Influência do Algoritmo de Escalonamento *Credit Scheduler* no Desempenho de Rede

Artur Baruchi, Edson Toshimi Midorikawa

Laboratório de Arquitetura e Computação de Alto Desempenho
Departamento de Engenharia de Computação e Sistemas Digitais – Escola Politécnica
da Universidade de São Paulo – São Paulo – Brasil

{artur.baruchi,edson.midorikawa}@poli.usp.br

Abstract. *This paper will analyze the influence of the scheduler algorithms in network throughput of a virtualized environment. For research tools, Xen will be used as Virtual Machine Monitor and the Credit Scheduler as scheduler algorithm. The experimental results show that under specific configurations changes in some scheduler's parameters can influence in high degree the observed network throughput.*

Resumo. *Neste artigo, será analisada a influência dos algoritmos de escalonamento na vazão de rede em um ambiente virtualizado. Como ferramentas de estudo serão utilizadas o Xen como monitor de máquinas Virtuais e o algoritmo de escalonamento Credit Scheduler. Após a análise dos resultados experimentais, pode-se verificar que, em determinadas configurações, algumas alterações nos parâmetros do escalonador podem influenciar de forma bastante acentuada a taxa observada de vazão da rede.*

1. Introdução

Algoritmos de escalonamento são utilizados em Sistemas Operacionais (SO) para gerenciar a utilização do processador. Em um SO, diversos processos podem estar prontos para utilizar o processador e quando isto ocorre o algoritmo de escalonamento deverá decidir, de acordo com uma determinada política, qual processo será executado em seguida [Tanenbaum e Woodhull, 2000]. As políticas de escalonamento serão determinadas de acordo com as características de cada SO. Em ambientes onde o tempo de utilização de CPU deve ser compartilhado, o escalonador deve tratar os processos de tal forma que evite, por exemplo, a ocorrência de *starvation*. Já em um ambiente de tempo real, o escalonador deve dar preferência para processos críticos em detrimento da execução de outros processos [Machado e Maia, 2002].

Em um ambiente virtualizado, as Máquinas Virtuais (MVs) são tratadas de forma parecida a um processo tradicional. Desta maneira pode-se observar que o desempenho de uma MV pode ser afetado de acordo com as decisões de escalonamento tomadas, por exemplo: seja uma MV que, de acordo com as políticas de escalonamento utiliza 50% do processador e, se nesta MV, existe um processo que utiliza 50% do tempo disponibilizado, então este processo irá usufruir, efetivamente, de apenas 25% do processador.

As aplicações que fazem uso de forma excessiva de I/O, isto é, aplicações do tipo *I/O Bound*, são bastante penalizadas ao serem virtualizadas, pois exigem um maior número de troca de contexto entre a Máquina Virtual e o seu Hospedeiro [Barham,

Dragovic *et al*., 2003]. Estas trocas de contexto tendem a ter um custo computacional alto, pois dependendo do tipo de suporte de hardware, pode ser extremamente lento e somam-se a isto as camadas de Software inerentes à Virtualização.

O principal objetivo deste trabalho é analisar e quantificar de que forma o algoritmo de escalonamento pode influenciar uma operação de I/O em ambientes virtualizados, mais especificamente no desempenho de rede. Para esta análise o Monitor de Máquinas Virtuais (MMV) escolhido foi o Xen [Xen, 2006], pois o acesso ao código fonte é livre e devido a sua flexibilidade de configuração. O algoritmo de escalonamento escolhido foi o *Credit Scheduler*, pois este é o algoritmo padrão do Xen.

Este artigo está dividido da seguinte forma: na seção 2 serão analisados alguns trabalhos já realizados a respeito dos algoritmos de escalonamento em MVs e estudos de desempenho de rede e na seção 3 serão apresentados os algoritmos de escalonamento disponíveis no Xen. A seção 4 demonstrará os resultados experimentais obtidos e as ferramentas utilizadas para a obtenção destes, bem como a descrição e configurações dos testes. Por fim, na seção 5, serão apresentadas as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

O problema de desempenho em operações de I/O nas Máquinas Virtuais torna este tema bastante atrativo para pesquisas implicando na confecção de diversos trabalhos, como no caso de [Urschei, Peregrine *et. al*., 2007]. Este artigo analisa o *overhead* da virtualização e observa que ao alterar alguns parâmetros de rede, pode-se obter um desempenho de rede muito próximo ao de uma máquina com Linux Nativo. Outro trabalho, também nesta linha de análise de *overhead*, pode ser encontrado em [Cherkasova, Gardner, 2005]. Este discute a carga no processador em operações de I/O no Xen, quantificando o *overhead* no Domínio[1] 0 (que é referenciado como *Device Driver Domain*), no qual os autores demonstram que o aumento de processamento pode variar em função da quantidade de *bytes* transferidos. Um grande número de conexões HTTP realizando requisições de arquivos pequenos não gera uma grande carga de processamento, pois são contrabalanceados pelo grande numero de interrupções correspondentes ao processamento das requisições HTTP.

Ao contrário de trabalhos relacionados a operações de I/O, existem poucos trabalhos referentes a algoritmos de escalonamento para MVs e seus efeitos no desempenho. Em um trabalho muito interessante, realizado por [Cherkasova, Gupta *et. al*., 2007], foi feita uma comparação dos dois escalonadores disponíveis no MMV Xen, o *Credit Scheduler* e *Simple Earliest Deadline First* (SEDF) com o algoritmo *Borrow Virtual Time* (BVT) [Duda, Cheriton, 1999], que não faz mais parte do projeto Xen. Entre as comparações realizadas neste artigo, os autores avaliam o comportamento dos algoritmos em máquinas SMP utilizando uma ferramenta desenvolvida por eles mesmos, chamada de ALERT. Dentre todos os algoritmos o *Credit Scheduler* utilizou de forma mais eficiente os processadores disponíveis. A métrica utilizada pelo ALERT é baseada na disponibilidade de recursos e na demanda de uma determinada MV. De acordo com estas informações a alocação da CPU será feita, de tal modo que cada MV consiga obter os recursos dos quais precisa.

Por fim, outro trabalho relevante pode ser encontrado em [Cherkasova, Gupta, *et. al.*, 2005] no qual é apresentado o Xenmon. O Xenmon é uma ferramenta criada para

[1] Neste artigo, Domínio e Máquina Virtual possuem o mesmo significado.

verificar o comportamento de alocação de processador no Xen, mais especificamente da alocação para as MV criadas. Desta forma é possível verificar como foi aproveitado o tempo disponibilizado a uma determinada MV, entre outras métricas que serão apresentadas com mais detalhes nas próximas seções.

Os trabalhos mostrados anteriormente abordam diversos aspectos de I/O em Máquinas Virtuais e comportamento dos algoritmos de *scheduler*. Neste trabalho será analisada a vazão de rede, independente da aplicação. Nos trabalhos discutidos acima, a vazão de rede é obtida através de servidores *Web* e por este motivo os testes experimentais podem variar bastante para qualquer outro tipo de aplicação que utilize a infra-estrutura de rede disponibilizada pelo Xen.

3. Escalonamento no XEN

Atualmente é possível escolher um entre dois algoritmos de escalonamento no XEN, o *SEDF* [Leslie, Macauley, *et. al.*, 1996] e o *Credit Scheduler* [Credit, 2006]. Os algoritmos de escalonamento do XEN podem ser classificados de acordo com duas categorias [Cherkasova, Gupta *et. al*., 2007]:

- *Work-conserving* (WC-mode) ou *Non Work-conserving* (NWC-mode): No WC-mode o processador fica ocioso se e somente se não houver nenhum processo para ser executado. Na prática isto significa que se dois processos têm o mesmo peso e um deles está bloqueado, o outro processo poderá utilizar todo o processador. Entretanto no NWC-mode, cada processo utilizará somente a parte que lhe foi concedida e nada mais, e desta forma, o processador pode ficar ocioso.
- *Preemptivo* ou *Não-Preemptivo:* Um algoritmo de escalonamento preemptivo verifica a prioridade de todos os processos que entram em estado de pronto para utilizar o processador. Caso a prioridade deste novo processo seja maior que o processo em execução este será substituído imediatamente pelo novo processo com maior prioridade. Em contrapartida um algoritmo não-preemptivo, permitirá que os processos em execução terminem de utilizar o seu *time slice* para depois verificar as prioridades dos processos em estado de pronto e escolher qual deles deverá utilizar o processador. Portanto, um algoritmo de escalonamento preemptivo será re-executado sempre que um novo processo ficar pronto para utilizar o processador.

Um algoritmo preemptivo é a melhor opção para aplicações que utilizam operações de I/O de forma intensiva, visto que estas aplicações costumam ficar bloqueadas a espera de I/O e podem ser prejudicadas ao concorrer com aplicações *CPU Bound*.

Os dois algoritmos disponíveis no XEN trabalham em WC-mode e NWC-mode, porém somente o SEDF é preemptivo e principalmente por este motivo apresenta melhor desempenho em operações de I/O que o algoritmo *Credit Scheduler* [Cherkasova, Gupta *et. al*., 2007].

O escalonador SEDF (*Simple Earliest Deadline First*) utiliza uma tupla (s_i, p_i, x_i) no qual um determinado domínio$_i$ (Dom_i) identifica as suas necessidades. Juntos o *slice* s_i e o período p_i representam a quantidade de tempo de execução que o Dom_i irá receber, isto é, o Dom_i receberá no mínimo s_i unidades de tempo em cada período de tamanho p_i. O valor de x_i indica se o Dom_i poderá ou não utilizar por mais tempo o processador do que lhe foi concedido (WC-mode ou NWC-mode) [Leslie, Macauley, *et. al.*, 1996].

No caso do algoritmo *Credit Scheduler*, para cada domínio é atribuído um peso (*weight*) e o *Cap*. O valor de *Cap* identifica se o domínio poderá ou não exceder o tempo de processamento que lhe foi concedido (WC-mode ou NWC-mode). Se o valor de *Cap* for zero, então o domínio não terá limite superior para utilizar processamento a mais. Um valor diferente de zero identifica a quantidade (em porcentagem) de processamento a mais que o domínio poderá exceder. Já o peso de um domínio está ligado à quantidade de vezes que este irá utilizar o processador. Em um ambiente onde há concorrência, um domínio de peso 512 irá utilizar duas vezes mais o processador que um domínio de peso 256 [Credit, 2006].

Cada um dos processadores é responsável por gerenciar uma fila de VCPU's executáveis. A prioridade de uma VCPU tem dois valores possíveis: alta (*over*) ou baixa (*under*), que identifica se a VCPU excedeu ou não o tempo disponibilizado a ela para utilizar o processador. Quando uma VCPU é inserida na *run queue* esta é colocada atrás de todas as outras VCPU's de mesma prioridade.

Conforme a VCPU é executada, ela consome créditos (razão pela qual este algoritmo chama-se *Credit Scheduler*). O algoritmo refaz o cálculo de quantos créditos cada domínio gastou ou ganhou depois de cada execução, créditos negativos implicam em uma prioridade baixa (*under*), até que a VCPU consuma seus créditos, esta é considerada de prioridade alta (*over*).

O padrão do *Credit Scheduler* é utilizar 30ms para cada período de tempo (*time slice*). Pelo fato deste escalonador não ser preemptivo, cada domínio receberá 30ms de processamento antes de ser substituído por outro domínio. Deste modo, a cada 30ms as prioridades, ou créditos, de cada domínio são re-calculados. Uma característica muito interessante deste escalonador é o balanceamento de carga entre os processadores de um hospedeiro SMP. Nele se um determinado processador não encontrar um domínio executável em sua *run queue*, ele irá buscar algum domínio executável em *run queues* de outros processadores, garantindo uma melhor utilização do processador e que cada domínio receba o tempo de processamento que lhe foi concedido.

4. Avaliação de Desempenho de Rede no Xen

A estrutura de rede concebida no Xen foi construída de forma a não causar um *overhead* muito grande nas operações de comunicação como se pode observar em [Urschei, Peregrine *et. al*., 2007].

Assim que o Xen inicia, é criado um domínio (domínio 0, ou Dom0) que possui acesso direto aos *drivers* de dispositivos [Fraser, Neugebauer, *et. al.*, 2004], além de possuir privilégios de gerenciamento sobre os outros domínios (como criação, remoção, alteração de configurações, entre outros). Este domínio realizará o processamento referente aos *drivers* de dispositivos, isto é, ele é o responsável pela comunicação com o *hardware* real. Deste modo, o Dom0 pode ser definido como um intermediário entre a máquina real e a MV. – Figura 1.

Desta maneira pode-se dizer que o Dom0 realiza as operações de I/O propriamente ditas – Figura 1b. Uma vez que o Dom0 processa a informação, esta é transmitida para a respectiva Máquina Virtual utilizando os anéis de I/O assíncrono [Barham, Dragovic *et al*., 2003]. Com isto é possível observar dois pontos importantes em relação ao modelo de I/O implementado atualmente no Xen:

- O processamento dos *Drivers* dos dispositivos físicos é feito pelo Dom0;

- Em uma operação de I/O, as MVs têm o trabalho de processar os anéis de I/O assíncrono;

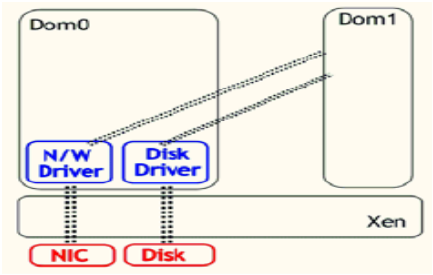


Figura 1. Atual Modelo de I/O [Cherkasova, Gupta *et. al*., 2007].

Com estas observações, surge uma questão: qual componente do ambiente virtualizado poderia ser privilegiado pelo escalonador. Existem duas linhas de raciocínio em relação ao estudo da influência do escalonador no desempenho de rede. A primeira, se o peso no Dom0 for maior, implicará que ele terá mais prioridade para utilizar o processador e por sua vez, conseguirá processar por mais tempo os *drivers* de dispositivos. Já a segunda analisa um aumento do peso de uma MV, assim esta terá mais prioridade para processar os anéis de I/O assíncronos. Os testes realizados neste trabalho têm como objetivo responder estas questões levantadas. Outro aspecto importante a ser verificado neste trabalho é determinar quando e de que forma é vantajoso alterar as configurações do escalonador para obter melhor desempenho.

4.1. Xenmon

A ferramenta Xenmon, apresentada em [Cherkasova, Gupta, *et. al.*, 2005], tem como principal objetivo registrar o uso do processador pelas Máquinas Virtuais no Xen e como elas estão se comportando. A arquitetura do Xenmon – Figura 2 – é composta por três camadas:

- *Xentrace*: É o gerador de eventos que já vem no próprio Xen. Com este componente é possível colher eventos de determinados pontos no MMV.
- *Xenbaked*: Este componente é responsável por transformar os dados coletados pelo Xentrace em informações úteis. O Xenbaked é um processo executado no espaço de usuário.
- *Xenmon*: É o *front-end* usado para apresentar e registrar as informações.

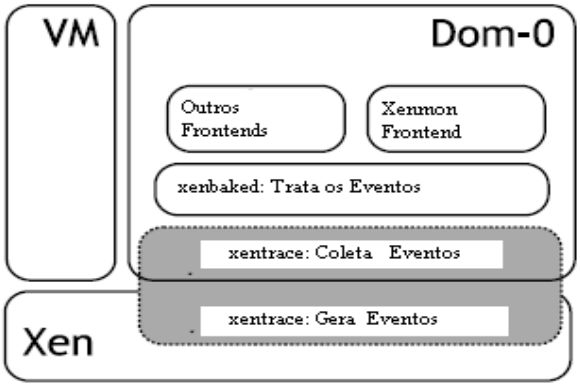


Figura 2. Arquitetura do Xenmon [Cherkasova, Gupta, *et. al.*, 2005]

Dentre as diversas informações coletadas pelo Xenmon, existem quatro que auxiliam a entender melhor o comportamento dos Domínios:

- *Utilização de Processador*: Informa a porcentagem de tempo (durante 1 segundo) que um determinado domínio efetivamente utilizou o processador;
- *Tempo de Bloqueio*: Porcentagem de tempo que um domínio gastou enquanto esperava por algum evento de I/O (*sleeping*);
- *Tempo de Espera*: Porcentagem de tempo que um domínio gastou esperando para utilizar o processador na *run queue.*
- *Execuções/segundo*: Número de vezes que um Domínio utilizou o processador. Esta métrica é inversamente proporcional ao Tempo de Espera, quanto maior o tempo de Espera menor é o número de vezes que o Domínio utiliza o processador.

Com estas informações é possível observar como as MVs estão se comportando ante a alteração dos parâmetros do escalonador.

4.2 Configurações do Ambiente

Nos testes realizados foram utilizadas duas máquinas conectadas por um cabo *cross* (para evitar interferência de outros equipamentos de rede). Apesar de ambas as placas de rede serem *Gigabit Ethernet*, a velocidade configurada para a realização dos testes foi de 100Mbps, pois ao utilizar Virtualização Completa ainda não é possível utilizar a velocidade de 1Gbps o que poderia impedir futuras comparações entre a Paravirtualização e a Virtualização Completa. A máquina hospedeira das Máquinas Virtuais (Máquina01) possui um processador dual *core* AMD Athlon 64 X2 de 2GHz e 4GB de memória. A segunda Máquina (Máquina02) contém um processador dual *core* AMD Athlon 64 X2 de 2.2GHz e 1GB de memória.

Apesar das duas máquinas possuírem processadores de dois *cores*, um deles foi desabilitado para que em um trabalho futuro seja possível compararmos os resultados de um ambiente SMP com um ambiente monoprocessado.

Na máquina Hospedeira foi criada uma máquina paravirtualizada (Máquina Virtual) [Barham, Dragovic *et al*., 2003] com um processador virtual (VCPU) e com 512MB de memória. Em todas as máquinas o sistema operacional instalado é o Linux Fedora Core 8 e a versão do Xen utilizada foi xen-3.1.2.

4.3 Análises da Influência do Escalonador nas Taxas de Vazão

Durante os testes experimentais, diversos parâmetros foram alterados para verificar em qual situação é possível melhorar a vazão da rede. Os parâmetros de rede alterados nos experimentos foram os mesmos parâmetros modificados no trabalho de [Urschei, Peregrine *et. al*., 2007]. O *buffer* de sistema (ou *socket size*) foi ajustado com o valor de 1048575, 65536, 57344, 32768, 8192, 4096 e 2048 *bytes*. Para cada valor de *buffer* de sistema, o *buffer* de aplicação (ou tamanho da mensagem transmitida) foi alterado para 512, 1024, 2048, 4096, 8192, 32768, 65536 e 1048576 *bytes* e por fim foi alterado o valor de MTU, primeiramente para 1500 *bytes* e depois para 500 *bytes*.

Os experimentos foram realizados entre uma MV na Máquina01, na qual foi executada a parte cliente do Netperf [Netperf, 2006] e a Máquina02, que executava a parte servidora. No primeiro experimento, com o objetivo de estabelecer um ponto de referência, o Dominio0 foi ajustado com o peso igual ao da MV (configuração padrão).

Nos experimentos posteriores variou-se o peso[2] do Dominio0 em relação ao da MV a partir de 25%, 50%, 100%, 125%, 150% e até 200% maior. Neste trabalho o desempenho será definido como a melhora ou piora da vazão em relação à configuração padrão.

4.4 Taxa de Vazão da Rede e Variação do Peso do Dom0

Nos gráficos abaixo (Figura 3), são apresentadas as taxas de vazão na configuração padrão e as eficiências de outras configurações de teste ao se alterar o peso do Dom0. Do gráfico da Figura 3.a, pode-se observar que, de uma forma geral, a vazão da rede permaneceu estável próximo do valor máximo para a maioria dos parâmetros de rede. Nota-se especialmente para o caso de *buffer* de sistema de 2048 bytes, que a vazão da rede ficou bem abaixo, apresentando valores próximos de 55Mbps. Para *buffer* de sistema de 4096 bytes, é possível verificar também um menor desempenho para certos tamanhos de *buffer* da aplicação.

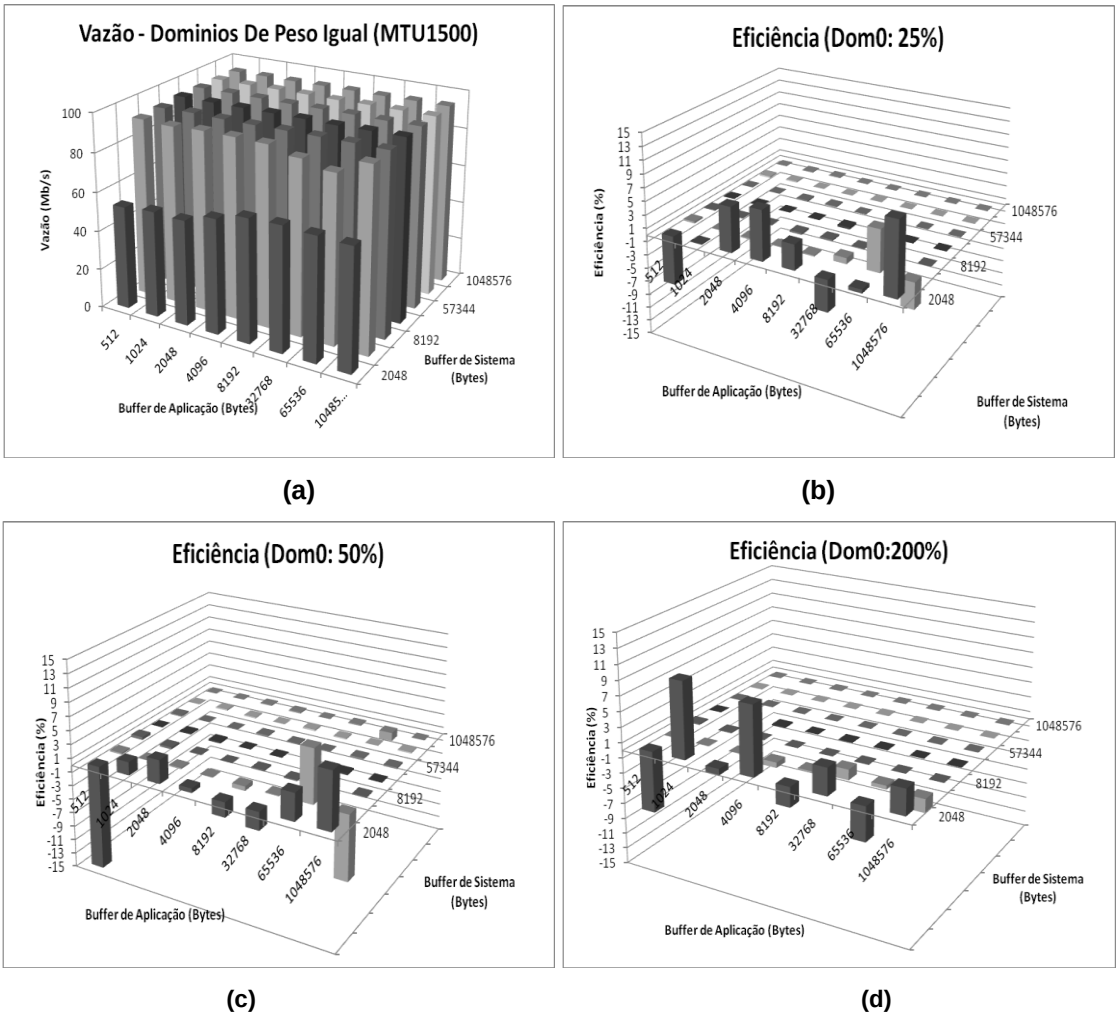


Figura 3. Comparação da Taxa de Vazão entre configuração padrão e Dom0 com diferentes Pesos. (a) Vazão com configuração Padrão. (b) Eficiência do Dom0 a 25% em relação a configuração padrão. (c) Eficiência do Dom0 a 50% em relação a configuração padrão. (d) Eficiência do Dom0 a 200% em relação a configuração padrão.

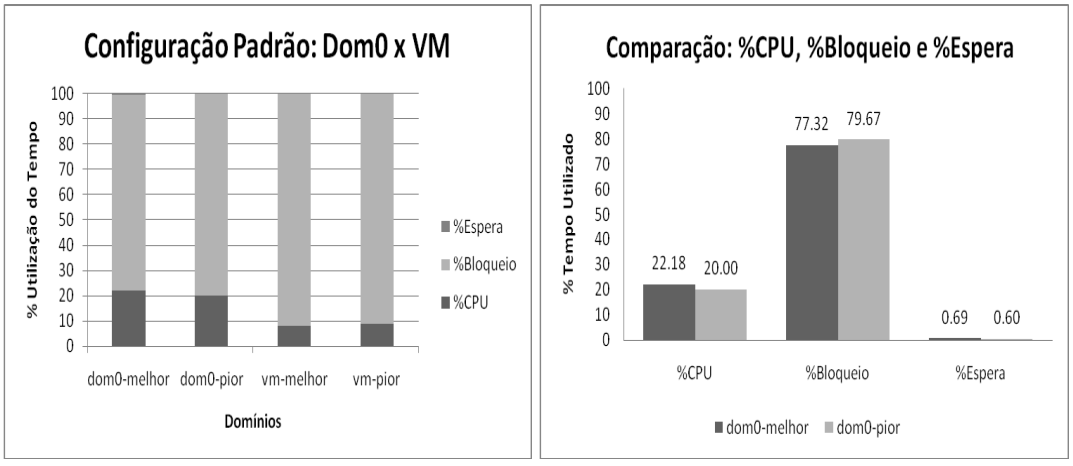
[2] Nos experimentos, foi alterado somente o peso (*weight*) dos domínios, o parâmetro *Cap* não sofreu alterações (WC-mode).

Variando-se os pesos do Dom0, verificou-se que houve variação significativa na vazão observada da rede apenas para *buffers* de sistema de 2048 e 4096 bytes, conforme pode ser observado nos gráficos das Figuras 3.b-d. Estas variações na vazão de rede abrangeram um intervalo de -15,8% (para a configuração peso 50% maior para Domínio 0, *buffer* de sistema de 2048 bytes e *buffer* de aplicação de 512 bytes) até +10,6% (para a configuração peso 25% maior para Domínio0, *buffer* de sistema de 2048 bytes e *buffer* de aplicação de 1MB). Estes resultados mostram que uma modificação adequada do peso no algoritmo *Credit Scheduler* pode levar a um aumento no desempenho de rede do ambiente virtualizado.

4.5 Análises com Xenmon

Na Figura 3a nota-se que com o *buffer* de sistema ajustado para 2048 *bytes* foi a situação na qual se obteve a pior taxa de vazão. Uma análise com o Xenmon desta configuração em comparação com uma configuração na qual a taxa de vazão é considerada boa (por exemplo, *buffers* de aplicação e sistema ajustados a 1MB) verificou que a causa do pior desempenho foi o período de bloqueio do Dom0 e o tempo de CPU usado – Figura 4.

A figura 4.a mostra o perfil de execução para o melhor e pior caso estudado. Durante um período de *time slice*, o Dom0 ficou cerca de 20% em execução (%CPU), 79% bloqueado aguardando o término das operações de I/O (%Bloqueio) e menos de 1% na fila (%Espera). Uma comparação entre ambos os casos mostrou – Figura 4.b – que a configuração com pior desempenho ficou bloqueado por mais tempo (+2,35%) aguardando o término das operações de I/O. A causa disto é o maior número de requisições por causa do menor tamanho de *buffer* de sistema. Isto acarretou em um menor tempo ocupando o processador (-2,2%).



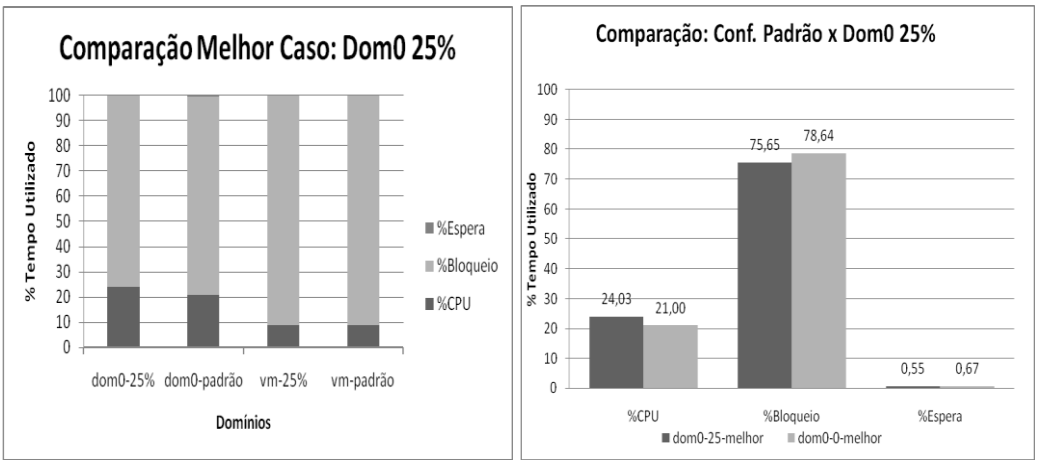
(a) (b)

Figura 4. Comparação de utilização do tempo entre o pior e melhor caso na configuração padrão. (a) Utilização do tempo pelo Dom0 e MV. (b) Comparação de utilização do processador, período de bloqueio e de espera no Pior e Melhor caso.

4.6 Análises da Variação dos Pesos

A configuração com peso do Dom0 25% maior que a MV foi a que apresentou o melhor resultado com uma melhora de 10,6% na vazão da rede para *buffer* de aplicação igual a 1MB e *buffer* de sistema igual a 2048 *bytes*.

A razão pela qual ocorreu esta melhora é o melhor aproveitamento do processador pelo Dom0 e soma-se a este a maior freqüência que o Dom0 é alocado para utilizá-lo – Figura 5. Verificou-se uma diminuição do período de bloqueio (-3%) e aumento na utilização do processador (+2,03%). De acordo com a análise do Xenmon, o Dom0 com peso 25% maior foi escalonado para utilização do processador 800 vezes a mais que o observado na configuração padrão.

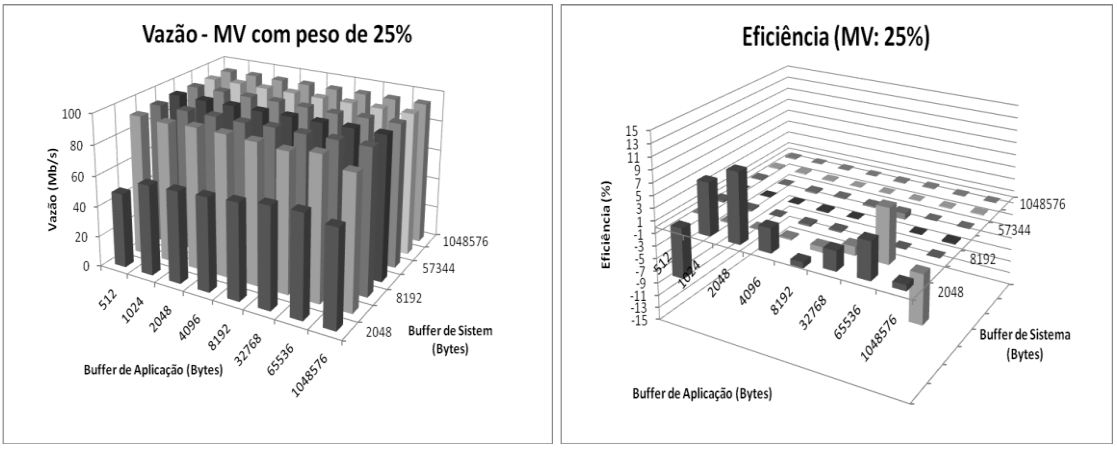


(a) (b)

Figura 5. Comparação entre configuração padrão e Dom0 com peso 25% maior. (a) Comparação da utilização do tempo. (b) Comparação de utilização do processador, período de bloqueio e de espera no melhor caso.

4.6.1 Máquina Virtual com peso maior

Nos próximos testes, foi alterado o peso da MV para 25%, 50% e 100% maior que o Dom0. A configuração em que se observou a melhor taxa de vazão em relação à configuração padrão foi a com MV com peso 25% maior – Figura 6. Em determinada configuração a melhora atingiu 10% (*buffer* de aplicação igual a 2048 *bytes* e *buffer* de sistema igual a 2048 *bytes*) e a pior taxa identificada foi de -7% (*buffer* de aplicação igual a 1MB e *buffer* de sistema igual a 2048 KB).

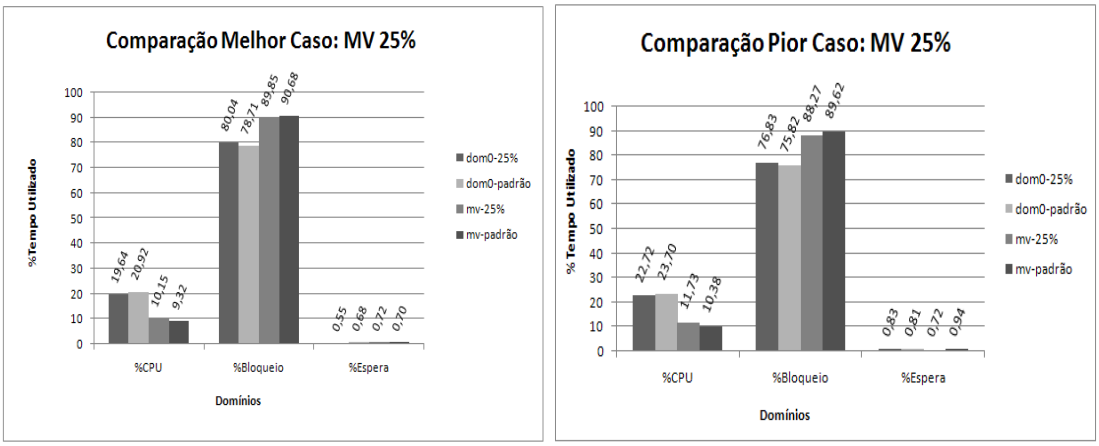


(a) (b)

Figura 6. Análise das Taxas de Vazão com MV com Peso 25% maior. (a) Taxas de Vazão. (b) Eficiência em relação à Configuração Padrão.

Apesar do maior peso da MV, a razão da melhora na taxa de vazão se deve também à melhor eficiência do Dom0. No melhor caso, o Dom0 foi escalonado no processador 1500 vezes a mais do que a configuração padrão. Se for observada apenas a MV, a única melhora foi no tempo de utilização do processador, mesmo assim, não chegou de 1%.

O mesmo ocorre ao verificar o comportamento dos domínios no pior caso. A MV teve pouca influência[3] na baixa taxa de vazão, ao contrário do que ocorre ao observar o Dom0. Com a MV de maior peso, o Dom0 aproveita com pouca eficiência o processador e também o utiliza com menor freqüência em comparação com a configuração padrão – Figura 7.



(a) **(b)**

Figura 7. Comparação entre configuração padrão e a MV com peso de 25% maior. (a) Melhora de 10% em relação à configuração padrão. (b) Piora de -7% em relação à configuração padrão.

Durante todos os testes, foi identificado que a pior taxa de vazão ocorria quando o *buffer* de aplicação estava ajustado para 512 *bytes* e *buffer* de sistema para 2048 *bytes*. Porém, mesmo com a alteração do peso dos domínios, tanto do Dom0 como da MV, em nenhum momento houve melhora, pelo contrário, a taxa de vazão somente piorou. Ao verificar esta mesma configuração com o Linux nativo, o mesmo comportamento foi observado.

4.8 MTU de 500 bytes[4]

Quando observados os números da vazão de rede com MTU igual a 500 *bytes* é razoável se esperar que ocorra alguma queda na vazão quando o Dom0 possui peso maior. Com um MTU menor, a quantidade de interrupções geradas para tratamento aumenta. Porém como a MV tem prioridade menor para utilizar o processador, ela acaba sendo prejudicada de forma mais acentuada. Ao ajustar o Dom0 com peso 50% maior que a MV houve uma piora média de -0,22% no desempenho em relação ao MTU igual a 1500 *bytes*.

Quando o peso da MV foi configurado para um peso maior, utilizando MTU de 500 *bytes*, praticamente não foram observadas quedas na taxa de vazão. Isso se deve ao

[3] Em alguns experimentos, os resultados do Xenmon referentes à MV apresentaram valores com discrepâncias muito altas. Após contato com os autores da ferramenta sobre a causa destes resultados, estes sugeriram que fossem ignorados.

[4] Por motivo de espaço, estes resultados não serão acompanhados de gráficos com os testes realizados.

fato da MV aproveitar melhor o tempo que lhe é concedido para utilização do processador. O comportamento foi exatamente o contrário do que ocorreu quando o Dom0 estava com peso maior (ganho médio de +0,21%). Portanto, a combinação entre o maior número de interrupções e maior peso para a MV resultou em melhor aproveitamento do processador por parte da MV. A maior quantidade de interrupções pode ser comprovada pelo menor tempo de bloqueio da MV.

5. Conclusão e Trabalhos Futuros

Os experimentos mostraram que, em algumas situações específicas, o escalonador possui um papel fundamental no desempenho observado de um ambiente virtualizado. Ao ser modificado com critério pode haver melhora de desempenho, ou pode influnciar de forma bastante negativa se alterado de forma errônea, principalmente no que diz respeito a operações de I/O, as quais são extremamente sensíveis em um ambiente virtualizado. Desta forma, a principal contribuição deste trabalho foi analisar a influência de uma alteração na prioridade de execução dos domínios do Xen, e que esta mudança de prioridade pode causar um aumento no desempenho de rede.

Os resultados obtidos permitem concluir que é bastante encorajador a alteração dos parâmetros do escalonador em conjunto com alguns outros parâmetros de rede, principalmente quando se trata de aplicações que exigem um desempenho bastante satisfatório no quesito de rede. Como também foi demonstrado em [Cherkasova, Gupta *et. al*., 2007], pode-se obter uma melhora considerável alterando-se, inclusive, o algoritmo de escalonamento.

Na maior parte das situações analisadas, a taxa da vazão de rede manteve-se próxima da configuração padrão, pois em nenhum experimento a média da eficiência passou de 1%. Porém, em casos pontuais tanto a melhora como a piora foi substancial. Com MTU de 1500 *Bytes*, a melhor taxa de vazão foi obtida quando o Dom0 aproveitou melhor o tempo de utilização do processador, ou obteve maior número de oportunidades para execução. Em operações de I/O, portanto, é melhor que o Dom0 possua maior prioridade de execução, pois este é o responsável pelo processamento dos *drivers*. Porém com MTU menor, a utilização do processador intensificou-se e, por este motivo, ao aumentar a prioridade da MV obteve-se melhor desempenho na taxa de vazão da rede.

Como trabalhos futuros, pode-se citar inicialmente a análise da influência de outros algoritmos de escalonamento do Xen no desempenho de rede, principalmente devido à possibilidade de analisar a influência da preempção dos processadores. Outro possível trabalho seria a verificação de como o escalonador pode modificar o comportamento de aplicações paralelas em um ambiente de cluster virtualizado usando, por exemplo, primitivas de comunicação MPI.

Além de máquinas paravirtualizadas, se satisfeitas determinadas condições, é possível utilizar máquinas completamente virtualizadas no Xen, porém, estas apresentaram um desempenho geral bastante inferior se comparada a uma máquina paravirtualizada [Barham, Dragovic *et al*., 2003]. Atualmente existem poucas pesquisas que tratam deste tipo de virtualização no Xen, portanto um trabalho bastante interessante seria a análise da influência dos escalonadores e a variação de seus parâmetros em máquinas completamente virtualizadas, bem como a verificação da influência de máquinas SMP ou multi-core com este tipo de virtualização.

6. Referências

Barham, P., B. Dragovic, et al. (2003). "Xen and the Art of Virtualization". 19th ACM Symposium on Operating Systems Principles (SOSP 2003). Bolton Landing, NY, USA: ACM Press. p. 164-177.

Cherkasova, L., Gardner, R. (2005). "Measuring CPU Overhead for I/O Processing in the Xen Virtual Machine Monitor". USENIX Annual Technical Conference. Anaheim, CA. p. 24-24.

Cherkasova, L., Gupta, D., et al (2007). "Comparison of the three CPU schedulers in Xen". ACM SIGMETRICS Performance Evaluation Review. Vol. 35, No.2, p. 42-51.

Cherkasova, L., Gupta, D., et al (2005). "Xenmon: QoS monitoring and performance profiling tool". Technical Report HPL-2005-187, HP Labs.

Credit (2006). Disponível em http://wiki.xensource.com/xenwiki/CreditScheduler. Acessado em 04/03/2008.

Fraser, K., Neugebauer, R., et al (2004). "Reconstructing I/O". Technical report. University of Cambridge, Computer Laboratory.

K. J. Duda e D. R. Cheriton (1999). "Borrowed-virtual-time (BVT) scheduling: supporting latency-sensitive threads in a general-purpose scheduler". Proceedings of the 17th ACM SOSP.

Leslie, I. M., Mcauley, D., et al (1996). "The Design and Implementation of an Operating System to Support Distributed Multimedia Applications". IEEE Journal of Selected Areas in Communications. Vol. 14, No.7, p. 1280-1297.

Machado, B. F. e Maia, P. L. (2002), Arquitetura de Sistemas Operacionais, LTC, 3ª edição.

Netperf (2006). Disponível em www.netperf.org. Acessado em 04/03/2008.

Tanenbaum, S. A. e Woodhull, S. Albert (2000), Sistemas Operacionais: Projeto e Implementação, Bookman, 2ª edição.

Urschei, F., Pelegrini, E. Jose, et al (2007). "Analise Multiparamétrica do Overhead de Rede em Máquinas Virtuais". IV Workshop de Sistemas Operacionais. Rio de Janeiro, RJ. p. 816-827.

Xen (2006). "Xen Source". Disponível em http://www.xensource.com/. Acessado em 02/02/2008.