mais adotado em soluções distribuídas [Mc Evoy et al. 2011].

Com o rápido desenvolvimento das tecnologias de processamento e armazenamento, e o grande sucesso da Internet, os recursos computacionais tornaram-se mais baratos, mais poderosos e mais ubiquamente disponíveis que antes. Esta nova tendência tecnológica tornou possível a existência da Computação em Nuvem, onde os recursos são providos sob demanda através da Internet [Zhang et al. 2010].

A virtualização é uma tecnologia chave da plataforma de Computação em Nuvem, onde as aplicações são encapsuladas dentro de máquinas virtuais (MV) e estas são mapeadas dinamicamente para um conjunto de servidores físicos [Sonnek and Chandra 2009]. Os serviços em execução nas MVs são executados separadamente de todos os processos das máquinas físicas, incluindo o sistema operacional destas. Esta separação traz benefícios como segurança e portabilidade [Chen and Noble 2001].

A Computação em Nuvem aproveita da tecnologia de virtualização para alcançar o objetivo de prover recursos computacionais como deseja a Computação Utilitária. O provedor de infraestrutura pode se aproveitar das tecnologias de migração de MVs para atingir um alto grau de consolidação dos servidores, e assim, maximizar a utilização de recursos enquanto diminui os custos com energia e refrigeração [Zhang et al. 2010].

No entanto, as técnicas de consolidação e alocação dinâmica de MVs têm tratado o impacto da sua utilização como uma medida independente de localização. É geralmente aceito que o desempenho de uma MV será o mesmo, independentemente de qual máquina física (MF) ela é alocada. Esta é uma suposição razoável para um ambiente homogêneo, onde as MFs são idênticas e as MVs estão executando o mesmo sistema operacional e aplicativos. No entanto, em um ambiente de Computação em Nuvem, esperamos compartilhar um conjunto composto por recursos heterogêneos, onde os servidores físicos podem variar em termos de capacidades de seus recursos e afinidades de dados [Sonnek and Chandra 2009].

Este artigo trata o problema de prover recursos computacionais, especificamente poder de processamento, em ambientes heterogêneos, onde existe uma grande variedade entre os processadores das máquinas da infraestrutura. Será proposto um *workflow* que visa padronizar a representação da capacidade de processamento, denominada por Unidade de Processamento (UP), aliado à limitação do uso da CPU para prover isolamento de desempenho (*performance isolation*[1]) e manter a capacidade de processamento independente da MF em que a MV tenha sido alocada.

O trabalho está organizado da seguinte forma. A Seção 2 apresenta o problema tratado; a Seção 3 descreve a proposta, detalhando o modelo e apresentando as soluções para limitar o uso da CPU; a Seção 4 apresenta a metodologia de avaliação utilizada; na Seção 5 são apresentados os experimentos e a discussão dos resultados; já na Seção 6 são apresentados os trabalhos relacionados, enquanto que a Seção 7 apresenta a conclusão e trabalhos futuros.

2. Definição do Problema

A maioria dos *middlewares* de Computação em Nuvem utiliza a quantidade de CPU e memória como unidades de escalonamento no momento de decidir em qual servidor físico as máquinas virtuais devem ser alocadas.

A maneira como esses dados são representados não identificam o real poder de processamento das MFs. Podemos observar na Figura 1, a maneira como os dados são representados no *middleware OpenNebula* [OpenNebula 2011].

A MF representada como no3 consta com 400% de processamento livre, enquanto

[1]É uma medida de quão bem as MVs visitantes estão protegidas do intenso consumo de recursos das outras MVs [Deshane et al. 2008].

```
pesquisa@controlador:~$ onehost list
  ID NAME            CLUSTER   RVM   TCPU   FCPU   ACPU    TMEM    FMEM STAT
   0 no1             default     0    400    400    400    3.9G    3.7G   on
   1 no2             default     0    400    400    400    3.9G    3.7G   on
   2 no3             default     0    400    400    400    3.9G    3.7G   on
   3 no4             default     0    800    800    800   23.6G   23.4G   on
```

Figura 1. Representação dos dados no *middleware* OpenNebula

a MF representada como no4 tem 800%, mas não é possível fazer nenhuma relação entre esses valores, ou seja, não necessariamente o no4 tem o dobro do poder de processamento do no3.

Como o ambiente de Computação em Nuvem é bastante heterogêneo, podemos ter máquinas com diferentes quantidades de CPU e estas podem possuir poder de processamento diferente. Essa diferença entre as CPUs pode influenciar o desempenho das MV, e consequentemente as aplicações que estiverem encapsuladas nelas.

Como experimento motivacional, foi realizado um *benchmark*[2] em duas máquinas da infraestrutura utilizada. O *benchmark Linpack* foi executado numa máquina virtual com uma VCPU (CPU Virtual), 1GB de memória e sistema operacional Ubuntu Server 10.10, 64bits e kernel 2.6.35-25. Esta MV foi executada em dois servidores diferentes, Servidor A e B, cujas configurações podem ser vistas na Tabela 1.

Servidor A	**Servidor B**
Intel Corei5-750 (8M Cache, 2.66 GHz) sem Hyper-Threading	Intel Corei7-930 (8M Cache, 2.80 GHz) com Hyper-Threading
4GB memória	24GB memória
Ubuntu Server 10.04, 64bits	Ubuntu Server 10.04, 64bits
Hypervisor KVM	*Hypervisor* KVM

Tabela 1. Configuração das máquinas físicas

O resultado[3] deste experimento foi um poder de processamento de 6,8 GFLOPS[4] para o Servidor A e 3,77 GFLOPS para o Servidor B, o que mostra que a MV executada no Servidor A obteve resultado muito superior à executada no Servidor B, causados pela diferença de servidor físico.

3. Proposta

O trabalho tem como objetivo propor um *workflow* que descreva uma maneira de trabalhar a capacidade de processamento de MVs. O objetivo desta proposta é melhorar a representação dos dados, para que estes possam demonstrar a diferença de processamento entre MFs, e apresentar uma maneira de manter a capacidade de processamento solicitada sempre no mesmo nível, independente da MF em que a MV está alocada.

A ideia é definir uma Unidade de Processamento (UP) que tenha um valor constante de GFLOPS, e representar o poder de processamento de cada MF em função dessa constante UP.

[2]Os testes e *benchmark* serão detalhados nas seções seguintes.
[3]Os gráficos desse experimento serão apresentados na Seção 5.
[4]flops = operações de ponto flutuante por segundo. gigaflop = 10^9 flops.

Uma vez que todas as MFs possuírem seu poder de processamento representado em função da UP, podemos alocar as MVs tendo como base a quantidade de UPs solicitada. Como o poder de processamento de uma UP no Servidor A deve ser igual ao do Servidor B, a MV terá o poder de processamento mantido, independente de onde ela for alocada, e caso haja alguma migração.

Para alcançar os resultados desejados, é preciso que os *hypervisors* permitam uma maneira de definir a percentagem da CPU que será utilizada por uma dada MV. Entre os *hypervisors* mais utilizados de código aberto, temos o Xen e o KVM (*Kernel-based Virtual Machine*).

O Xen possui esta funcionalidade implementada nativamente através do seu algoritmo de escalonamento *Credit Scheduler*. Utilizando esse algoritmo, para cada MV é atribuído um peso (*weight*) e o *cap*. Através destes parâmetros é possível limitar o uso da CPU e definir se o processo será com (WC-mode) ou sem (NWC-mode) conservação de trabalho. No modo com conservação do trabalho, caso a MV esteja consumindo a CPU sozinha, ela utilizará todos os recursos disponíveis e só obedecerá o limite quando outra MV estiver ativa e houver concorrência pela mesma CPU. No outro modo, mesmo que não haja concorrência pela CPU, a MV utilizará apenas o limite que foi dado [Cherkasova et al. 2007].

O KVM, que foi lançado oficialmente em 2007, ainda não possui uma solução nativa que permita limitar o uso da CPU. Apesar deste fato, as MVs criadas com KVM são processos do Linux e integram-se perfeitamente com o restante do sistema. Aproveitando disto, será proposta e avaliada uma solução utilizando a ferramenta cpulimit [Cpulimit 2011] para definir a percentagem da CPU que vai ser utilizada pela MV.

3.1. *Workflow*

O *workflow* a seguir descreve as atividades necessárias para garantir que a capacidade de processamento requisitada para uma MV seja mantida, independente da MF em que ela esteja alocada. A Figura 2 apresenta um diagrama de atividades para o *workflow* proposto, de forma a facilitar a visualização e entendimento das atividades.

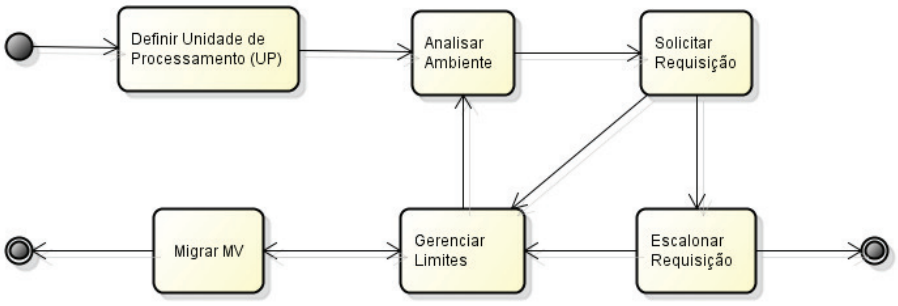


Figura 2. Atividades do *workflow* proposto

Atividade 1: Definir Unidade de Processamento (UP).
Em termos de GFLOPS, é necessário definir um valor padrão UP, para que ele seja a referência de capacidade de processamento para as MVs. Esse valor pode representar um percentual da capacidade de processamento de uma CPU. Na seção de experimentos será

apresentada uma estratégia para definição do valor da UP para a infraestrutura utilizada como *testbed*.

Atividade 2: Analisar Ambiente.
Tendo a UP definida, é necessário que a capacidade de processamento de todas as MFs da infraestrutura seja definida em termos da UP. *Benchmarks* para a medição de GFLOPS devem ser executados nas MVs de cada MF, e o poder de processamento de cada MF será descrito em termos da UP. Com isso, deve-se ter uma padronização na representação do poder de processamento.

Atividade 3: Solicitar Requisição.
Cada requisição de MV deve solicitar certa capacidade de processamento em função da quantidade de UPs.

Atividade 4: Escalonar Requisição.
As requisições devem ser escalonadas para uma MF conforme um algoritmo de escalonamento que agora é baseado na quantidade de UPs.

Atividade 5: Gerenciar Limites.
Como o valor de uma UP pode representar também uma fração do valor total de uma CPU, em uma MF específica, é preciso que haja um módulo gerenciador de limitações, que aplique o limite de CPU para cada MV, logo após esta ser alocada em uma MF.

Atividade 6: Migrar MV.
Dependendo das requisições e das políticas de consolidação de servidores da infraestrutura, pode ser necessário que uma MV seja migrada. Para isso, sabendo-se da quantidade de UPs solicitada para esta MV, é preciso analisar a MF fonte e destino da migração e fazer os ajustes necessários no limite da CPU, através do módulo gerenciador de limitações.

3.2. Limitando o uso da CPU com *cpulimit*

A ferramenta *cpulimit*, versão 1.1, é um programa de código aberto[5] utilizado para limitar o uso da CPU de um processo no Linux. Ele age no uso da CPU real e é capaz de se adaptar à carga total do sistema, de forma dinâmica e rápida.

O trabalho do *cpulimit* é realizado inteiramente no espaço do usuário, de modo que ele não interfere no escalonador do Linux. O processo é continuamente interrompido e reiniciado, enviando-lhe sinais *SIGSTOP*[6] e *SIGCONT*[7]. Os sinais são enviados pelo *cpulimit* nos momentos adequados, com base no limite especificado pelo usuário e as estatísticas do processo de leitura a partir de */proc*.

A Figura 3 apresenta um exemplo de execução da ferramenta *cpulimit*, onde a MV, cujo identificador de processo é igual a 4432, é limitada em 75% do uso de CPU. A

[5]Foi necessário realizar algumas alterações no código fonte da ferramenta para que ela funcionasse corretamente em máquinas com mais de uma CPU. Com a última versão estável só é possível utilizar limites de 0-100%, mas como os processadores utilizados possuem mais de um núcleo, a ferramenta não tratava corretamente os valores de entrada. Por exemplo, ao utilizar uma MV com duas VCPUs o processamento pode atingir até o valor 200%, e não foi possível definir limites acima de 100%. Sendo assim, a utilização de 1 CPU inteira e metade de outra não era possível. Após a correção, foi possível realizar tal limitação. O código fonte com as modificações será disponibilizado seguindo a licença da ferramenta, GPLv2.

[6]Nome do sinal emitido pelo sistema operacional, que faz com que um processo pare a sua execução enquanto não receber o sinal *SIGCONT*.

[7]Nome do sinal enviado para reiniciar um processo pausado por um sinal *SIGSTOP* ou *SIGTSTP*.

ferramenta, que neste exemplo foi executada em modo *verbose*, apresenta a quantidade de tempo em que o processo ficou ativo e dormindo.

```
pesquisa@no3:~$ sudo cpulimit -p 4432 -l 75 -v
Process 4432 detected

%CPU    work quantum    sleep quantum   active rate
76.38%   72934 us        27065 us       74.28%
75.26%   74347 us        25652 us       74.61%
75.25%   74467 us        25532 us       74.72%
```

Figura 3. Exemplo de execução da ferramenta *cpulimit* no modo verbose

4. Metodologia de Avaliação

Para avaliar a proposta apresentada, foram realizados diversos experimentos. Para isso, foram utilizados, como *testbed*, 4 máquinas do tipo A, 2 máquinas do tipo B (mesmas configurações apresentadas na Tabela 1), 1 *Storage* com 4TB de disco rígido, todos ligados a uma rede *gigabit*. A arquitetura pode ser vista na Figura 4.

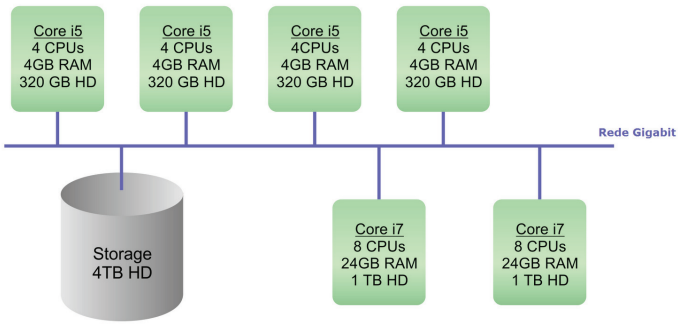


Figura 4. Infraestrutura do *testbed*

Todas as máquinas estão conectadas ao repositório de imagens do *Storage* através de NFS (*Network File System*).

Na próxima seção serão apresentados mais detalhes sobre o *benchmark* escolhido e, na seção seguinte, os experimentos realizados e discussão dos resultados.

4.1. *Intel Linpack benchmark*

O *Intel Linpack* é uma generalização do *benchmark Linpack 1000*. Ele resolve um sistema denso de equações lineares, e mede a quantidade de tempo que leva para fatorar e resolver o sistema. Após isso, ele converte o tempo em uma taxa de desempenho e testa os resultados por precisão. A generalização[8] está no número de equações que este *benchmark* pode resolver, que não se limita a 1000. Ele usa pivoteamento parcial para assegurar a precisão dos resultados e estes são informados em quantidade de operações de ponto flutuante por segundo(FLOPS), mas especificamente em termos de GFLOPS.

O *benchmark Linpack* foi selecionado por várias razões, dentre elas:

[8]Em todos os *benchmarks* executados, foram utilizados 7000 equações e uma matriz de dimensão 7000, como entrada.

- Álgebra linear densa é predominante em aplicações científicas. As necessidades das aplicações para computação científica são bem diferentes das aplicações voltadas para *Web*, compostas principalmente por requisições a banco de dados, que tem sido o foco dos provedores de Computação em Nuvem [Napper and Bientinesi 2009, Keahey et al. 2007]
- Algoritmos de computação intensiva. O *Linpack* fornece um bom limitante superior para o desempenho esperado das cargas de trabalho científico [Napper and Bientinesi 2009]
- O *Linpack* é semelhante ao utilizado pelo TOP 500 [TOP500.Org 2011]

5. Experimentação e Discussão dos Resultados

Nesta seção serão apresentados os experimentos relacionados às atividades do *workflow*. O Experimento 1 foi realizado para avaliar a ferramenta *cpulimit*. O Experimento 2 está relacionado à Atividade 1 e ao experimento motivacional. O Experimento 3 visa responder a pergunta ”O poder de processamento de 1 UP será mantido independente do servidor?”. Já o Experimento 4, a pergunta ”O poder de processamento de 2 UPs é o dobro do de 1 UP?”e o Experimento 5 visa responder a pergunta ”O poder de processamento de 2 UPs será mantido independente do servidor?”.

Em todos os testes, o *benchmark* foi executado 30 vezes em cada MV e os resultados agrupados para calcular as médias, limites inferiores e superiores, com confiabilidade de 95%.

5.1. Experimento 1

Este experimento visa avaliar a utilização da ferramenta *cpulimit* ao coletar o resultado do *benchmark* para a mesma MV, sujeita a limites variados de utilização de CPU. As Figuras 5 e 6 mostram a variação do poder de processamento, ao variar a fração da CPU alocada para a MV no Servidor A e B, respectivamente. As configurações dos servidores são as mesmas apresentadas na Tabela 1.

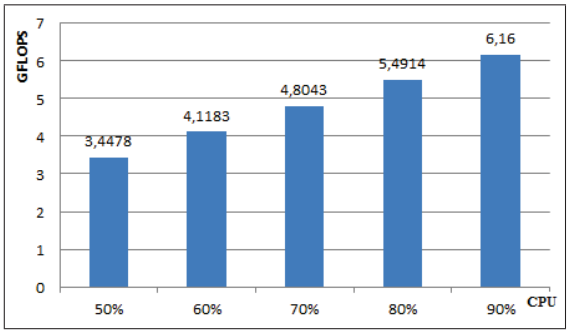


Figura 5. Variação do poder de processamento para o Servidor A

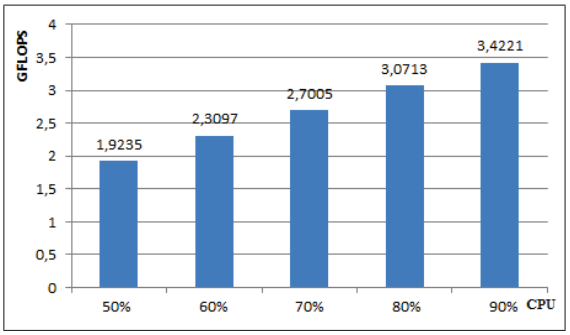


Figura 6. Variação do poder de processamento para o Servidor B

Foram utilizados limites de 50%, 60%, 70%, 80% e 90% para analisar a tendência de crescimento dos resultados, onde é possível notar uma curva de crescimento quase linear e proporcional ao valor de limitação. Constata-se que estes resultados reforçam a escolha da ferramenta *cpulimit*, uma vez que esta mantém a proporção para o valor indicado.

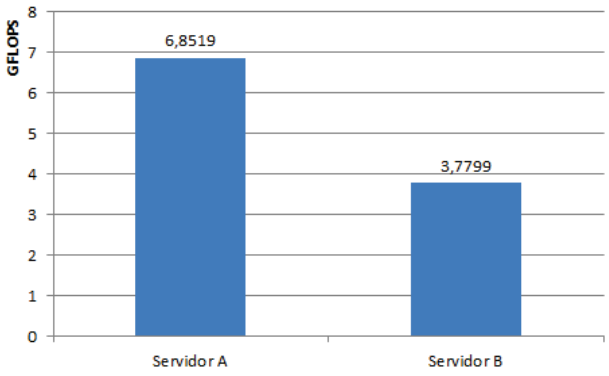
5.2. Experimento 2

Este experimento visa auxiliar o processo de definição da UP, que é a Atividade 1 do *workflow*, conforme a seguinte estratégia: 4 MVs foram criadas no Servidor A e 8 MVs no Servidor B, e o *benchmark* foi executado em todas as MVs ao mesmo tempo. As MVs criadas tem a configuração apresentada na Tabela 2 e a quantidade de MVs para cada MF foi escolhida devido à quantidade de CPUs que as MFs possuem. Como as máquinas do tipo B possuem a tecnologia de *Hyper-Threading*, o sistema operacional identifica 8 CPUs, diferente das máquinas do tipo A, que não possuem essa tecnologia e apresentam 4 CPUs.

VCPU	1
Memória	1Gb
Sistema Operacional	Ubuntu Server 10.10 64bits
Kernel	2.6.35-25

Tabela 2. Configuração da máquina virtual

Para este experimento, os resultados e limites inferiores e superiores são os mesmos apresentados no experimento motivacional e podem ser vistos na Figura 7.



	Limite Inferior	Limite Superior
Servidor A	6,8334	6,8703
Servidor B	3,7608	3,799

Figura 7. Resultados em termos de GFLOPS para os Servidores A e B

Uma vez definido o poder de processamento de uma MV com 1 VCPU para cada máquina da infraestrutura, conclui-se a Atividade 1 ao definir o valor da UP. Para simplificar, a UP considerada foi 3,77 GFLOPS, exatamente 100% de uma CPU do Servidor B e, aproximadamente, 55% de uma CPU do Servidor A.

5.3. Experimento 3

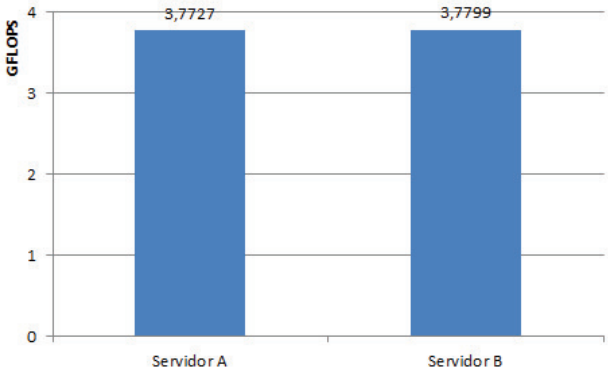
A Atividade 2 já foi parcialmente realizada devido à estratégia utilizada para definição da UP. Com os resultados do Experimento 2 foi possível definir o poder de processamento de cada MF em termos de UP. Como o Servidor A tem 4 CPUs e cada uma tem poder de processamento 6,8 GFLOPS, tem-se que esta máquina possui aproximadamente 7,2 UPs. Já o Servidor B equivale a 8 UPs, uma vez que tem 8 CPUs e cada uma delas tem poder de processamento de 3,77 GLFOPS. Estes dados foram compilados na Tabela 3.

Com as Atividades 1 e 2 completas, este experimento visa responder à seguinte pergunta:
- O poder de processamento de 1 UP é igual independente de MF?

	Servidor A	Servidor B
CPUs	4	8
Poder de processamento para cada CPU	6,85 GFLOPS	3,77 GFLOPS
Total de poder de processamento	27,4 GFLOPS	30,16 GFLOPS
Total de UPs	7,2	8

Tabela 3. Resultados compilados a partir do Experimento 2

Observando a Figura 8, percebe-se que o resultado do *benchmark* é praticamente igual para 1 UP, independente de servidor em que a MV esteja alocada. Os limites inferiores e superiores são apresentados, com confiabilidade de 95%.



	Limite Inferior	Limite Superior
Servidor A	3,7647	3,7806
Servidor B	3,7608	3,799

Figura 8. Resultado do *benchmark* para avaliar se 1 UP do Servidor A equivale a 1 UP do Servidor B

Este resultado era o esperado, uma vez que a ideia chave da proposta apresentada é manter o poder de processamento independente da máquina física.

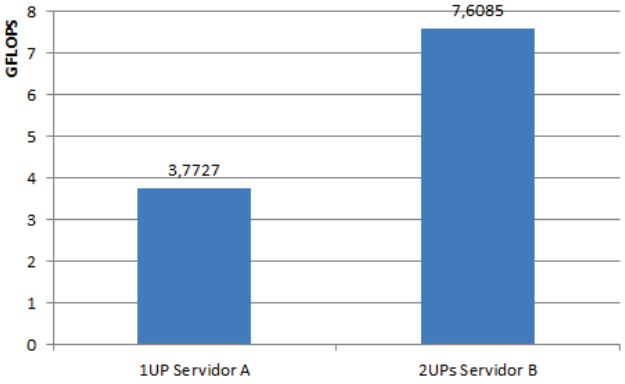
5.4. Experimento 4

Este experimento visa avaliar a relação entre o poder de processamento de 2UPs e 1UP. Para atingir o poder de processamento de 2 UPs, uma MV com 2 VCPUs foi criada no Servidor B. As configurações das MVs são as mesmas da Tabela 2, com exceção da quantidade de VCPUs. Na Figura 9, observa-se que 2 UPs possuem o poder de processamento praticamente igual ao dobro do poder de processamento de uma UP. Os limites inferiores e superiores também são apresentados, com confiabilidade de 95%.

Estes resultados eram esperados, uma vez que é percebida uma curva de crescimento quase linear nas Figuras 5 e 6, que mostra o poder de processamento para 50%, 60%,..., 90% da CPU, tanto no servidor do tipo A, quanto no servidor do tipo B. Ao definir uma UP, é garantido que independente do servidor, se houver o correto ajuste da CPU, garante-se o mesmo poder de processamento.

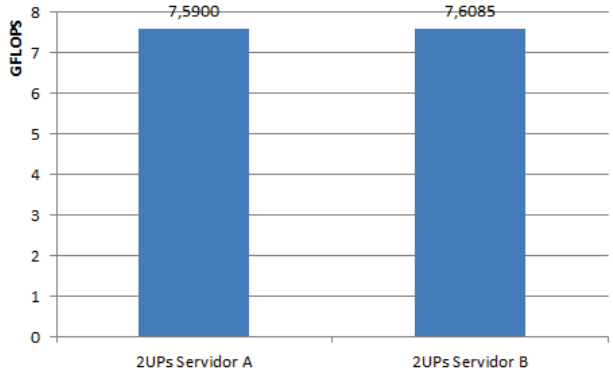
5.5. Experimento 5

Este experimento visa comparar o poder de processamento de 2 UPs no Servidor A com 2 UPs no servidor B. Foi observado que os resultados são praticamente iguais. Na Figura 10 podem ser vistos os resultados e os limites inferiores e superiores, com confiabilidade de 95%.



	Limite Inferior	Limite Superior
1 UP A	3,7647	3,7806
2 UPs B	7,5992	7,6178

Figura 9. Resultado do *benchmark* para avaliar se 2 UPs equivalem ao dobro de 1 UP



	Limite Inferior	Limite Superior
Servidor A	7,5811	7,6403
Servidor B	7,5992	7,6178

Figura 10. Resultado do *benchmark* para avaliar se 2 UPs do Servidor A equivalem a 2 UPs do Servidor B

É importante notar que, neste experimento, a MV criada no Servidor A possuía 2 VCPUs, mas limitada a 110% de processamento, uma vez que 1 UP neste servidor equivale a 55% de uma CPU. Já a MV criada no servidor B possuía 2 VCPUs e estas eram usadas completamente, seguindo a definição de UP para este servidor, realizada na Atividade 1.

5.6. Conclusão dos Experimentos

Os resultados apresentados mostram que é possível utilizar a ideia proposta de definir uma Unidade de Processamento, apoiada à limitação do uso da CPU com a ferramenta *cpulimit*, para possibilitar que o poder de processamento permaneça no mesmo nível independente da máquina física.

As atividades de criação e migração de MVs podem ser validadas com os resultados apresentados nas Figuras 8 e 10, uma vez que foi avaliada a execução do *benchmark* em MV alocadas em diferentes máquinas físicas e os ajustes no limite do uso da CPU foram realizados. Entretanto, essas atividades serão melhor exploradas em trabalhos futuros.

Uma das vantagens dessa abordagem, é que, ao utilizar a ferramenta *cpulimit*, está sendo implantada uma solução sem conservação do trabalho, pois mesmo que não exista outra MV utilizando a CPU, será mantida a limitação. Esta é uma das formas de prover

isolamento de desempenho e garantia dos recursos.

6. Trabalhos Relacionados

Há diversos trabalhos que exploram a análise de desempenho de MVs em ambientes de Computação em Nuvem, utilizando diversos *benchmarks*. Em [Deshane et al. 2008], os autores apresentam uma análise quantitativa do Xen e KVM, comparando o desempenho geral ao apresentar resultados de *benchmarks* sobre o isolamento de desempenho e escalabilidade. Os resultados quanto ao desempenho da CPU são praticamente iguais para Xen e KVM.

[Iosup et al. 2008] analisaram a utilidade dos serviços de Computação em Nuvem atuais para computação científica, ao fazer testes na plataforma Amazon EC2. Eles utilizam um *benchmark Linpack* para avaliação. Outro trabalho que utiliza o *Linpack* está em [Napper and Bientinesi 2009], onde os autores investigaram se utilizando a Amazon EC2 era possível criar um *cluster* que tivesse poder de processamento para entrar na lista TOP 500.

Outros trabalhos exploram a limitação da capacidade de processamento, como [Baruchi and Midorikawa 2008]. Este trabalho analisa a influência de algoritmos de escalonamento na vazão da rede em um ambiente virtualizado. É estudado o algoritmo *Credit Scheduler* do Xen para analisar sua influência sobre o desempenho do I/O da rede. [Lee et al. 2010] propõem um novo escalonador que gerencia a latência, implementado sobre o Xen, de forma que ele se torne mais adaptável para aplicações em tempo real. [Bai et al. 2010] apresentam um escalonador orientado a tarefas para um sistema de máquinas virtuais, baseado em técnicas de inferência.

Todos esses trabalhos utilizaram o Xen, e manipularam o algoritmo padrão de escalonamento, *Credit Scheduler*, para limitar/ajustar o uso da CPU pelas MVs.

Diversos trabalhos são realizados sobre o *hypervisor* Xen. Um trabalho que utiliza o KVM e a ideia de limitar o uso de CPU está em [Sangpetch et al. 2010]. Os autores apresentam um sistema de controle adaptativo, que automatiza a tarefa de ajustar a alocação de recursos e a manutenção dos Objetivos de Nível de Serviço (SLOs). Este trabalho foca em manter o tempo de resposta esperado para aplicações Web, onde a alocação e o limite de CPU para as MVs são ajustados constantemente através do uso da ferramenta *control groups* (*cgroups*). Apesar de haver algumas semelhanças entre este trabalho e o proprosto neste artigo, os autores utilizam uma solução diferente para realizar a limitação do uso da CPU. Outra diferença é que seu foco são as aplicações Web, então os parâmetros para ajustar o uso da CPU são diferentes dos nossos, sem contar que, com o ajuste dinâmico realizado, o isolamento de desempenho pode ser comprometido.

Outros trabalhos propõem modelos para garantir a Qualidade de Serviço (QoS). [Wood et al. 2008] apresenta um modelo para estimar os recursos requisitados por uma aplicação quando elas são transferidas para um ambiente virtualizado. Seu objetivo é eliminar o processo manual, que eles julgam ser propenso a erros.

O trabalho [Song et al. 2009] discute sobre um modelo para consolidação de servidores baseados em máquinas virtuais, modelando a interação entre a chegada de requisições no servidor com requisitos de QoS, e a capacidade de fluxo entre serviços simultâneos, baseado na teoria das filas, para a previsão do escalonamento de MVs no

datacenter. Seu estudo de caso utilizou o Xen, e foi implementado um protótipo para a validação desse modelo através dos *benchmarks* TPC-W and SPECweb2005.

Diversos trabalhos utilizaram diferentes *benchmarks* e *hypervisors* para testar a infraestrutura e buscar alcançar certos níveis de QoS, mas poucos trabalharam com o *hypervisor* KVM. Até onde foi pesquisado, nenhum outro trabalho procurou ajustar o uso da CPU a fim de garantir que o desempenho da MV, em termos de processamento, seja mantido, independente da máquina física em que ela foi alocada, e de/para onde ela foi migrada.

7. Conclusão e Trabalhos Futuros

Este trabalho teve como objetivo a criação de um *workflow* que definisse como a capacidade de processamento poderia ser utilizada para a criação de Unidades de Processamento (UPs) e como elas seriam utilizadas para alocar a capacidade de processamento às Máquinas Virtuais (MVs), independente de qual Máquina Física (MF) elas estivessem alocadas.

Durante o projeto de implementação foi identificado um problema relacionado ao KVM, que é o fato de que ele não possui uma funcionalidade nativa para limitar o uso da CPU das MVs, o que impossibilitava a criação das UPs. Foi proposta uma solução temporária para a resolução deste problema, ao utilizar a ferramenta *cpulimit*.

Algumas das atividades do *workflow* foram avaliadas em laboratório, com exceção do escalonamento das requisições de MVs, e a migração de MVs, sendo estas duas atividades trabalhos futuros do projeto. A implementação do modelo e integração ao *OpenNebula*, que está instalado na infraestrutura, será a validação desses dois pontos. O escalonador implementado será incorporado ao *OpenNebula*, para que a UP seja utilizada como unidade de escalonamento na alocação de MVs. A validação do escalonador de requisições e a avaliação da migração das MVs são trabalhos futuros relacionados ao *OpenNebula*.

Como existem vários trabalhos semelhantes que utilizam o Xen, outro trabalho futuro será a comparação dos resultados entre Xen e KVM. Várias ferramentas foram utilizadas neste trabalho, como o *cpulimit* para limitar o processamento, e o *Linpack* como *benchmark*. Outros trabalhos futuros seriam a identificação e avaliação de outros utilitários para a realização dessas atividades.

Salienta-se que o trabalho proposto está sendo desenvolvido no contexto do projeto INCT-MACC (Instituto Nacional de Ciência e Tecnologia - Medicina Assistida por Computação Científica)[INCT-MACC 2011], cujo GREat é um dos laboratórios associados.

Agradecimentos

Os autores agradecem ao projeto INCT-MACC (processo CNPq 573710/2008-2) e à FUNCAP (Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico), pelo apoio financeiro.

Referências

Bai, Y., Xu, C., and Li, Z. (2010). Task-aware based co-scheduling for virtual machine system. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 181–188, New York, NY, USA. ACM.

Baruchi, A. and Midorikawa, E. (2008). Influência do algoritmo de escalonamento credit scheduler no desempenho de rede. In *Workshop de Sistemas Operacionais (WSO), SBC 2008*.

Chen, P. M. and Noble, B. D. (2001). When virtual is better than real [operating system relocation to virtual machines]. In *Hot Topics in Operating Systems, 2001. Proceedings of the Eighth Workshop on*.

Cherkasova, L., Gupta, D., and Vahdat, A. (2007). Comparison of the three cpu schedulers in xen. *SIGMETRICS Perform. Eval. Rev.*, 35:42–51.

Cpulimit (2011). Cpu usage limiter for linux. `http://cpulimit.sourceforge.net/`.

Deshane, T., Shepherd, Z., Matthews, J., Ben-Yehuda, M., Shah, A., and Rao, B. (2008). Quantitative comparison of Xen and KVM. In *Xen summit*, Berkeley, CA, USA. USENIX association.

INCT-MACC (2011). Instituto nacional de ciência e tecnologia - medicina assistida por computação científica. `http://www.lncc.br/macc/`.

Iosup, R., Ostermann, S., Yigitbasi, N., Prodan, R., Fahringer, T., and Epema, D. (2008). An early performance analysis of cloud computing services for scientific computing. *TU Delft, Tech. Rep., Dec 2008, [Online] Available*.

Keahey, K., Freeman, T., Lauret, J., and Olson, D. (2007). Virtual workspaces for scientific applications. *Journal of Physics: Conference Series*, 78(1):012038.

Lee, M., Krishnakumar, A. S., Krishnan, P., Singh, N., and Yajnik, S. (2010). Supporting soft real-time tasks in the xen hypervisor. In *Proceedings of the 6th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, VEE '10, pages 97–108, New York, NY, USA. ACM.

Mc Evoy, G. V., Schulze, B., and Garcia, E. L. M. (2011). Performance and deployment evaluation of a parallel application on a private cloud. *Concurrency and Computation: Practice and Experience*, pages n/a–n/a.

Napper, J. and Bientinesi, P. (2009). Can cloud computing reach the top500? In *Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop*, UCHPC-MAW '09, pages 17–20, New York, NY, USA. ACM.

OpenNebula (2011). The open source toolkit for cloud computing. `http://opennebula.org/`.

Sangpetch, A., Turner, A., and Kim, H. (2010). How to tame your vms: an automated control system for virtualized services. In *Proceedings of the 24th international conference on Large installation system administration*, LISA'10, pages 1–16, Berkeley, CA, USA. USENIX Association.

Song, Y., Zhang, Y., Sun, Y., and Shi, W. (2009). Utility analysis for internet-oriented server consolidation in vm-based data centers. In *Cluster Computing*, pages 1–10.

Sonnek, J. and Chandra, A. (2009). Virtual Putty: Reshaping the Physical Footprint of Virtual Machines. In *Proc. of Workshop on Hot Topics in Cloud Computing (HotCloud'09)*.

TOP500.Org (2011). Top 500 supercomputer sites. `http://www.top500.org/`.

Wood, T., Cherkasova, L., Ozonat, K., and Shenoy, P. (2008). Profiling and modeling resource usage of virtualized applications. In *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, Middleware '08, pages 366–387, New York, NY, USA. Springer-Verlag New York, Inc.

Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18.