Identificação de gargalos de desempenho em ambientes virtuais para uso em computação em nuvem

Carlos H. G. Ferreira[1], João B. Ribeiro[1], Wellington D. B. Júnior[1], Júlio C. Estrella[1], Dionísio M. L. Filho[1], Maycon L. M. Peixoto[2]

[1]Instituto de Ciências Matemática e de Computação - Universidade de São Paulo
Avenida Trabalhador São - Carlense, 400 - Centro - São Carlos - SP

[2]Departamento de Administração - Universidade Federal de Juiz de Fora
Rua Israel Pinheiro, 2000 - Bairro Universitário - Governador Valadares - MG

chgferreira@usp.br, dionisio@icmc.usp.br, maycon.leone@ufjf.edu.br

Abstract. *Cloud computing uses virtualization to manage and handle with resources with no interaction to users raising the performance of servers and/or physical resources. Virtualization allows many users have independent machines in a same server, bringing benefits as resource isolation. However, depending on a workload in a virtual machine, this one can put down the performance of another virtual machines. Thus, is essential to identify when these interferences occur and how to avoid them. Our proposal is to use three basic types of workload (memory, CPU and I/O) to verify among the KVM or Xen what is the most suitable for the activities of the cloud. We find that there is no good virtualizer but rather a virtualizer suitable for each type of workload.*

Resumo. *A computação em nuvem se apoia na virtualização para fazer a manutenção e gerência de recursos de maneira transparente aos clientes de forma a aumentar o desempenho dos servidores ou recursos físicos. O uso de virtualização permite que vários clientes tenham máquinas independentes dentro de um mesmo servidor físico trazendo benefícios de isolamento de recursos, principalmente. No entanto, dependendo da carga de trabalho imposta a uma máquina virtual, a mesma pode gerar degradações de desempenhos em outras máquinas virtuais. Sendo assim, é fundamental identificar quando essas interferências acontecem e como evitá-las. Nossa proposta foi utilizar três tipos básicos de carga de trabalho (memória, cpu e E/S) para verificar, dentre o Virtualizador KVM ou Xen, qual o mais adequado para as atividades da nuvem. Nós verificamos que não há um virtualizador bom e sim um virtualizador adequado para cada tipo de carga de trabalho.*

1. Introdução

A computação em nuvem é uma opção para a alocação ou desenvolvimento de serviços de forma distribuída devido a sua capacidade de fornecimento de recursos por demanda [Khan et al. 2014], permitindo que empresas contratem serviços de acordo com suas necessidades, utilizando os recursos contratados para processar e armazenar os dados de forma eficiente, econômica, com alta escalabilidade e fácil acesso [Zhang et al. 2010].

Essa aquisição de recursos ou serviços é apoiada pelos acordos em nível de serviço (SLA - *Service Level Agreement*), onde o desejável pelos clientes é o desempenho máximo

de suas aplicações. Em contrapartida, os provedores de serviços e recursos priorizam minimizar os custos operacionais e de infraestrutura da nuvem [Zhang et al. 2010]. Essa atividade é apoiada pelo uso de virtualização, que permite manipular e gerenciar máquinas virtuais (MV) tornando assim transparente a manutenção dessas máquinas aos clientes e permitindo aos provedores alocar mais de uma máquina virtual em uma mesma máquina física, aumentando assim o uso dos recursos disponíveis [Head et al. 2010]. Implantações típicas de servidores originam uma média de utilização de 10% a 15% da capacidade total, além da geração de custos maiores devido a manutenção, suporte, gerenciamento e consumo de energia [VMware 2014]. Por isso, a virtualização é fundamental para a computação em nuvem, tanto pela transparência no gerenciamento das MVs como no aumento do uso de recursos físicos disponíveis [Daniels 2009].

Por meio da garantia que a virtualização oferece em executar diversos sistemas operacionais simultaneamente sobre um mesmo *hardware*, existe uma evolução do gerenciamento dos recursos e do ambiente [Zaghloul 2013]. Em que possibilita atingir um melhor desempenho afim de atender as aplicações de maior custo computacional ou que demandam maior disponibilidade e segurança [Santos and Charão 2008].

A virtualização utiliza um *hypervisor* que funciona como um gerenciador de máquinas virtuais e utilização de recursos. Entretanto, apesar da atual evolução desses *hypervisores*, as múltiplas MVs podem interferir nos recursos já estabelecidos para as VMs em uma mesma máquina física, o que pode violar as SLAs estabelecidas entre clientes e provedores [Srinivasan et al. 2012].

Sendo assim, este artigo tem dois principais objetivos: o primeiro é a apresentação de avaliações de desempenho com tipos diferentes de cargas de trabalho visando identificar quanto cada máquina virtual influencia em outra quando está executando no máximo da capacidade definida para ela. O segundo objetivo é a definição de um cenário de experimentos totalmente reproduzível, onde outros pesquisadores poderão identificar novas características referentes ao estudo apresentado neste artigo, como, por exemplo, a definição de cenários científicos distribuídos.

As demais seções que compõe este artigo são compostas assim: a Seção 2 apresenta os trabalhos relacionados com o objetivo de demonstrar as contribuições obtidas. Na Seção 3 é descrita a metologia utilizada. Já a Seção 4 discute a definição do ambiente, a descrição das cargas de trabalho definidas como base para este artigo. Por fim, a Seção 5 elabora as conclusões e direcionamentos futuros.

2. Trabalhos Relacionados

[Koh et al. 2007] realizaram um trabalho avaliando a interferência causada por várias MVs hospedadas em única máquina hospedeira com cargas de trabalho simuladas que caracterizassem diferentes aplicações do mundo real. A partir de métricas de desempenho e tempo de execução em um ambiente utilizando o virtualizador Xen foram identificadas as interferências geradas de acordo com as diferentes aplicações e as cargas de trabalhos. Com isto, propuseram um modelo matemático experimental com erro médio de 5% capaz de prever o desempenho e a interferência de uma nova aplicação de acordo com a característica da carga de trabalho.

[White and Pilbeam 2010] destacam a degradação do desempenho de MVs relacionando os cuidados com as limitações físicas e interferências geradas na virtualização.

São apresentados os métodos de virtualização atuais juntamente com as características e arquiteturas. Foi concluído que cada metodologia de virtualização apresenta os respectivos benefícios e que a seleção de um modelo de virtualização se deve às políticas das aplicações e tecnologias que serão trabalhadas.

[Matthews et al. 2007] utilizaram diferentes métodos de virtualização para realização de um *benchmark* diversificando os tipos de carga de trabalho afim de quantificar a interferência que uma MV pode causar em outras MVs hospedadas em uma mesma máquina física. Para realização dos testes foram utilizadas pelo menos um virtualizador para representar os métodos de virtualização apresentados. Os resultados apresentados mostraram superioridade da paravirtualização e da virtualização completa.

[Somani and Chaudhary 2010] mostraram a importância dos provedores de nuvem garantirem os SLAs dos parâmetros e suas restrições como inatividade, exigência de CPUs, largura de banda de rede e espaço em disco. Alguns pontos importantes de serviços oferecidos em nuvem foram discutidos, como o isolamento de desempenho que trata o quesito de uma MV não afetar o desempenho de outras hospedeiras na mesma máquina hospedeira. Em outro ponto eles citam a associação de aplicações em virtualização, em que aplicações que demandam os mesmos recursos irão disputá-los aumentando a interferência.

De acordo com o que foi exposto nos trabalhos relacionados, podemos considerar os seguintes pontos em relação à nossa proposta: Segundo [Koh et al. 2007], as características específicas do virtualizador devem ser levadas em consideração. No entanto, nem sempre a interferência é a mesma para os virtualizados em uso. Porém, é importante que o virtualizador seja levado em questão. Apesar de [White and Pilbeam 2010] considerar limitações entre *hosts* e máquinas virtuais, análises com diferentes cargas de trabalhos não foram consideradas. A contribuição deste artigo em relação ao apresentado por [White and Pilbeam 2010] é no uso de diferentes cargas de trabalhas em duas dessas metodologias.

Em [Matthews et al. 2007] foi importante observar quais virtualizadores deveriam ser estudados para conclusão de uma melhor análise, sendo assim, este trabalho se baseou nas duas técnicas no qual os mesmos concluíram apresentar melhores desempenho. Conforme [Somani and Chaudhary 2010] apresentou, é inerente que MVs que consomem recursos em comum tendem a interferir de maneira mais significante umas nas outras. No entanto, é fundamental analisar a carga de trabalho, uma vez que cada carga de trabalho tem um foco diferente em recursos.

Os trabalhos citados apresentam de maneira geral a ocorrência de interferência causada por MVs hospedadas em única máquina hospedeira, porém é preciso compreender como os diferentes tipos de carga de trabalho limitam as MVs levando em consideração os recursos definidos para as mesmas, e o quanto os fatores envolvidos influenciam no desempenho. A próxima seção descreve a técnica utilizada neste trabalho para resolver esse problema abordado.

3. Metodologia e cenários utilizados

Para a correta definição do cenário, bem como da quantidade de experimentos a serem gerados e executados, foi utilizado o livro *The Art of Computer Systems Performance*

Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling [Jain 1991]. Nesse texto é apresentado como definir um conjunto de experimentos, bem como isolar o ambiente para que os resultados sejam pertinentes apenas ao que está sendo analisado sem sofrer influências de características externas.

O modelo aqui utilizado é o fatorial completo. Nesse modelo é necessário identificar o que são fatores e o que são níveis e a partir de então formar os conjuntos de experimentos para obter as variáveis de resposta – que são os resultados a serem analisados. Inicialmente foram definidos os fatores e seus respectivos níveis conforme mostra a Tabela 1.

Tabela 1. Fatores e Níveis

Fator	**Níveis**	
Virtualizador	XEN	KVM
Memória	512MB	1024MB
vCPU	1	2
Número de MVs	3	6

Este modelo utiliza um arranjo de 2^4, que é o número de níveis elevado ao número de fatores. Este arranjo é alcançado a partir da Equação 1.

$$\begin{aligned} y &= q_0 + q_A x_A + q_B x_B + q_C x_C + q_D x_D + q_{AB} x_{AB} + q_{AC} x_{AC} + q_{AD} x_{AD} \\ &+ q_{BC} x_{BC} + q_{BD} x_{BD} + q_{CD} x_{CD} + q_{ABC} x_{ABC} + q_{ABD} x_{ABD} q_{ACD} x_{ACD} \\ &+ q_{BCD} x_{BCD} + q_{ABCD} x_{ABCD} \end{aligned} \quad (1)$$

Substituindo-se os valores dos experimentos, obtêm-se os valores de $q_A, q_B, q_C, q_D, q_{AB}, q_{AC}, q_{AD}, q_{BC}, q_{BD}, q_{CD}, q_{ABC}, q_{ABD}, q_{ACD}, q_{BCD}, q_{ABCD}$ como mostra a Equação 2, onde é calculado o valor de q_0.

$$\begin{aligned} q_0 = 1/16 * (&y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 + y_9 \\ &+ y_{10} + y_{11} + y_{12} + y_{13} + y_{14} + y_{15} + y_{16}) \end{aligned} \quad (2)$$

A partir dos valores obtidos pode-se determinar a soma dos quadrados. A variação total ou Soma Total dos Quadrados (SST) é dada pela Equação $SST = \sum_{i,j} (y_{ij} - \bar{y})$. Nesta equação, $\bar{y}$ representa a média das respostas de todas as repetições de todos os experimentos. Na simulação realizada o SST é dado por: $SST = 2^4(q_A^2 + q_B^2 + q_C^2 + q_D^2 + ... + q_{ABCD}^2)$. Por meio da utilização do modelo de regressão, a SST fornecerá a variação total das variáveis de resposta e a influência de cada fator e suas interações. Para obter a influência de um determinado fator, por exemplo o fator A, é necessário utilizar *y = SSA/SST*, onde *SSA* = $2^4 * q_A^2$. O projeto de experimentos final é apresentado na Tabela 2.

Após o planejamento de experimentos, os ambientes foram desenvolvidos utilizando o Xen e o KVM levando em consideração os fatores e níveis definidos, seguindo

Tabela 2. Combinação dos Fatores e Níveis

Exp.	Virtualizador	Memória MV	vCPU MV	Nº MVs
1	Xen	512 MB	1	3
2	Xen	512 MB	1	6
3	Xen	512 MB	2	3
4	Xen	512 MB	2	6
5	Xen	1024 MB	1	3
6	Xen	1024 MB	1	6
7	Xen	1024 MB	2	3
8	Xen	1024 MB	2	6
9	KVM	512 MB	1	3
10	KVM	512 MB	1	6
11	KVM	512 MB	2	3
12	KVM	512 MB	2	6
13	KVM	1024 MB	1	3
14	KVM	1024 MB	1	6
15	KVM	1024 MB	2	3
16	KVM	1024 MB	2	6

o planejamento fatorial completo. As configurações das MVs foram estabelecidas conforme a Tabela 3. Exceto a memória RAM alocada na MV, que é estabelecida de acordo com o nível do fator Memória, apresentado na Tabela 2. De maneira semelhante tem-se o fator número de vCPUs.

Tabela 3. Configurações das Máquinas Virtuais

Máquina Virtual		
Virtualizador	XEN	KVM
Tamanho do Disco	5 GB	5 GB
S.O.	Ubuntu 12.04 LTS x64	Ubuntu 12.04 LTS x64

A arquitetura utilizada para execução das MVs se resume em um servidor, a Tabela 4 apresenta maiores detalhes e informações.

Tabela 4. Configurações do Servidor

Servidor	
Memória	12GB (3x4GB) 1333MHz DDR3
Processador	Intel Core i5-3470 3.20GHz
Número de Núcleos	4
Número de Threads	4
Cache	6MB
Disco	SATA 1TB 3.5 7.200rpm
Sistema Operacional	Ubuntu 12.04 LTS x64

A cada experimento e após cada replicação, o sistema era reiniciado, evitando que a memória cache influenciasse nos resultados. O isolamento também foi prezado na ordem de execução dos experimentos, no qual replicações do mesmo experimento não poderiam ser executadas sequencialmente.

4. Carga de trabalho e variáveis de resposta

Com todo o ambiente para experimentação definido, a próxima etapa foi a realização do *benckmark* em cada um dos dezesseis experimentos utilizando a ferramenta *Phoronix* para aferir as variáveis de resposta. O *Phoronix* por sua vez possui um conjunto de pacotes que permitem executar o *benckmarks* de acordo com um recurso específico (CPU, E/S, Memória, etc). Neste projeto foram utilizados os seguintes *benchmarks*[1]:

- Apache: *Benchmark* de sistema, que por característica demanda diferentes tipos de recursos, entre eles, CPU e E/S. Retornando o número de requisições atendidas por segundo.
- SmallPT: Simula um renderizador de imagens em C++. Avalia o processamento da CPU retornando o tempo de execução para renderização de uma imagem.
- Postmark: O Postmark é projetado para simular testes com pequenos arquivos semelhantes a transações que ocorrem em servidores web e correio. Os testes realizam 25 mil operações com 500 arquivos simultâneos de tamanho que varia entre 5Kb e 512Kb avaliando E/S e retornando o número de transações por segundo.
- RAMSpeed: O RAMSpeed é um *benchmark* de memória que avalia a performance da memória RAM através de diferentes operações aritméticas, mostrando a quantidade de *mega bytes* por segundo.

Durante a aplicação de cada *benchmark* foi definido que, para cada variável de resposta, doze replicações seriam realizadas, sendo as duas primeiras descartadas, considerando o *warm-up*. Foram calculados o intervalo de confiança com 95% de acordo com a tabela *T-Student*. Por fim, os resultados foram submetidos ao modelo de regressão, apresentado por Jain [Jain 1991], para verificar as influências de fatores.

5. Resultados

5.1. Apache

Para o *benchmark* de sistema Apache, é possível observar o resultado da influência na Figura 1.

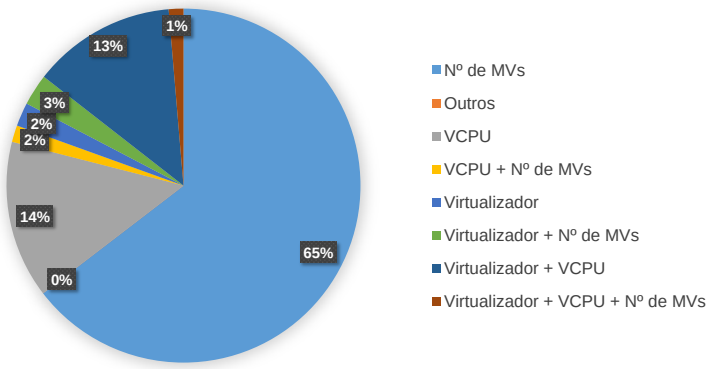

Figura 1. Influência no desempenho do sistema Apache

O fator relacionado a quantidade MVs representa 64,55% de influência sobre o desempenho, sendo quem mais influenciou. Isso se da pelo alto nível na degradação no

[1]As informações sobre os testes foram retiradas em `http://openbenchmarking.org/tests/pts`

desempenho causado pela quantidade de MVs co-hospedadas. A medida que a quantidade de MVs aumentam consequentemente aumenta a disputa dos recursos da máquina hospedeira. Em segundo lugar, o fator que trata a quantidade de vCPUs alocadas por MV é o mais influente, representando 14,37%. O que se percebe é que apesar do Apache ser um *benchmark* de sistema e exigir diferentes tipos de recursos, a CPU é um dos recursos que mais influencia no resultado. O terceiro fator mais influente se da pela interação entre dois fatores, sendo eles os fatores Virtualizador e vCPU com 13,11%.

Ainda em fatores que se interagem, tem-se os fatores virtualizador e a quantidade de MVs com 2,86% de influência, devido ao KVM apresentar um melhor desempenho em alguns cenários considerando a quantidade MVs, desde que não seja considerado o fator vCPU. Por fim, tem-se o fator virtualizador apresentou 2,15% da influência total.

Os resultados dos experimentos considerando o planejamento de experimento apresentado no capítulo anterior para o sistemas Apache são apresentados na Figura 2, que mostra a comparação entre médias levando em consideração os virtualizadores em questão.

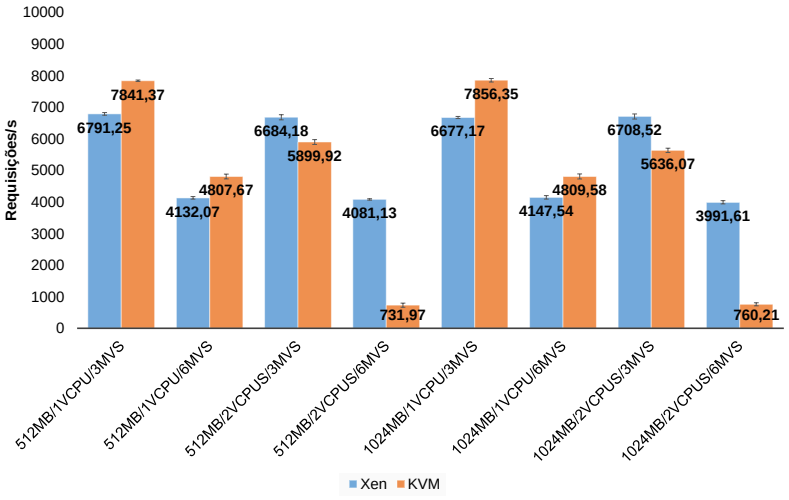

Figura 2. Resultado desempenho dos virtualizadores com o sistema Apache.

Os experimentos em que o número de MVs é igual a 6 apresentam desempenho inferior independente do virtualizador. Outro ponto relevante, são os experimentos com 2 vCPUs e 6 MVs. Nestes, a diferença de resultado do KVM é extremamente inferior aos demais. Nota-se, que são 6 MVs, cada uma com 2 vCPUs, totaliza-se 12 vCPUs em concorrência. Isso em uma máquina hospedeira que possui apenas 8 núcleos (núcleos + *threads*). O que se percebe é que há uma diferença na forma em que os virtualizadores escalonam a CPU da máquina hospedeira.

Ao investigar os algoritmos de escalonamentos da CPU em ambos os virtualizadores, de acordo com [Arthur Baruchi 2008], há diferença entre os algoritmos de escalonamento padrão do Xen e o KVM adotados para a máquina hospedeira, conforme apresentado na Tabela 5.

Enquanto o Xen utiliza o escalonamento por crédito, em que a cada MV é adicionado um peso que por padrão são iguais e na máquina hospedeira este mesmo peso é diferenciado, representando uma prioridade maior [Lee et al. 2010], o KVM adota o

Tabela 5. Algoritmos de escalonamento da CPU na máquina hospedeira

Escalonador/Virtualizador	Xen	KVM
Host	Escalonamento por crédito	CFS

algoritmo de escalonamento por CFS *(Completely Fair Scheduler)*, em que um *quantum* estabelecido em milissegundos determina o tempo em que o processo ocupa o processador de forma justa [Jiang et al. 2009]. Neste caso em específico, em que o número de vCPUs é elevado e é preciso escalonar de maneira precisa a CPU [Niyato 2011], e o virtualizador KVM se mostrou bem inferior mediante a essa situação.

5.2. Postmark

No *benchmark* Postmark os resultados das influências são demonstrados na Figura 3.

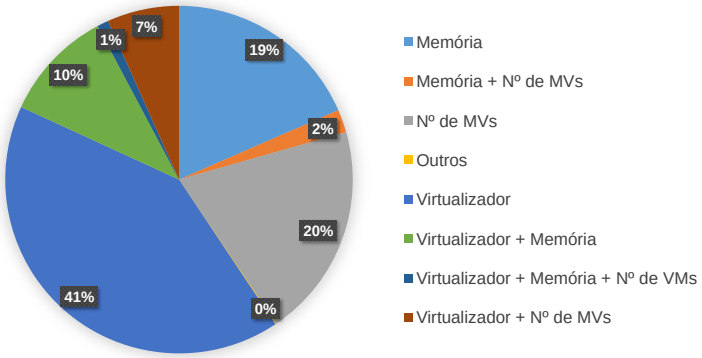

Figura 3. Influência E/S

É possível observar na figura 3 uma alta influência de 41,17% no fator virtualizador, isso ocorre devido a uma discrepância nos resultados. Quando o Xen apresenta ser 100% superior ao KVM na maioria dos cenários. O segundo fator mais influente é a quantidade de MVs, com aproximadamente 19,94%, representando novamente a degradação causada em MVs co-hospedadas. Um terceiro fator com influência semelhante ao fator quantidade de MVs foi o fator memória alocada a cada MV com 18,44%. Esta alta influência se deu por resultado superior do virtualizador Xen, quando as MVs do Xen tinha 1024MB de memoria alocada.

Quanto a influência nas interações dos fatores, a interação entre os fatores virtualizador e memória da MV apresentaram uma influência de 10,37%. Quando considerado o fator memória da MV, com um nível de 1024MB e a quantidade de MVs igual a 3, o Xen foi superior a todos os experimentos. Já a interação entre virtualizador e quantidade MVs alocadas apresentaram 6,69% na influência do desempenho do sistema, isto é perceptível devido ao KVM apresentar um desempenho inferior quando considerado o aumento de MVs . Os demais fatores e interações obtiveram resultados próximos de 0%.

Na Figura 4 é possível observar os resultados em que a comparação entre as médias é utilizada.

Comparando experimento por experimento em função do virtualizador, independente do cenário, o Xen se destaca nos resultados. Entretanto, ao considerar o Xen como

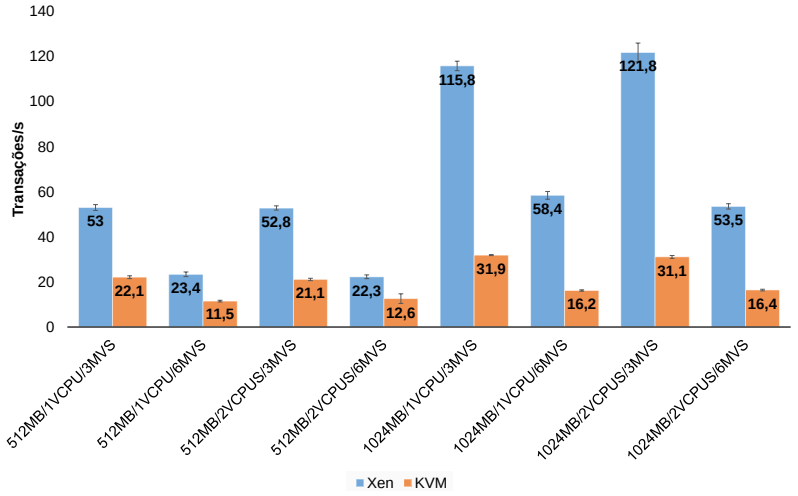

Figura 4. Resultado desempenho dos virtualizadores com o sistema Postmark.

uma boa escolha para cargas de trabalhos que demandam recursos de E/S, é possível observar algumas características. Os experimento em que o número de MVs é igual a 6 apresentam resultado muito inferior, é evidente, pois o principal recurso demandado se trata de memória secundária, em que gargalos são comuns pela sua velocidade de leitura e escrita.

Um resultado que difere dos demais é quando o Xen tem suas MVs com 1024MB de memória alocado, fica evidente o quanto superior este experimento é quando observado. A memória alocada a MV e a quantidade de MV neste cenário fazem toda a diferença.

Ao observar o fator quantidade de MVs, nota-se que quando o virtualizador Xen possui o número de MVs igual a 6, cada uma com 512MB de memória alocada, totalizando 3GB alocados na máquina hospedeira, é bem inferior do que 3 MVs cada qual com 1024MB, que também totaliza 3GB de memória alocado na máquina hospedeira. Esta comparação permite observar com clareza a presença de interferência gerada entre as próprias MVs e mais ainda o gargalo causado nos processos de leitura e escrita na memória secundária.

5.3. RamSpeed

Diferentemente dos demais *benchmarks*, o RAMSpeed apresentou resultados em que a influência no desempenho apresenta está bem dividida, conforme observado na Figura 5.

Quando considerado ambientes em que o principal recurso compartilhado é a memória RAM, o fator que mais influenciou no desempenho do sistema e não diferindo muito dos resultados até então encontrados, foi a quantidade de MVs com aproximadamente 38,71%. Quando aumentado a quantidade de MVs, consequentemente perde-se em MB transferidos por segundo. O fator virtualizador influenciou 13,98%, devido ao virtualizador KVM, que na maioria dos experimentos apresentou resultados superiores ao Xen.

Na interação entre fatores, os fatores virtualizador e memória se mostraram 13,07% influentes, isso devido a alguns experimentos do KVM apresentar resultados que

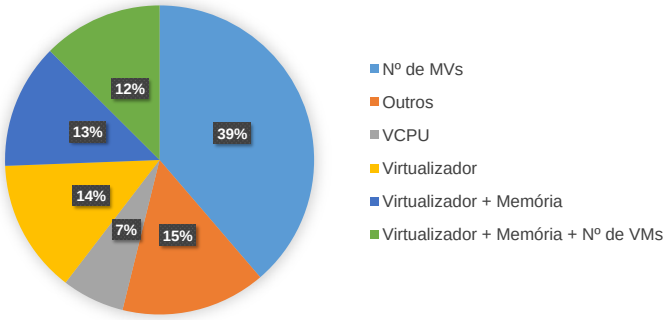

Figura 5. Influência na Memória

quando equiparados a diferença de memória em 512MB e 1024MB, não apresentaram melhoras nos resultados. Ao contrário do Xen, em que a memória aumenta os resultados aumentam proporcionalmente.

Este fato também é representado na interação entre os fatores virtualizador, memória e quantidade de MVs, que representa 12,54% da influência. O fator quantidade de MVs vem acrescentado devido a quantidade de MVs hospedadas não interferir tanto no virtualizador KVM quando a condição de memória alocada é de 1024MB. Em outras palavras, os resultados do KVM com 1024MB de memória alocada para cada MV independem muito pouco de fatores como a quantidade de MVs co-hospedadas, memória alocada a MV. Exceto a quantidade de vCPU, que inclusive tem 7% de representativa na influência.

A Figura 6 apresenta a comparação entre as médias dos experimentos dos resultados obtidos nos experimentos. É possível observar que o KVM se mostrou equivalente ou até mesmo superior na maioria dos experimentos. Quando a carga de trabalho analisada é a memória RAM, percebe-se que o aumento na quantidade de MVs também é um problema, pois a sobrecarga é visível, principalmente para o virtualizador Xen.

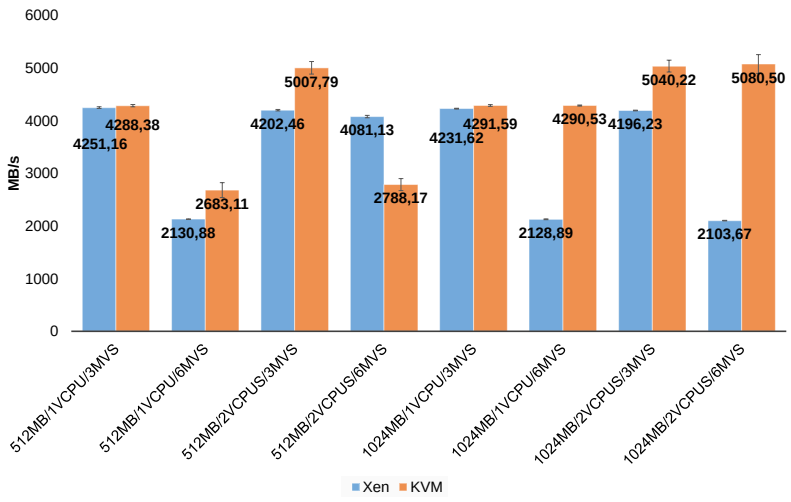

Figura 6. Resultado desempenho dos virtualizadores com o RAMSpeed

A presença de sobrecarga maior no virtualizador Xen é perceptível, uma vez que quando aumentado o número de memória alocada a MV, considerando a vCPU e a quanti-

dade MVs iguais, o mesmo apresenta resultado igual ou até mesmo inferior. A disputa por recursos que não sejam memória RAM na máquina hospedeira é tão grande que mesmo sendo o principal recurso avaliado pelo *benckmark*, a memória RAM não interfere tanto nos resultados. No entanto, uma particularidade do Xen será discutida mais adiante.

5.4. Smallpt

O resultado da influência no SmallPT pode ser visualizado na Figura 7:

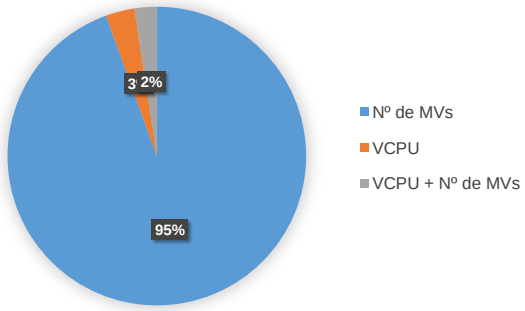

Figura 7. Influência no desempenho da CPU

O fator quantidade de MVs com 94,36% da influência reforça ainda mais a presença de gargalos em MVs co-hospedadas. Entre os demais fatores, a quantidade de vCPU alocada as MVs obteve 3,11% da influência. Apesar de ser uma pequena influência, a principal causa é quando há duas vCPUs alocadas para cada MV, em que na maioria dos casos o desempenho não tem uma mudança significativa em relação ao aumento deste recurso, chegando a ser inferior de quando se tem uma única vCPU alocada as MVs.

A interação entre os fatores vCPU e quantidade de MVs representaram ainda 2,40% da influência no desempenho, que representa algo similar a quantidade de vCPU alocadas as MVs. Isso porque o aumento na quantidade de vCPU nas MVs chegam a influenciar nos resultados de maneira negativa, principalmente no virtualizador KVM.

Sendo a última carga de trabalho analisada, o Smallpt corresponde ao uso de CPU. A Figura 8 apresenta a comparação entre as médias dos experimentos.

Quando a carga de trabalho se caracteriza por inteira em CPU, os virtualizadores se mostraram totalmente equivalentes. Pelos respectivos intervalos de confiança comparados média a média é possível afirmar a igualdade. Neste cenário, o problema que vem sendo estudado e discutido que é a interferência entre máquinas co-hospedadas pode ser melhor observado do que em todos as outras cargas de trabalhos.

Em [Benevenuto et al. 2004], é apresentada a influência em pontos específicos da CPU, pode ser que seja na cache L2 conforme fundamentado por ele. Entretanto, não se pode afirmar, uma vez que o projeto de experimentação deste trabalho considera a CPU como um todo, não há como extrair informações tão específicas. Mesmo assim, percebe-se que independente da forma de escalonamento do virtualizador, ambos apresentam queda no desempenho médio quando aumentado o número de MVs.

Portanto, diferentemente do Apache que utiliza recursos diversos, dentre eles CPU e E/S e os algoritmos de E/S diferem um do outro, o SmallPT firma-se apenas na CPU.

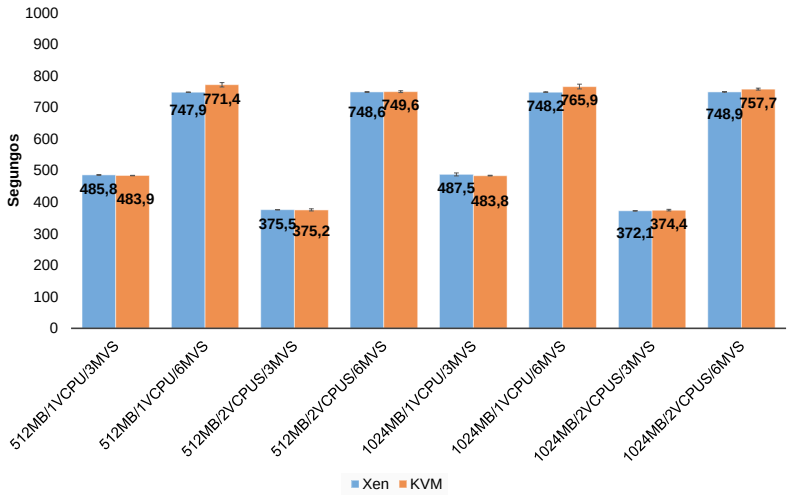

Figura 8. Resultado desempenho dos virtualizadores com o SmallPT.

Quando analisado o algoritmo de escalonamento da CPU da MV, conforme a Tabela 6 percebe-se que ambos os virtualizadores utilizam o mesmo.

Tabela 6. Algoritmos de escalonamento da CPU da Máquina Virtual

Escalonador/Virtualizador	Xen	KVM
MV	CFS	CFS

6. Conclusão

A virtualização é muito utilizada na computação em nuvem. Além disto, seu uso é propriamente benéfico aos provedores de serviço, meio ambiente no que se diz a respeito de computação verde, entre outros. Porém, um conceito que também deve considerado na computação em nuvem é o SLA, em que cliente e provedores devem estar a par das garantias estabelecidas. Mais do que isso, o SLA está ligado a reputação e qualidade do serviço oferecido pelo provedor. Neste contexto, este trabalho tinha como objetivo identificar os gargalos que degradam o desempenho em um ambiente em nuvem e, em função de como as MVs co-hospedadas interferem umas nas outras, tendo como resultado os principais fatores e sua respectiva significância.

Analisando os resultados de forma geral, em função da carga de trabalho imposta pelo sistema Apache, o KVM apresenta ser uma boa escolha mediante ao Xen. Quanto ao número de MVs, percebe-se que de maneira proporcional a quantidade de MVs tem-se o número de requisição atendidas, é viável manter um número elevado de MVs. Entretanto, as condições mínimas estabelecidas no SLA sejam atendidas conforme os resultados considerando o nível de confiança estabelecido. No Postmark o virtualizador Xen prevalece como melhor escolha, desde que características como a quantidade de memória RAM alocada a MV e quantidade de MV sejam consideradas. O KVM em função de cargas de trabalho que tem como base processos de E/S se torna inviável.

No *benckmark* de memória, percebeu-se que o aumento de vCPU exige que a memória RAM tenha mais vazão, independente do número de MVs estabelecidas através do processo de experimentação, quantidade de vCPU sendo igual a 2 e a memória alocada

é igual a 1024MB, o KVM consegue manter um bom desempenho ao contrário do Xen. Enfim, torna-se viável adotar o uso do KVM quando a carga de trabalho imposta tem por características o uso excessivo da memória RAM. Mesmo com um número elevado de MVs, basta que o número de vCPU também seja superior, conseguindo obter melhor desempenho e consequentemente custo benefício.

No que se diz respeito a CPU, o *benckmark* Smallpt mostrou que a quantidade vCPU interfere positivamente nos resultados, diminuindo o tempo de processamento para renderização em ambos os virtualizadores, o que se justifica devido ao algoritmo de escalonamento ser o mesmo. Por outro lado, a memória alocada a MV pouco interfere diante esta carga de trabalho. Enfim, ambos os virtualizadores são boas opções neste quesito.

Ter como base informações, que quando utilizadas de maneira correta podem ser fundamentais a tomada de decisão e que consequentemente podem vim a minimizar prejuízos em ambas as partes, provedor e cliente, é fundamental para um bom prestador de serviço em nuvem. Este trabalho apresentou uma análise de acordo com a carga de trabalho, como se comportou cada um dos virtualizadores, além de observar características que são importantes quando se quer obter o máximo de desempenho possível a um menor custo de infraestrutura, manutenção, etc.

Referências

Arthur Baruchi, E. T. M. (2008). Influência do algoritmo de escalonamento credit scheduler no desempenho de rede. In *Anais do XXVIII Congresso da SBC : WSO - Workshop de Sistemas Operacionais*. SBC.

Benevenuto, F., Ismael Jr, J., and Almeida, J. (2004). Quantitative evaluation of unstructured peer-to-peer architectures. In *Proceedings of the 2004 International Workshop on Hot Topics in Peer-to-Peer Systems*, HOT-P2P '04, pages 56–65, Washington, DC, USA. IEEE Computer Society.

Daniels, J. (2009). Server virtualization architecture and implementation. *Crossroads*, 16(1):8–12.

Head, M., Kochut, A., Schulz, C., and Shaikh, H. (2010). Virtual hypervisor: Enabling fair and economical resource partitioning in cloud environments. In *Network Operations and Management Symposium (NOMS), 2010 IEEE*, pages 104–111.

Jain, R. (1991). *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley Professional Computing. Wiley.

Jiang, W., Zhou, Y., Cui, Y., Feng, W., Chen, Y., Shi, Y., and Wu, Q. (2009). Cfs optimizations to kvm threads on multi-core environment. In *Parallel and Distributed Systems (ICPADS), 2009 15th International Conference on*, pages 348–354.

Khan, A., Othman, M., Madani, S., and Khan, S. (2014). A survey of mobile cloud computing application models. *Communications Surveys Tutorials, IEEE*, 16(1):393–413.

Koh, Y., Knauerhase, R., Brett, P., Bowman, M., Wen, Z., and Pu, C. (2007). An analysis of performance interference effects in virtual environments. In *Performance Analysis*

of Systems Software, 2007. ISPASS 2007. IEEE International Symposium on, pages 200–209.

Lee, M., Krishnakumar, A. S., Krishnan, P., Singh, N., and Yajnik, S. (2010). Supporting soft real-time tasks in the xen hypervisor. *SIGPLAN Not.*, 45(7):97–108.

Matthews, J. N., Hu, W., Hapuarachchi, M., Deshane, T., Dimatos, D., Hamilton, G., and McCabe, M. (2007). Quantifying the performance isolation properties of virtualization systems. In *Experimental Computer Science on Experimental Computer Science*, ecs'07, pages 5–5, Berkeley, CA, USA. USENIX Association.

Niyato, D. (2011). Optimization-based virtual machine manager for private cloud computing. In *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, pages 99–106.

Santos, R. and Charão, A. S. (2008). Análise comparativa de desempenho do hipervisor xen: Paravirtualização versus virtualização total. *Universidade Federal de Santa Maria. Brasil*.

Somani, G. and Chaudhary, S. (2010). Performance isolation and scheduler behavior. In *Parallel Distributed and Grid Computing (PDGC), 2010 1st International Conference on*, pages 272–277.

Srinivasan, S., Raja, K., and Muthuselvan, S. (2012). Futuristic assimilation of cloud computing platforms and its services. In *Emerging Trends in Electrical Engineering and Energy Management (ICETEEEM), 2012 International Conference on*, pages 178–180.

VMware (2014). What is virtualization? https://www.vmware.com/br/virtualization/virtualization-basics/what-is-virtualization. Accessed: 2014-01-30.

White, J. and Pilbeam, A. (2010). A survey of virtualization technologies with performance testing. *CoRR*, abs/1010.3233.

Zaghloul, S. (2013). The mutual effect of virtualization and parallelism in a cloud environment. In *AFRICON, 2013*, pages 1–5.

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.