PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 39/08

Gerenciamento de Recursos no Monitor de Máquinas Virtuais Xen

Valéria Quadros dos Reis
Renato Fontoura de Gusmão Cerqueira

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900
RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 39/08 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena September, 2008

Gerenciamento de Recursos no Monitor de Máquinas Virtuais Xen[1]

Valéria Quadros dos Reis e Renato Fontoura de Gusmão Cerqueira

{vreis, rcerq}@inf.puc-rio.br

Abstract. With the popularization of grid and utility computing systems, virtual machines have reappeared as a feasible option for multiplexing resources among the different services hosted in a same machine. Virtual machines are able to ensure security and performance isolation for the different active domains of a system, in addition to increasing the system resource utilization. This paper presents, in a detailed manner, the architecture of a virtual machine monitor which has gained a great relevance in the academic area, the Xen. The paper sections describe the virtualization technique that Xen adopts for computer devices. The goal of these sections is to provide a basic knowledge about how Xen's resource management works and is implemented.

Keywords: Resource management, virtualization, scheduling

Resumo. Com a popularização de Sistemas de Computação em Grade e de Utilidade (Utility Computing), Máquinas Virtuais têm ressurgido como uma opção viável para a multiplexação de recursos entre serviços hospedados em uma mesma máquina. Isso porque máquinas virtuais garantem o isolamento de desempenho e a segurança para os diferentes domínios ativos em um sistema, além de aumentar a utilização dos recursos disponíveis. Este trabalho apresenta, de maneira detalhada, a arquitetura de um monitor de máquinas virtuais que têm obtido grande relevância entre a comunidade acadêmica, o Xen. Seções descrevem o método de virtualização adotado pelo Xen para os diversos dispositivos de um computador. O objetivo dessas seções é prover um conhecimento básico de como o gerenciamento de recursos do Xen funciona e é implementado.

Palavras-chave: Gerenciamento de recursos, virtualização, escalonamento

[1]Trabalho patrocinado pelo Ministério de Ciência e Tecnologia da Presidência da República Federativa do Brasil. Processo CNPq número 142035/2006-8.

Responsável por publicações:
Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22451-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3527-1516 Fax: +55 21 3527-1530
E-mail: bib-di@inf.puc-rio.br
Web site: http://bib-di.inf.puc-rio.br/techreports/

1 Introdução

O atual desenvolvimento de computadores, com altas capacidades de processamento e grande disponibilidade de memória e armazenamento, tem levado a uma ressurgimento do interesse pelo uso da tecnologia de máquinas virtuais. Máquinas virtuais possibilitam a divisão de recursos entre diferentes instâncias de Sistemas Operacionais, aumentando assim a utilização dos recursos, facilitando a administração do sistema e ainda garantindo a segurança dos ambientes [Keahey, Doering e Foster 2004].

Em ambientes de Computação em Grade e *Utility Computing*, onde diferentes usuários compartilham o mesmo *hardware* e executam códigos possivelmente maliciosos, as características de máquinas virtuais são bastante desejáveis [Figueiredo, Dinda e Fortes 2003, Krsul et al. 2004, Santhanam et al. 2005, Adabala et al. 2005]. No que diz respeito ao gerenciamento de recursos em especial, existem diversas maneiras de se hospedar diferentes aplicações em um único servidor. As abordagens tradicionais, geralmente se embasam em proteções de segurança de técnicas convencionais dos Sistemas Operacionais. Essas técnicas não garantem, de forma adequada, o isolamento de desempenho entre as aplicações. Prova disso é que as prioridades de escalonamento, demandas de memória, tráficos de rede e acessos ao disco de uma determinada aplicação podem impactar no desempenho das demais. Algumas iniciativas visam solucionar esse problema através da inserção de mecanismos de reserva de recurso no núcleo do Sistema Operacional, dando origem aos chamados *Resource Kernels* ou *Resource Containers* [Oikawa e Rajkumar 1999, Sundaram et al. 2000, Banga, Druschel e Mogul 1999]. Porém, tais mecanismos não garantem que todo uso de recurso seja contabilizado do processo correto. Considere, por exemplo, a complexa interação que existe entre aplicações que concorrem por um espaço na *cache* ou pela substituição de páginas em memória.

Monitores de Máquinas Virtuais são capazes de prover isolamento de desempenho entre os diferentes domínios presentes no *hardware* através da multiplexação dos recursos das máquinas. Apesar de existirem diversos monitores de máquinas virtuais tais como o KVM, VMWare e UMLinux, o Xen[2], um monitor de código aberto, vem adquirindo espaço entre pesquisadores da área de virtualização.

Este trabalho descreve, em detalhes, a arquitetura do Xen, um monitor de máquina virtual previamente desenvolvido para arquiteturas x86 que tem ganho especial destaque na literatura por permitir a diversas aplicações compartilharem recursos convencionais de maneira segura e sem grandes perdas de desempenho ou funcionalidade [Barham et al. 2003, Santhanam et al. 2005, Ramakrishnan et al. 2006, Wolinsky et al. 2006]. A escolha do Xen como objeto de estudo deste trabalho se deve principalmente pelo fato do seu código ser aberto e haver bastante documentação a respeito de sua arquitetura e funcionamento.

O objetivo deste documento é descrever os mecanismos de gerenciamento de recursos implementados pelo Xen como, por exemplo, qual é a política de escalonamento de MVs adotada pelo monitor, se essa política garante reserva de recursos, se o escalonamento é flexível e extensível, como é realizada a migração de processos e o balanceamento de carga entre diferentes máquinas, e se há alguma forma de controle de admissão para novos domínios.

Este trabalho está dividido da seguinte forma: a Seção 2 descreve a arquitetura do Xen destacando os diferentes componentes de um sistema virtualizado e as interações entre

[2]http://www.xensource.com/

eles. A Seção 3 apresenta a técnica de virtualização implementada pelo Xen e explica como essa técnica garante desempenhos muito similares aos encontrados em ambientes não-virtualizados. Na Seção 4, são expostos os mecanismos de gerenciamento de recursos encontrados no Xen. Nessa seção são discutidos os gerenciadores de carga e os escalonadores implementados, os processos envolvidos na migração de domínios, além de projetos que visam a expansão das funcionalidades no monitor tal como o escalonamento de MVs consciente de comunicação e a migração de domínios em WANs. A Seção 7 apresenta uma discussão do que foi exposto nas seções anteriores de modo a resumir como o gerenciamento de recursos é realizado no Xen. Por fim, a Seção 8 expõe as conclusões do trabalho.

2 Arquitetura

No projeto de desenvolvimento do Xen, um dos objetivos básicos é separar políticas dos mecanismos de implementação em si. Assim, apesar do monitor estar envolvido em tarefas como escalonamento de uma CPU entre domínios, não há nem mesmo a necessidade de ele ter conhecimento de questões de mais alto nível como, por exemplo, de que maneira essa CPU será compartilhada. A arquitetura resultante dessa prática modularizada entre políticas e mecanismos faz com que o monitor apresente somente operações de controle básicas exportadas para domínios autorizados através de interfaces. Decisões mais complicadas, tais como o controle de admissão, devem ser tratadas por sistemas de gerenciamento dos sistemas visitantes que apresentem permissão de acesso à interface de controle.

A figura 1 ilustra a estrutura de uma máquina física executando o monitor Xen com diferentes SOs clientes. A estrutura é formada por múltiplas camadas com diferentes privilégios, onde o monitor Xen pertence à camada com maior prioridade.

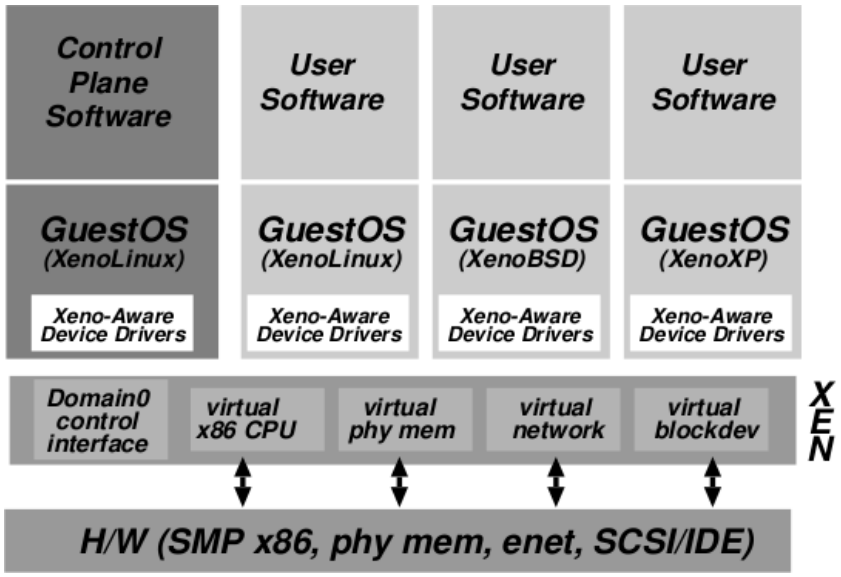

Figura 1: Estrutura de uma máquina executando o monitor Xen e diversos SOs clientes [Barham et al. 2003].

Durante a inicialização do sistema, um domínio, denominado *Domínio0*, é criado e habilitado para invocar métodos da interface de controle (na figura, esse domínio é repre-

sentado pelo *GuestOS* de cor mais escura). Essa interface provê meios para criar e finalizar domínios assim como para controlar os parâmetros de escalonamento, alocação de memória física e o acesso aos discos e dispositivos de rede da máquina local. Além disso, a interface de controle em questão permite a criação e a destruição de interfaces virtuais de rede e dispositivos virtuais de blocos os quais apresentam controles de acesso para a determinação de quais domínios podem acessá-los e com quais restrições. É através de um conjunto de aplicações que manipulam a interface de controle que os administradores de sistema gerenciam a execução das MVs em ambientes virtualizados com o Xen.

2.1 Transferência de Controle

A comunicação entre o monitor e os domínios ocorre via chamadas síncronas dos domínios para o monitor, e via notificações do monitor para os domínios. As chamadas síncronas são invocadas para a execução de operações privilegiadas do *kernel* e podem ser feitas através de *hypercalls*[3], enquanto as notificações são entregues através de mecanismos de eventos assíncronos. Esses eventos são usados, por exemplo, para indicar aos domínios que novos dados foram recebidos na rede ou que uma requisição ao disco virtual foi completada. Eventos pendentes são armazenados em um *bitmask* presente em cada domínio. Esse *bitmask* é atualizado pelo Xen antes de invocar um tratador de eventos (*callback*) especificado pelo Sistema Operacional visitante. O tratador de *callback* é responsável por reiniciar o conjunto de eventos pendentes e responder às notificações de forma apropriada.

2.2 Transferência de Dados

A presença de um monitor entre os Sistemas Operacionais e os dispositivos de E/S representa uma camada de *software* adicional a ser ultrapassada a cada operação de entrada e saída requisitada pelos domínios. Conseqüentemente, é essencial implementar mecanismos de transferência que permitam mover dados no sistema com o mínimo de sobrecarga possível. A solução proposta pelos desenvolvedores do Xen foi uma estrutura em anel assim como ilustrado na figura 2.

Um anel consiste em uma fila circular de descritores alocados a um domínio, mas acessíveis a partir do Xen. Esses descritores não contém dados de E/S, mas *buffers* de dados que são alocados indiretamente (*out-of-band*) pelos SOs visitantes e referenciados pelos descritores de entrada/saída. O acesso a cada anel envolve dois pares de ponteiros do tipo produtor-consumidor. Ao inserir requisições no anel, os domínios avançam o seu respectivo ponteiro produtor da requisição enquanto o Xen remove essas requisições para tratá-las avançando o respectivo ponteiro consumidor. As requisições provenientes do Xen são processadas de forma semelhante, exceto que o Xen passa a ser o produtor enquanto os domínios passam a ser os consumidores.

2.3 Segurança

O modelo de proteção de processadores Intel x86 (primeira infra-estrutura de testes do Xen) é composto por quatro anéis, onde o anel rotulado 0, e com maior prioridade, executa o Sistema Operacional; o anel 3, as aplicações dos usuários; enquanto os anéis 1 e 2

[3]Uma *hypercall* é uma abstração dos domínios para o *hypervisor* assim como uma chamada ao sistema é uma abstração da aplicação para o *kernel*. Domínios utilizam *hypercalls* para requisitar operações com privilégios ao monitor Xen.

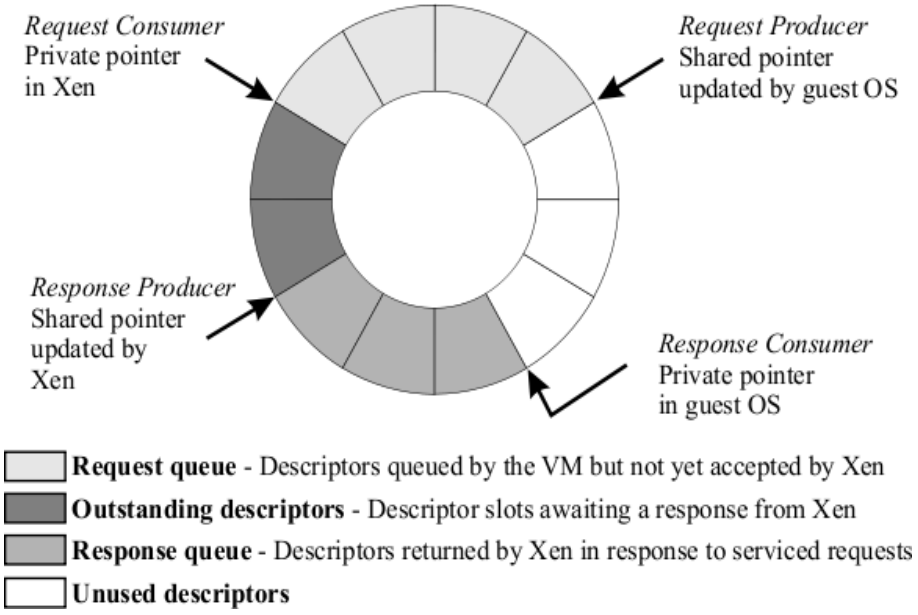

Figura 2: Estrutura de um anel com E/S assíncrona utilizado para a transferência de dados entre o Xen e os Sistemas Operacionais visitantes [Barham et al. 2003].

são raramente utilizados. No Xen, essa estrutura de anéis é alterada: o monitor executa no anel 0 enquanto os SOs visitantes executam no anel 1 [4] e as aplicações dos usuários no anel 3. Durante a inicialização, o Xen inicia um novo *kernel* alterado no anel 1, formando juntamente com o anel 0 o *Domínio0*. A figura 3 apresenta a utilização dos anéis dos processadores x86 para garantir a segurança entre os componentes de um sistema Xen. Nela, domainUs representam os domínios sem nenhum acesso especial ao *hardware* (*unprivileged domain*).

O *Domínio0*, como anteriormente exposto, apresenta privilégios especiais de gerenciamento sendo capaz de, por exemplo, criar novos domínios, destruí-los ou migrá-los. Dentro do domínio 0 é executado um processo *daemon* denominado *xend* responsável por funções de gerenciamento relacionadas às máquinas virtuais. Dessa maneira, o *xend* deve ser executado com privilégios de administrador do sistema.

3 Técnica de Virtualização

A técnica de virtualização adotada pelo Xen multiplexa recursos físicos na granularidade de um Sistema Operacional o que permite a co-existência de diversos SOs em uma mesma máquina garantindo-se o isolamento de desempenho entre eles. Por outro lado, essa fina granularidade resulta em sobrecargas elevadas para a inicialização de uma nova máquina virtual e o seu consumo de recursos já que cada MV corresponde a uma instância de um Sistema Operacional em execução.

No Xen, uma abstração de máquina virtual similar, mas não idêntica ao *hardware*

[4] Por esse motivo de alteração de prioridades entre os Sistemas Operacionais clientes e o monitor é que o último é chamado de *hypervisor*, uma analogia à superioridade do monitor em relação ao sistema operacional tipicamente chamado *supervisor*.

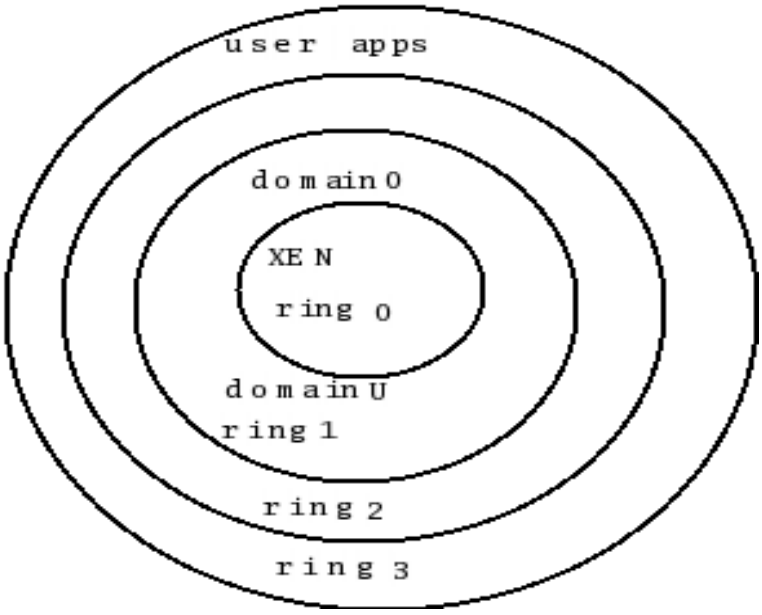

Figura 3: Uso da estrutura de segurança da arquitetura x86 baseada em anéis.

de nível mais baixo, é apresentada. Trata-se da paravirtualização, uma técnica onde o Sistema Operacional visitante deve ser modificado para ser executado no topo do monitor da máquina virtual. Essa modificação permite que servidores virtuais colaborem com o monitor para alcançar melhores desempenhos para as aplicações sendo executadas na máquina virtual. No caso do *hardware* da máquina física ser equipada com as tecnologias Intel VT-x ou AMD Pacifica em seu processador, os Sistemas Operacionais visitantes podem ser executados de forma nativa no *hardware* porém, apresentando uma significativa perda de desempenho.

A figura 4 apresenta a pilha de componentes existente em um sistema Xen. Na camada mais inferior, encontra-se o *hardware* da máquina. Acima dela, posiciona-se o monitor Xen composto por duas APIs: uma de gerenciamento e outra de virtualização dos dispositivos da máquina. No topo do monitor encontram-se os Sistemas Operacionais visitantes, os *drivers* de dispositivos e um componente de gerenciamento de código sendo esses dois últimos pertencentes ao *Domínio0*. Por fim, acima dos Sistemas visitantes, encontram-se as aplicações dos usuários. Nota-se na figura que o domínio apresentado por uma instância do SO Windows acessa diretamente o *hardware* da máquina hospedeira. Isso porque o Windows representa sistemas legados que são virtualizados em processadores habilitados com a tecnologia Intel VT ou AMD Pacifica [XenSource 2005].

3.1 Virtualização de CPU e Memória

Ao atualizar estruturas de dados do *hardware* tais como a tabela de páginas ou durante a inicialização de uma operação na DMA, o Sistema Operacional visitante colabora com o monitor da máquina virtual realizando chamadas à API oferecida pelo monitor. Dessa maneira, o monitor torna-se ciente de toda alteração realizada pelo SO e capaz de decidir de maneira mais apurada como gerenciar o estado das estruturas de dados do *hardware* durante as trocas de contexto.

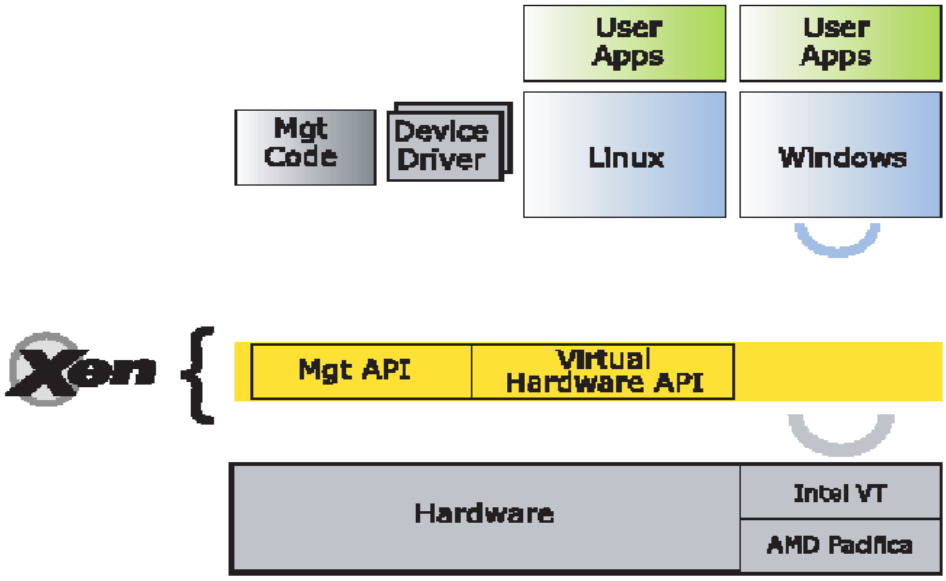

Figura 4: Pilha de componentes de um sistema Xen paravirtualizado [XenSource 2005].

No Linux, a API do monitor consiste em uma *jump table* populada durante o carregamento do *kernel*. Quando o *kernel* está executando em uma implementação nativa, isto é, diretamente no topo do *hardware* da máquina e não acima de um monitor de máquina virtual, a tabela é populada com operações nativas padrão. No caso do *kernel* estar executando no topo do Xen, a tabela é populada com *hypercalls* do Xen. Dessa maneira, um mesmo *kernel* pode ser executado de forma nativa ou virtualizada sem que haja a necessidade de re-certificar as aplicações como aptas à execução naquele ambiente.

Para evitar excesso de trocas de contexto entre os SOs e o monitor, este último é mapeado no espaço de endereço de cada SO visitante. A cooperação entre as entidades do Xen faz com que o monitor seja capaz de intervir na execução de uma máquina virtual, indicando, por exemplo, momentos em que a MV esteja excedendo um montante de uso de CPU previsto.

3.2 Virtualização de Entrada e Saída

A virtualização de entrada/saída é realizada através de um único conjunto de *drivers* para todo o sistema, incluindo o monitor e os SOs visitantes. Em cada SO visitante, *drivers* de dispositivos paravirtualizados substituem *drivers* de dispositivos específicos para as plataformas físicas. Os *drivers* paravirtualizados permitem uma virtualização de alto desempenho ao transferir o controle de processamento de E/S para o monitor sem que os Sistemas Operacionais visitantes tomem conhecimento.

Os *drivers* na arquitetura Xen executam fora da base do monitor, em um nível de segurança inferior ao deste último. Essa separação protege o monitor de falhas nos *drivers* de dispositivos e permite a ele o uso de qualquer *driver* disponível no mercado.

4 Gerenciamento de Recursos

Existem conceitos fundamentais para o bom funcionamento de ambientes formados por máquinas virtuais tais como segurança, alocação de recursos, isolamento de faltas e isolamento de desempenho de modo que a execução de uma aplicação em determinado domínio não afete a execução de aplicações em diferentes domínios.

O Xen permite a escolha de diferentes escalonadores de máquina virtual durante a sua inicialização. Na sua última versão (3.0), existem 3 implementações desses escalonadores os quais serão descritos em detalhes a seguir. Antes, porém, é necessário uma breve introdução aos termos utilizados na área de gerenciamento de carga de trabalho e escalonamento de processos [Cherkasova, Gupta e Vahdat 2007].

- **Compartilhamento Proporcional X Compartilhamento Justo** - Um escalonador de compartilhamento proporcional divide a CPU proporcionalmente ao peso atribuído a cada MV. De maneira diferente, um comportamento justo provê maior tempo de CPU às MVs que, por um determinado período, utilizaram menos capacidade de processamento;

- **Conservativo X Não-Conservativo** - Em escalonadores conservativos, o compartilhamento de CPU é apenas uma garantia de término em um determinado prazo estipulado. Nesse modo de divisão de processamento, a CPU só se encontra ociosa caso não haja nenhum cliente pronto para a execução. Por outro lado, no escalonamento não-conservativo, uma vez que a CPU é alocada para um processo, mesmo que esse encontre-se bloqueado à espera de E/S, a sua fatia de processamento não pode ser cedida a outro processo;

- **Preemptivo X Não-Preemptivo** - No escalonamento preemptivo, a CPU é alocada ao processo de maior prioridade entre os processos que se encontram no estado pronto para a execução. No caso de um escalonamento não-preemptivo, as decisões de alocação de CPU são feitas somente quando um processo cede a CPU, seja voluntariamente ou devido ao término da fatia de tempo alocada a ele.

4.1 Gerenciadores de Carga de Trabalho

As características dos escalonamentos proporcionais, conservativos e preemptivos permitem a criação de políticas de escalonamento dinâmicas onde a decisão de quais processos serão atribuídos ao processador, de que forma e em que momento são decididas durante a execução dos domínios virtuais. A variação no comportamento de execução das máquinas virtuais indica que políticas dinâmicas de alocação de recursos devem ser empregadas de modo a reagir a possíveis desbalanceamentos de carga. Porém, o emprego desse tipo de política no gerenciamento de CPU é desaconselhável visto que pode comprometer o isolamento de desempenho entre os domínios criados[5] [Cherkasova, Gupta e Vahdat 2007].

Para garantir o isolamento de desempenho, o Xen utiliza um gerenciador de carga baseado em um escalonador de compartilhamento proporcional utilizado no modo não-conservativo. Nesse escalonador, cada MV apresenta inicialmente uma taxa de utilização de (5 segundos por exemplo) reservada exclusivamente a ela, não importando se novas

[5] Ao atribuir maior processamento a um domínio, por exemplo, uma máquina virtual terá a sua fatia de CPU alocada reduzida.

cargas de trabalho são introduzidas no sistema. De acordo com a demanda de cada MV e da disponibilidade dos recursos, as alocações podem ser ajustadas para que cada MV atinja a capacidade de processamento necessária.

4.2 Escalonadores no Xen

O Xen inclui em sua inicialização opções de escalonamento similares aquelas que dividem o tempo de CPU pelos processos dos usuários. Na versão 2.0, a opção padrão consistia no algoritmo BVT(*Borrowed Virtual Time*). No BVT, a execução de MVs é monitorada em termos de *tempo virtual*, onde a MV com menor tempo virtual é escalonada primeiro. Para aplicações interativas e de tempo real, o BVT provê baixa latência permitindo às aplicações retornarem no tempo virtual e assim ganharem prioridade no escalonamento [Duda e Cheriton 1999]. A falta de um modo não-conservativo no BVT limitou o seu uso e levou ao desenvolvimento de um escalonador chamado *Simple Earliest Deadline First*(SEDF) [Cherkasova, Gupta e Vahdat 2007].

O SEDF utiliza algoritmos de tempo real para garantir os tempos de execução previamente determinados. Para ser executado, é preciso especificar 3 parâmetros (s_i,p_i,x_i), onde s_i representa a fatia de tempo de CPU que a MV receberá no período de tamanho p_i. O terceiro parâmetro x_i permite informar se o modo de escalonamento conservativo deve ou não ser habilitado para o Dom_i. Caso seja habilitado, o Dom_i é um candidato a receber ciclos de processamento extras caso a CPU não se encontre 100% ocupada.

Para cada domínio gerenciado pelo SEDF, uma variável d_i monitora o tempo de término do período corrente do domínio i, chamado *deadline*. O domínio com o menor *deadline* é escolhido para ser o próximo a executar. Nota-se que no SEDF o escalonamento é baseado em uma única fila por CPU, o que o inviabiliza para uso em máquinas multiprocessadas. Uma máquina duo-processada, por exemplo, com cada CPU processando domínios alocados com 80% de CPU cada, é incapaz de hospedar um novo domínio com necessidade de 30% de CPU.

Na versão 3.0, o escalonador padrão passou a ser baseado em créditos. Esse escalonador balanceia, de forma automática, a carga das CPUs virtuais dos sistemas clientes entre todas as CPUs reais de uma máquina SMP. O balanceamento global também permite que a porcentagem de processamento atribuída a uma MV seja dividido entre diferentes CPUs. No caso anterior da máquina duo-processada, por exemplo, o escalonador de Créditos é capaz de atender ao terceiro domínio atribuindo a ele 20% de processamento de uma CPU e 10% da outra.

No escalonador de Créditos, cada domínio, incluindo o Sistema Operacional hospedeiro, apresenta um peso e uma *flag* que, assim como no SEDF, determina se o escalonamento será realizado no modo conservativo (valor 0) ou não-conservativo (porcentagem que indica o limite de CPU que um domínio pode receber no modo não-conservativo). Um domínio com peso 512, por exemplo, terá o dobro de CPU alocada se comparado a um domínio de peso 256.

Cada CPU gerencia uma fila de CPUs virtuais (VCPUs) sendo executadas. Toda fila é ordenada pela prioridade de suas CPUs virtuais que pode assumir dois valores: "acima"e "abaixo", representando se uma determinada VCPU excedeu ou não, respectivamente, o seu uso de CPU durante o período corrente. Por padrão, cada período do escalonador de Créditos apresenta tempo de 30 ms e a utilização dos recursos é monitorada a cada 10 ms.

Ao ser executada, uma VCPU consome créditos que são debitados no saldo de cada

MV. Enquanto uma VCPU consome os créditos alocados a ela, a sua prioridade é "abaixo". Créditos negativos implicam em uma prioridade "acima". Durante uma decisão de escalonamento, a próxima VCPU a ser executada é obtida da cabeça da fila de cada CPU s(VCPU com prioridade mais "abaixo"). Quando uma CPU não encontra uma VCPU de prioridade "abaixo"em sua fila local, antes de entrar no estado ocioso, ela verifica com outras CPUs se há VCPUs que possam ser executadas por ela. Essa prática garante que não haja nenhum ciclo de CPU ocioso enquanto houver carga de trabalho a ser processada no sistema.

4.3 Escalonador Consciente de Comunicação

Para aplicações que realizam muita comunicação entre seus componentes é preciso prover processamento no momento correto para que não haja desperdício de CPU e nem atrasos no tempo de execução das aplicações. Os escalonadores propostos nas implementações do Xen não apresentam características que possibilitem a inserção de parâmetros que considerem a comunicação no momento de decisão do escalonamento. Em [Govindan et al. 2007] há uma proposta de um novo escalonador consciente de comunicação entre processos de uma aplicação. Nesse trabalho, foi desenvolvido um algoritmo de escalonamento que incorpora o comportamento de E/S das MVs no momento de decisão do escalonamento do escalonador SEDF descrito na Seção 4.2. O SEDF foi escolhido como base para o algoritmo, pois, no momento da implementação, o escalonador baseado em créditos ainda não estava disponível e o BVT apresenta um desempenho muito dependente da seleção correta de parâmetros pelo administrador do sistema.

O algoritmo proposto adapta seus parâmetros dinamicamente de acordo com a intensidade de E/S realizada pela aplicação. O objetivo principal do escalonador é reduzir o atraso agregado induzido por escalonamento aos domínios de um servidor sem que se deixe de prover garantias na alocação de CPU. Três fontes de atrasos induzidos por escalonamento podem ser identificadas:

- **Atraso associado ao escalonamento do *Domínio0*** - É o atraso relacionado *i)* à duração entre a recepção de um pacote na interface de rede física e o escalonamento do *Domínio0* seguido pela configuração de um canal de eventos para a notificação do domínio destinatário; ou *ii)* à duração entre o momento em que o domínio remetente copia um pacote para o anel de E/S do *Domínio0* e o momento em que o *Domínio0* é escalonado para enviar o pacote para a interface de rede física. Esses atrasos podem ser reduzidos com o escalonamento do *Domínio0 i)* imediatamente após um pacote ser recebido na interface de rede e *ii)* imediatamente após um domínio ter enviado um pacote à interface de rede, respectivamente;

- **Atraso associado ao escalonamento do Destinatário** - Corresponde ao atraso gerado no momento em que o *Domínio0* configura um canal de notificação para um domínio destinatário (chegada de um pacote) e o momento em que esse domínio é escalonado para o recebimento do pacote. Esse retardo pode ser reduzido com o escalonamento do domínio destinatário tão breve o recebimento de um pacote destinado a ele pelo *Domínio0*;

- **Atraso associado ao escalonamento do Remetente** - Consiste na diferença de tempo gerada quando há um domínio pronto para enviar um pacote mas o monitor escalona outros domínios para execução, e quando somente o domínio com dados a

ser enviados executa no sistema isoladamente. Esse atraso pode ser reduzido através da antecipação de quando um domínio estará pronto para o envio de um pacote e assim escalonando o mesmo para executar próximo a esse momento.

No primeiro tipo de fonte de atrasos, foi implementado um mecanismo que escalona o *Domínio0* nos momentos em que o mesmo está para causar retardos na execução das MVs, seja porque pacotes foram escritos nas interfaces de rede virtuais dos domínios visitantes, seja porque pacotes foram recebidos na interface de rede para esses domínios.

Já no segundo tipo de fonte de atrasos, foi implementado um algoritmo reativo que escalona um domínio logo após o recebimento de um pacote. Um problema ocorre se houver diversos domínios destinatários de pacotes a serem escalonados. Nesse caso, o algoritmo proposto utiliza uma heurística para escolher o domínio que mais será beneficiado pelo escalonamento prévio - aquele que recebeu o maior número de pacotes entre os domínios.

Já o atraso associado ao escalonamento tardio do remetente é tratado através da utilização de uma técnica bastante simples de predição do momento em que um domínio irá enviar um pacote para a interface de rede. A técnica se baseia na duração entre as últimas duas operações de transmissão de cada domínio. Se houver múltiplos domínios estimados para envio de dados, aquele com o maior número de pacotes é o escolhido como o primeiro a ser escalonado.

Foram realizados testes experimentais para a avaliação do trabalho proposto com aplicações de serviço e *benchmarks* para Internet. Através dos experimentos, conclui-se que o algoritmo consciente de comunicação garante aplicabilidade com benefícios de desempenho e custo bastante significativos. O *benchmark* TCP-W, por exemplo, exibiu melhorias de aproximadamente 35% para uma variedade de cenários enquanto um servidor de mídia hospedado em uma plataforma virtualizada foi capaz de atender satisfatoriamente 3.5 vezes mais clientes que um servidor executando o Xen padrão.

5 Migração de Domínios

Máquinas virtuais do Xen podem ser movidas de uma máquina a outra de maneira regular ou direta. Na maneira regular, antes de ser transferida, a máquina virtual é parada e tem seu conteúdo de memória copiado. Na máquina destino, a MV transferida é então reiniciada. Já na migração direta, a MV é transferida sem que nenhuma pausa na sua execução seja realizada.

No Xen, para haver a migração de domínios, ambas as máquinas envolvidas devem estar executando o *daemon xend*. Além disso, a máquina destino deve contar com recursos físicos suficientes para a acomodação do novo domínio.

O endereço MAC e IP de um domínio são movidos com ele durante a migração. Dessa maneira, a migração entre máquinas localizadas em diferentes sub-redes IP consiste em uma difícil tarefa, pois pode haver quebra de conexões e associações de rede. Esse motivo, associado à dificuldade de se transferir estados locais persistentes em disco, faz com que trabalhos relacionados à migração de domínios sejam, em sua maioria, realizados em infra-estruturas de teste compostas por uma única sub-rede [Clark et al. 2005, Hansen e Jul 2004, Huang et al. 2007].

5.1 Migração de Domínios em uma WAN

Bradford *et al* [Bradford et al. 2007] propuseram um sistema que permite a migração direta de MVs que utilizam armazenamento local e que apresentam conexões de rede abertas sem que interrupções severas dos serviços hospedados nas MVs aconteçam[6]. Para isso, o sistema proposto transfere o estado persistente de uma MV a ser migrada para o destino enquanto a MV ainda opera na máquina origem. Durante a transferência, um bloco no nível do usuário é utilizado para armazenar e enviar ao destino qualquer operação de escrita que aconteça na origem. Um redirecionamento temporário de rede é combinado com um anúncio do novo endereço da MV via DNS dinâmico para permitir à MV abrir novas conexões. Para que não haja degradação de desempenho de rede, a migração de processo pode ser limitada a uma taxa fixa de transferência.

O processo de migração se inicia com o cliente fonte contactando o *daemon* de migração executando na máquina destino assim como ilustra a figura 5. Se a migração for aceita, ocorre um processo de configuração e, em seguida, um estágio de transferência, onde se pré copia a imagem de disco da MV para o destino enquanto a MV ainda se encontra em execução. Então o sistema invoca a interface de migração do Xen para a migração incremental do estado de execução da MV.

Para garantir consistência entre os dados das MVs fonte e destino, operações de escrita ocorridas na fonte são interceptadas e geram *deltas* - unidades de comunicação que consistem nos dados escritos, na localização dos dados em disco e no tamanho dos dados. Então esses *deltas* são armazenados em uma fila na máquina destino para posterior aplicação nas imagens de disco. Se a taxa de leitura for muito alta para a migração finalizar, uma técnica de *write throttling* é utilizada. Nessa técnica, são definidos um *threshold* e um *delay*. Quando a MV atinge o número de escritas definida pelo *threshold*, cada operação de escrita posterior é atrasada de acordo com o parâmetro *delay*. Por fim, a MV origem é parada e a destino é inicializada. Nesse momento, o sistema de redirecionamento temporário de rede é iniciado caso a MV tenha sido migrada em um ambiente de Internet e, como conseqüência, tenha que ter o seu IP alterado.

A figura 6 provê uma visão dos componentes do sistema de migração e foca no tratamento das requisições de E/S durante a migração.

No destino, a autenticação, autorização e controle de acesso são tratados pelo *software* XenoServer que se comunica com o *daemon* de migração. Após a requisição de migração, o cliente da máquina fonte realiza um *fork* criando um processo *listener* separado e sinaliza ao bloco do *driver* de dispositivo para o mesmo entrar no modo de gravação. Nesse modo, o *driver* copia os detalhes de escrita realizados através do bloco de dispositivo ao processo *listener* o qual transfere os detalhes para o *daemon* destino. Paralelamente a esse processo de escuta, o cliente realiza transferências da imagem do disco no destino. No destino, o *daemon* de migração se divide em dois (*forks*) para ser capaz de receber os *deltas* e as transferências paralelamente.

O Xen suporta uma arquitetura do tipo *split-driver* para blocos de dispositivos. Nessa arquitetura, um *frontend* que provê um bloco de dispositivo no *kernel* da MV visitante (*blkfront*) se comunica com um *backend* em uma MV de armazenamento privilegiada via um *buffer* em anel. Para interceptar operações de escrita na fonte, um *framework* chamado *block tap (blktap)* é utilizado. Esse *framework* provê um pequeno módulo de *kernel* do

[6]Foi medido uma retardo de 3 segundos para a migração de um servidor web provendo conteúdo dinâmico a 250 clientes simultâneos em uma rede local

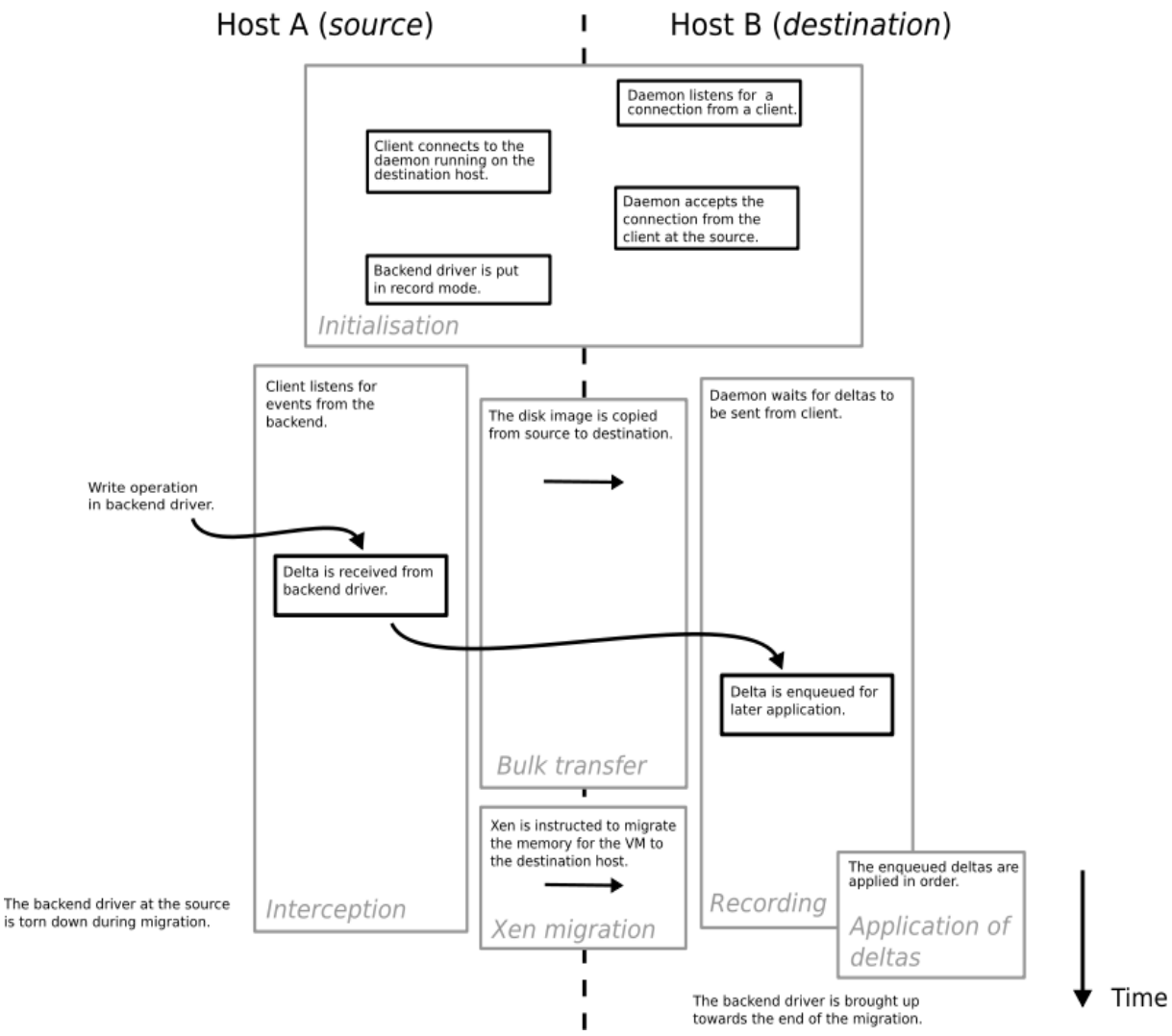

Figura 5: Processo de migração sem redirecionamento de rede e *write throttling* [Bradford et al. 2007].

tipo *backend* que permite o direcionamento de requisições para o *backend* de um processo no nível do usuário. Essa prática permite a implementação de um *driver* de blocos de dispositivo no espaço do usuário, garantindo segurança e praticidade.

Quando um acesso ao sistema de arquivos é realizado por um processo em uma MV visitante, uma requisição de E/S é passada através da camada de sistema de arquivos, do *frontend driver* e do *blktap driver* na MV de armazenamento para o *backend* no nível do usuário utilizando uma interface de memória compartilhada via um dispositivo de caracteres (operações 1-4 da figura 6). O *backend* então escreve na imagem do disco local e envia os detalhes da operação de escrita ao cliente de migração o qual empacota esses detalhes com um *delta* e o envia ao *daemon* (operações 5-9a).

Quando a migração de domínios acontece entre diferentes redes, um redirecionamento temporário de pacotes é realizado utilizando-se técnicas de tunelamento de IPs com DNS dinâmicos. Esse redirecionamento ocorre momentos antes da MV fonte ser parada. Com

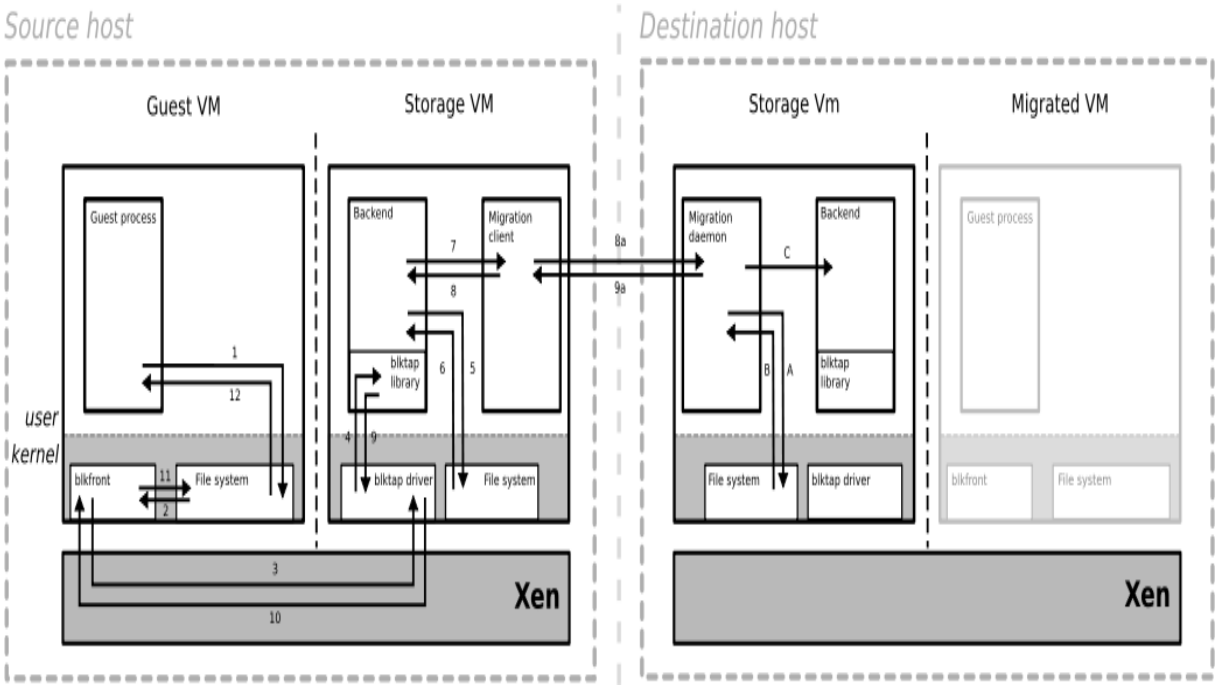

Figura 6: Implementação da arquitetura de migração [Bradford et al. 2007].

a ajuda da ferramenta *iproute2*, um tunelamento de IP é construído entre o endereço IP antigo da fonte o novo endereço IP do destino. Uma vez que a migração tiver sido completada e a MV puder responder através de seu novo endereço, a tabela de DNS é dinamicamente alterada para os serviços que a MV provê. O tunelamento é destruído quando não houver mais nenhuma conexão que utilize o endereço IP da MV fonte.

Experimentos de avaliação da migração de domínios foram realizados e mostraram que o sistema proposto não impacta de maneira significativa os serviços em execução na MV sendo migrada de forma direta. Além disso, a técnica apresentada supera o desempenho de migrações regulares do tipo *freeze-and-copy* por ordens de magnitude. Testes também mostraram que o sistema é capaz de lidar com cargas onde a MV realiza uma taxa muito alta de escrita.

6 Isolamento de Desempenho

Máquinas Virtuais são freqüentemente criadas a partir de contratos do tipo SLA (*Service Level Agreement*). Dessa maneira, o consumo de recursos por uma MV não deve impactar nas garantias de recursos prometidas a outras MVs na mesma máquina física. Apesar do Xen apresentar escalonadores que reservam parte dos recursos a cada MV, esses escalonadores não consideram os recursos consumidos pelo monitor em prol de determinados domínios como o que acontece, por exemplo, durante o processamento de requisições de entrada e saída.

O modelo de E/S do Xen é constituído por IDDs (*Isolated Driver Domains*) que hospedam *drivers* não-modificados dos dispositivos de rede e disco assim como ilustra a figura 7 [Gupta et al. 2006, Gupta, Gardner e Cherkasova 2005]. Esse modelo resulta

em um uso de CPU bastante complexo. Para aplicações com E/S intensa, o uso de CPU apresenta duas fases: o tempo de CPU consumido pelo domínio hospedeiro onde a aplicação reside e o tempo de CPU consumido pelo IDD que gerencia o *driver* e realiza o processamento em prol do domínio visitante. O processamento realizado pelo IDD não é debitado do tempo de processamento reservado para o domínio destinatário de um pacote de E/S. Essa deficiência resulta em quebras de isolamento de desempenho bastante grandes em sistemas virtualizados pelo Xen. Considere, por exemplo, um domínio visitante limitado ao uso de 30% de CPU. Se o processamento no IDD utilizar 20% de CPU relativos ao tratamento de E/S desse domínio visitante, então esse domínio totalizará 50% do total de CPU, o que potencialmente pode prejudicar o desempenho de outros domínios em execução.

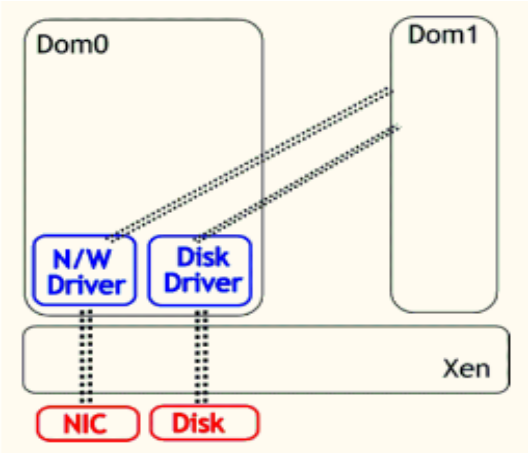

Figura 7: Arquitetura de E/S do Xen [Gupta, Gardner e Cherkasova 2005].

Objetivando solucionar o problema existente no Xen, Gupta *et al* propuseram um conjunto de mecanismos cooperativos que controlam o total de CPU consumido entre os domínios de forma efetiva. Esses mecanismos deram origem ao algoritmo de escalonamento denominado SEDF-DC (*Simple Earliest Deadline First - Debt Collector*). O SEDF-DC realiza de forma acurada o controle da quantidade de recursos que é consumida por cada domínio visitante além de atribuir o consumo de CPU nos IDDs aos domínios apropriados.

Para garantir o isolamento de desempenho entre os domínios, os seguintes componentes são utilizados:

- *XenMon* - Consiste em uma ferramenta de monitoração de uso de CPU em diferentes MVs. Ele inclui mecanismos para a medição de CPU no processamento de rede para cada domínio visitante [Gupta, Gardner e Cherkasova 2005];
- *SEDF-DC* - Um novo escalonador que aloca CPU entre domínios concorrentes de maneira efetiva ao contabilizar o consumo de processamento tanto do domínio quanto dos net-IDDs;
- *ShareGuard* - Um mecanismo de controle que garante um limite de processamento nos net-IDDs aos domínios visitantes.

Os passos para a contabilidade de E/S de rede são apresentados na figura 8. Ao receber um pacote (1), o dispositivo de *hardware* lança uma interrupção tratada pelo monitor Xen

(2). O Xen então determina qual é o domínio responsável por aquele dispositivo e agenda uma interrupção virtual ao *driver* de domínio correspondente via um canal de eventos (3). Quando o *driver* de domínio é escalonado, ele nota que há uma interrupção pendente e então invoca um tratador de interrupção apropriado. O tratador do *driver* de domínio só serve para remover o pacote do *driver* real do dispositivo (4) e entregá-lo ao *netback driver*[7] (5).

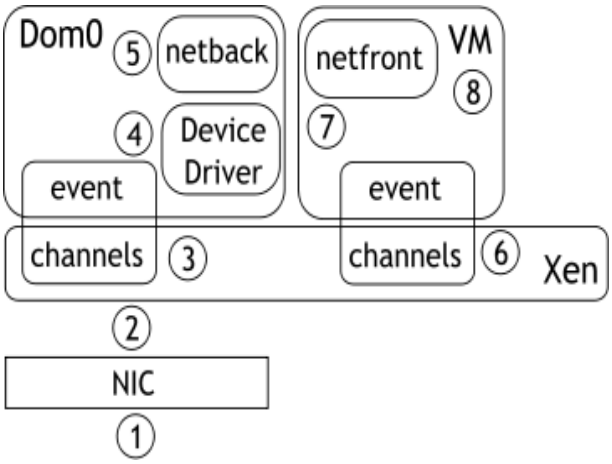

Figura 8: Processamento de E/S do Xen [Gupta et al. 2006].

É responsabilidade do *netback* enviar o pacote ao *netfront driver*. O *driver* de domínio então transfere a posse da página de memória contendo o pacote para o domínio destinatário e, em seguida, o notifica com uma interrupção virtual (6). Ao ser escalonado, o domínio alvo trata a interrupção pendente (7). Então, o *netfront driver* no domínio visitante passa o pacote às camadas mais altas da pilha de rede para posteriores processamentos (8).

O limite de uso de CPU pelos domínios é especificado assim como no SEDF, isto é, através de uma tupla do tipo (s_i,p_i), onde s_i representa a fatia de tempo dedicada a um determinado domínio a cada período p_i. Periodicamente, o SEDF-DC recebe informações do XenMon a respeito da CPU consumida pelos IDDs no processamento de requisições de E/S dos domínios. As informações são resultado de uma heurística baseada no número de pacotes enviados/recebidos por um domínio durante a monitoração do número de pacotes processados pelo net-IDD.

Dadas as informações providas pela heurística, o SEDF-DC restringe a alocação de CPU aos domínios alterando dinamicamente os valores atribuídos à tupla (s_i,p_i). Os parâmetros determinantes dessas informações são o *deadline* de cada domínio dentro de seus respectivos períodos e o montante de processamento gasto pelos mesmos em seus próprios domínios e no *Domínio0*. A inserção desse último parâmetro é o principal responsável pelo isolamento de desempenho no SEDF-DC.

Experimentos com o SEDF-DC mostraram que esse escalonador é capaz de limitar o uso combinado de CPU entre domínios, mas não é capaz de controlar o uso de CPU no *driver* de domínio. Dessa maneira, o ShareGuard foi proposto para garantir o isolamento de desempenho dentro dos *drivers* de domínio que, nesse caso, assume-se, estão sendo hospedados no *Domínio0*.

[7] *Back-end driver* que multiplexa a E/S para múltiplos *front-end drivers* nas MVs visitantes.

O ShareGuard consulta periodicamente o XenMon para obter os tempos de CPU consumidos pelos IDDs e, se um domínio visitante tiver excedido o seu limite de uso especificado, o ShareGuard interrompe o tráfego de rede desse domínio e abate o tempo de processamento excedente do próximo período do domínio.

Testes experimentais mostraram a eficácia do ShareGuard no isolamento de desempenho de domínios inclusive considerando-se o processamento realizado nos net-IDDs.

7 Discussão

Pelo exposto anteriormente neste trabalho, o gerenciamento de recursos em ambientes virtualizados pelo Xen é implementado pelo monitor e acessado via interfaces de controle exportadas para o *Domínio0*. Dessa maneira, cabe ao *Domínio0* gerenciar a criação e destruição de MVs, a migração de domínios e o controle de admissão de novas MVs. Políticas de escalonamento flexíveis e balanceamento de carga nas máquinas podem ser implementadas como módulos do SO executando no *Domínio0*. Essa possibilidade confirma a prática dos desenvolvedores do Xen em separar mecanismos de implementação de políticas de uso.

Uma vez que Sistemas Operacionais Linux não apresentam reserva de recurso de maneira geral, conclui-se que o Xen, por apresentar escalonadores tais como o SEDF que implementam reserva de recursos através da determinação de tuplas, implementa a garantia de reserva de processamento na camada de *software* correspondente ao monitor. Detalhes de como essa implementação de reserva é feita, porém, não são descritas nos documentos consultados. Não foi possível concluir também o porquê a política de escalonamento do Xen não poder ser alterada de maneira dinâmica. Dessa maneira, esses pontos tendem a ser objetos de um estudo futuro.

Por padrão, o *Domínio0* é o único domínio com acesso garantido à interface de controle exportada pelo monitor. Dessa maneira, qualquer chamada ao sistema para gerenciamento de recursos deve passar por esse domínio. Além disso, o *Domínio0* tipicamente hospeda os *drivers* dos diversos dispositivos das máquinas o que faz com que toda E/S de dados tenha que ser processada por ele. Por esses motivos, nota-se que o *Domínio0* pode consistir em um gargalo do sistema virtualizado. Estudos mais específicos quanto a essa característica devem ser realizados com o objetivo de analisar a arquitetura Xen e, possivelmente, identificar componentes que podem ser reimplementados de modo a apresentar melhores desempenhos.

8 Conclusão

Este documento apresentou uma revisão bibliográfica a respeito do monitor de máquinas virtuais Xen. Seções apresentaram a arquitetura desse monitor assim como a interação entre os seus componentes. Em especial, foram apresentadas seções sobre o gerenciamento de recursos do Xen com especial destaque ao isolamento de desempenho dos domínios e à migração de MVs entre diferentes redes de computadores. A técnica de virtualização implementada pelo Xen foi exposta em detalhes por dispositivos de uma máquina para que se entendesse como é possível garantir um desempenho tão próximo a ambientes não-virtualizados com a execução do Xen.

Este trabalho serve como um documento introdutório para a pesquisa com o monitor Xen, principalmente, se tratando do gerenciamento de recursos implementado por esse sistema. Com a análise desse material é possível notar que há muito trabalho a ser feito no sentido de desenvolver políticas de escalonamento e balanceamento de carga baseado na migração de processos. Os mecanismos para essas tarefas já se encontram implementados pelo monitor, bastando invocá-los nos momentos propícios.

Referências

[Adabala et al. 2005]ADABALA, S. et al. From virtualized resources to virtual computing grids: the in-vigo system. *Future Gener. Comput. Syst.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 21, n. 6, p. 896–909, 2005. ISSN 0167-739X.

[Banga, Druschel e Mogul 1999]BANGA, G.; DRUSCHEL, P.; MOGUL, J. C. Resource containers: A new facility for resource management in server systems. In: *In proceedings of the 3rd Conference on Symposium on Operating Systems Design & Implementation - OSDI '99*. New Orleans, USA: USENIX Association, 1999. p. 45–58.

[Barham et al. 2003]BARHAM, P. et al. Xen and the art of virtualization. In: *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*. New York, NY, USA: ACM Press, 2003. p. 164–177. ISBN 1-58113-757-5.

[Bradford et al. 2007]BRADFORD, R. et al. Live wide-area migration of virtual machines including local persistent state. In: *VEE '07: Proceedings of the 3rd international conference on Virtual execution environments*. New York, NY, USA: ACM Press, 2007. p. 169–179. ISBN 978-1-59593-630-1.

[Cherkasova, Gupta e Vahdat 2007]CHERKASOVA, L.; GUPTA, D.; VAHDAT, A. Comparison of the three cpu schedulers in xen. *SIGMETRICS Perform. Eval. Rev.*, ACM, New York, NY, USA, v. 35, n. 2, p. 42–51, 2007. ISSN 0163-5999.

[Clark et al. 2005]CLARK, C. et al. Live migration of virtual machines. In: *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*. Berkeley, CA, USA: USENIX Association, 2005. p. 20–20.

[Duda e Cheriton 1999]DUDA, K. J.; CHERITON, D. R. Borrowed-virtual-time (bvt) scheduling: supporting latency-sensitive threads in a general-purpose scheduler. In: *SOSP '99: Proceedings of the seventeenth ACM symposium on Operating systems principles*. New York, NY, USA: ACM Press, 1999. p. 261–276. ISBN 1-58113-140-2.

[Figueiredo, Dinda e Fortes 2003]FIGUEIREDO, R. J.; DINDA, P. A.; FORTES, F. A case for grid computing on virtual machines. *icdcs*, IEEE Computer Society, Los Alamitos, CA, USA, v. 00, p. 550, 2003. ISSN 1063-6927.

[Govindan et al. 2007]GOVINDAN, S. et al. Xen and co.: communication-aware cpu scheduling for consolidated xen-based hosting platforms. In: *VEE '07: Proceedings of the 3rd international conference on Virtual execution environments*. New York, NY, USA: ACM Press, 2007. p. 126–136. ISBN 978-1-59593-630-1.

[Gupta et al. 2006]GUPTA, D. et al. Enforcing performance isolation across virtual machines in xen. In: *Middleware*. [S.l.: s.n.], 2006. p. 342–362.

[Gupta, Gardner e Cherkasova 2005]GUPTA, D.; GARDNER, R.; CHERKASOVA, L. *XenMon: QoS Monitoring and Performance Profiling Tool*. [S.l.], October 2005. Disponível em http://www.hpl.hp.com/techreports/2005/HPL-2005-187.html. Acesso em setembro de 2008.

[Hansen e Jul 2004]HANSEN, J. G.; JUL, E. Self-migration of operating systems. In: *EW11: Proceedings of the 11th workshop on ACM SIGOPS European workshop: beyond the PC*. New York, NY, USA: ACM Press, 2004. p. 23.

[Huang et al. 2007]HUANG, W. et al. Nomad: migrating os-bypass networks in virtual machines. In: *VEE '07: Proceedings of the 3rd international conference on Virtual execution environments*. New York, NY, USA: ACM Press, 2007. p. 158–168. ISBN 978-1-59593-630-1.

[Keahey, Doering e Foster 2004]KEAHEY, K.; DOERING, K.; FOSTER, I. From sandbox to playground: Dynamic virtual environments in the grid. In: *GRID '04: Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*. Washington, DC, USA: IEEE Computer Society, 2004. p. 34–42. ISBN 0-7695-2256-4.

[Krsul et al. 2004]KRSUL, I. et al. Vmplants: Providing and managing virtual machine execution environments for grid computing. In: *SC '04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing*. Washington, DC, USA: IEEE Computer Society, 2004. p. 7. ISBN 0-7695-2153-3.

[Oikawa e Rajkumar 1999]OIKAWA, S.; RAJKUMAR, R. Portable RK: A portable resource kernel for guaranteed and enforced timing behavior. In: *IEEE Real Time Technology and Applications Symposium*. [S.l.: s.n.], 1999. p. 111–120.

[Ramakrishnan et al. 2006]RAMAKRISHNAN, L. et al. Grid allocation and reservation—toward a doctrine of containment: grid hosting with adaptive resource control. In: *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*. New York, NY, USA: ACM Press, 2006. p. 101. ISBN 0-7695-2700-0.

[Santhanam et al. 2005]SANTHANAM, S. et al. Deploying virtual machines as sandboxes for the grid. In: *II Workshop on Real, Large Distributed Systems (WORLDS 2005)*. San Francisco, CA: [s.n.], 2005.

[Sundaram et al. 2000]SUNDARAM, V. et al. Application performance in the qlinux multimedia operating system. In: *MULTIMEDIA '00: Proceedings of the eighth ACM international conference on Multimedia*. New York, NY, USA: ACM Press, 2000. p. 127–136. ISBN 1-58113-198-4.

[Wolinsky et al. 2006]WOLINSKY, D. et al. On the design of virtual machine sandboxes for distributed computing in wide area overlays of virtual workstations. In: *In proceedings of the First Workshop on Virtualization Technologies in Distributed Computing (VTDC)*. Tampa, Florida, USA: [s.n.], 2006.

[XenSource 2005]XENSOURCE. *Xen Source: Enterprise Grade Open Source Virtualization. Inside Xen 3.0: A XenSource White Paper*. December 2005. Documento disponível em http://www.xensource.com/files/xensource_wp2.pdf. Acesso em setembro de 2008.