Virtualização na arquitetura x86

Rodrigo Grumiche Silva
grumiche@inf.ufsc.br
Laboratório de Pesquisa em Sistemas Distribuídos
Departamento de Informática e Estatística
Universidade Federal de Santa Catarina

1 Introdução

Os conceitos e tecnologias para virtualização de computadores não são recentes, como mostra o survey realizado por Robert Goldberg em 1970 [1]. Durante muito tempo o uso de máquinas virtuais estava restrito aos *mainframes*. Porém no final da década de 90 e início dos anos 2000 com os processos de Consolidação e Simplificação de TI, que buscavam otimizar ao máximo o uso da infra-estrutura de informática, o uso de virtualização se tornou comum em servidores e computadores na plataforma x86.

O objetivo deste trabalho é fazer uma revisão sobre virtualização, aplicada a arquitetura de hardware x86, com o objetivo de entender as diferentes implementações e soluções tecnológicas disponíveis atualmente. Para tanto inicialmente é descrita uma classificação e taxonoma de máquinas virtuais, Em sequência é definida o que é um VMM - Machine Virtual Monitor, levantando-se as propriedades que um artefato de software deve ter para ser considerado uma VMM. Após, descreve-se o que seria uma implementação clássica de VMM. São apresentadas soluções de virtualização para a plataforma x86, fornecidas pela VMWare e Xen. Por fim, considerações finais são realizadas.

2 Máquinas Virtuais

As técnicas de virtualização de hardware pode ser aplicada a vários níveis de indireção entre os recursos reais de hardware e o que é fornecido ao software acima desses níveis, seja ele um programa em execução ou um sistema operacional rodando. James Smith e Ravi Nair em [2] analisam esses níveis de indireção, proponto uma classificação e taxonomia para máquinas virtuais. Estas são classificadas em dois tipos: as máquinas virtuais de processos e as máquinas virtuais de de sistema. Abaixo a classificação proposta é resumida, sendo possível visualizar a taxonomia na figura 1.

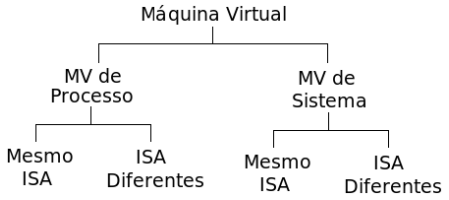

Figura 1. Classificação de máquinas virtuais[2]

2.1 Máquinas Virtuais de Processos

Máquinas virtuais de processo fornecem um ambiente de *Application Binary Interface - ABI* e de *Application Program Interface - API* para programas de usuário. É a forma mais comum e ubíqua de máquina virtual, pois é o tipo de mecanismo utilizado por sistemas operacionais tais como Linux, Windows e MacOS X para suportar a execução de múltiplos processos concorrentemente com compartilhando de recursos. Cada processo possui seu próprio espaço de endereçamento, pilha, registradores e tudo mais que compõe o seu contexto de execução.

Uma subclassificação é dada: máquinas virtuais de processos de mesmo ISA ou máquinas virtuais de processo de ISA diferentes. Na primeira, as instruções do programa em execução são executadas nativamente pelo hardware ou no máximo há o uso de otimizadores binários para melhorar a performance de um programa. Na segunda ocorre a tradução do código binário do programa em execução para o ISA da plataforma de hardware, que são diferentes entre si. Neste caso se enquadram por exemplo a máquina virtual java da Sun e a máquina virtual da Microsoft para .NET.

2.2 Máquinas Virtuais de Sistema

Fornecem ambiente completo onde vários sistemas operacionais com os seus processos em execução podem coexistir num único computador, de modo isolado e concorrente. A plataforma de hardware é replicada, dividindo os recursos de hardware entre os diversos ambiente de sistemas operacionais em execução. O sistema operacional é denominado de convidado por estar exatamente sendo executado em uma ambiente.

Os autores da taxonomia também classificam as máquinas virtuais de sistemas se as mesmas executam sistemas operacionais cujo as instruções em seu código binário são de mesmo ISA da plataforma de hardware ou se são de ISA diferentes.

3 Virtual Machine Monitor

O Virtual Machine Monitor é software que transforma uma única interface de máquina na ilusão de muitas. Cada uma dessas interfaces é uma réplica eficiente do sistema computacional original.[1] Essas interfaces são as máquinas virtuais,e o papel do VMM é acessar, gerenciar e multiplexar os recursos computacionais entre estas.

Gerald Popek e Robert Goldberg definiram em [3] três características essenciais que um artefato de software deve possuir para ser chamado de VMM. E a partir destas derivaram três propriedades também essenciais e construíram teoremas que formalizam os requisitos a serem atendidos por uma VMM. Um resumo das características é dado abaixo.

A primeira característica é a de que o VMM deve fornecer um ambiente para a execução de programas que seja essencialmente idêntico ao da máquina original. Os programas devem exibir o mesmo comportamente quando executados na máquina virtual que exibem quando executados diretamente na máquina original, com as excessões de diferenças causadas pela disponibilidade de recursos e de dependência de tempo.

Já a segunda característica é a eficiência, demandando que um subconjunto de instruções da arquitetura de um processador (ISA) seja executada diretamente, sem a intervenção da VMM. Ou seja, softwares emuladores e simuladores não são VMMs.

Por fim a terceira e última característica diz respeito ao controle de recursos. A VMM tem completo controle dos recursos sendo que não é possível a um programa em execução em seu ambiente criado acessar recursos que não foram explicitamente alocados para este ambiente, e é possível em determinadas circunstâncias que a VMM retome o controle de recursos alocados para um ambiente.

As propriedades derivadas das características acima são:

- Eficiência: Todas as instruções inócuas são executadas pelo hardware diretamente, sem nenhuma intervenção nenhuma da VMM.
- Controle de Recursos: Deve ser impossível que um programa arbitrário afete recursos de sistema;
- Equivalência: Qualquer programa executando sob uma VMM executa de modo indistinguível como se a VMM não existisse e o programa tivesse qualquer liberdade de acesso a instruções privilegiadas, com as duas excessões possíveis de disponibilidade de recursos e de dependência de tempo.

Cruzando-se as características e propriedades acima com a taxonomia proposta por Smith e Nair, as máquinas virtuais gerenciadas por VMMs se enquadram na classificação de máquinas virtuais de sistema de mesmo tipo de ISA. A partir deste ponto do texto, toda vez que o termo máquina virtual for utilizado no contexto de VMMs, estará se referindo a máquina virtual de sistema de mesmo ISA.

3.1 Implementação Clássica

A virtualização clássica respeita as propriedades definidas por Popek e Goldberg, e é baseada no princípio trap-and-emulate cujo as principais idéias são: desprivilegio, estruturas sombra e rastreamento[4].

A respeito de desprivilegio, numa arquitetura clássica virtualizável todas as instruções que lêem ou gravam um estado privilegiado e são executados num estado de desprivilegio geram uma *trap*. Um VMM clássico executa um SO diretamente no computador, porém com um nível reduzido de privilégio. O VMM captura as *traps* geradas por instruções do SO que tentar acessar um estado privilegiado e emula estas. Por exemplo, a execução de uma instrução desativando uma interrupção por sistema operacional convidado irá disparar uma *trap*. A mesma será capturada pela VMM, que a irá emular esta instrução para o sistema operacional.

Em sequência tem-se as estruturas de sombra. O estado privilegiado de uma máquina virtual difere do estado da máquina real e um dos papéis básicos da VMM é prover um ambiente de execução para o sistema operacional que esconda essas diferenças. Para tanto, para cada máquina virtual o VMM mantém estruturas de dados de sombra daquelas primárias do sistema operacional convidado. Como exemplos de estruturas tem-se registrador de máscara de interrupção e ponteiro para tabela de páginas de memória da máquina virtual.

Por fim, rastreamento diz respeito a controle de acesso e proteção a regiões de memória utilizada pelos diferentes sistemas operacionais convidados, VMM e dispositivos. É impedir que instruções de um SO convidado acessem áreas de memória de outros convidados, da própria VM ou de dispositivos que são mapeados em memória. Para tanto o acesso a estruturas primárias de gerenciamento de memória das máquinas virtuais são protegidas por um mecanismo de *trap*, e os acessos a essas áreas pelo sistema operacional convidado são tratadas do mesmo modo que o desprivilegio: o VMM captura a *trap* e emula o acesso a memória.

Uma implementação clássica de VMM necessita de que o hardware tenha suporte a virtualização, de modo que qualquer instrução que necessite acessar um recurso ou estado privilegiado e que seja executada num estado de desprivilegio cause uma *trap*. E esta *trap* então é tratara pelo VMM. A figura 2 mostra o que seria uma implementação clássica de um VMM.

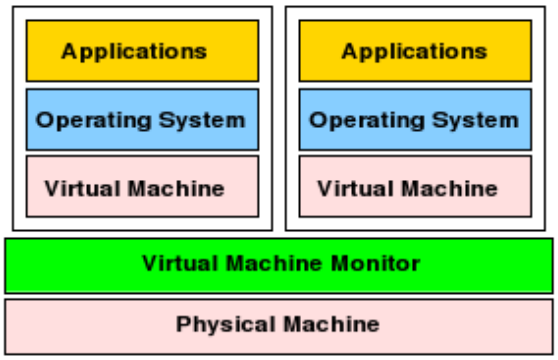

Figura 2. Arquitetura clássica de um VMM [5]

4 VMMs para as plataformas x86

A arquitetura x86 não foi projetada para ser classicamente virtualizável. Dentre as várias razões, em [4] são levantadas a visibilidade de estado privilegiado, pois o convidado pode observar que foi desprivilegiado quando lê o seletor de código de segmento (%cs) já que o nível do estado atual é armazenado nos dois últimos bits do mesmo; e por fim falta de *traps* quando instruções privilegiadas são executadas no nível de usuário.

Recentemente a Intel com a tecnologia Vanderpool e a AMD com a tecnologia Pacifica adicionaram extensões a esta arquitetura para suportar virtualização clássica. Algumas implementações de VMM disponíveis antes da disponibilização das extensões utilizam técnicas diversas para permitir a virtualização da arquitetura x86, e suas versões recentes já tiram benefício das extensões. Alternativas são VMMs que utilizam paravirtualização, o que é detalhado posteriormente, passaram a suportar virtualização a partir das extensões.

Abaixo são descritos algumas implementações de VMMs para a plataforma x86.

4.1 VMWare

A VMWare fornece uma família de produtos para virtualização da arquitetura x86 destinadas a diferentes tipos de uso. Dois produtos em especial, por possuírem características particulares, são descritos abaixo.

4.1.1 VMWare Workstation

O VMWare Workstation [6] é destinado a permitir a execução de múltiplos sistemas operacionais num único computador pessoal ou notebook de plataforma x86. Seu foco é o usuário comum que necessita executar outros sistemas operacionais e não deseja comprar outros computadores, nem deseja ter de instalar dois sistemas operacionais diferentes no mesmo computador e no momento da iniciação do equipamento fazer a escolha de qual será executado.

O VMWare Workstation segue a arquitetura de máquina virtual hospedada [5], vista na figura 3, que tira proveito de um sistema operacional existente para suporte de dispositivos de I/O. O mesmo é instalado como se fosse uma aplicação normal ao sistema operacional já existente, chamado de sistema operacional hospedeiro. A aplicação VMApp é utilizada para uso e gerenciamento de máquinas virtuais. Quando é executada a mesma utiliza o VMDriver, um driver instalado carregado no núcleo do sistema operacional, para estabelecer o privilégio do VMM do VMWare Workstation que executa diretamente no hardware. A partir deste momento a VMM e o sistema operacional hospedeiro compartilham tempo de processamento de CPU, escalonado pelo sistema operacional hospedeiro e utilizando o VMDriver para salvar e recuperar todas as informações de contexto do sistema operacional hospedeiro e da VMM.[5]

Um sistema operacional convidado e os programas em execução no mesmo utilizam as instruções de computação pura diretamente na CPU. Já quando realiza uma tarefa de I/O, a VMM intercepta, troca de contexto para que o sistema operacional hospedeiro comece a executar e encaminha a tarefa a VMWApp. A VMApp irá efetivamente realizar a atividade de I/O através das chamadas de sistema do SO hospedeiro.

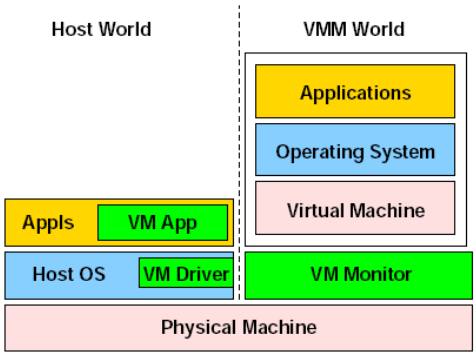

Figura 3. Arquitetura do VMWare Workstation [5]

Deste modo, todo o suporte a dispositivos existentes no SO hospedeiro fica disponível para a VMM. Tal arquitetura tem desvantagens: há perda de performance de I/O e o SO hospedeiro tem total controle dos recursos da máquina, incluindo podendo realizar paginação de memória das máquinas virtuais.

Para conseguir virtualizar a arquitetura x86, o VMM utilizam um mecanismo de tradução binária de instruções da ISA x86. Ou seja, não segue rigidamente o modelo de virtualização clássica. A entrada deste tradutor é um conjunto de instruções originais da arquitetura incluindo todas aquelas privilegiadas, e a saída é subconjunto seguro de instruções da própria ISA x86, formado em sua grande parte somente por instruções de modo usuário. A atividade de tradução é realizada de modo otimizado, em blocos de 12 (doze) instruções. Estes blocos formam fragmentos de código compilado que por sua vez compõem uma cache de tradução. Esses fragmentos de código compilado podem ser encadeados de modo a não ser necessário consultar a cache de tradução sempre que uma sequência de instruções é executada. Nenhuma melhoria ou otimização é realizada no código de instruções originais. A atividade de tradução é que é otimizada.

As versões mais recentes também possuem suporte as extensões da Intel e da AMD para virtualização da arquitetura x86. Porém em muitas CPUs o tratamento de *traps* é dispendioso, o que pode fazer implementação de um VMM usando tradução binária mais performático que um VMM clássica, por evitar *traps* disparadas por instruções privilegiadas. Experimentos realizados em [5] comparam entre uma implementação de VMM do VMWare Workstation que utiliza tradução binária e outra que utiliza as extensões ao ISA x86 para suportar virtualização em processadores Intel Pentium 4. O uso das extensões só se sobressaiu em desempenho nos testes de uso intenso de chamadas de sistema.

4.1.2 VMWare ESX Server

O VMWare ESX Server [7] é um produto destinada a *datacenters*, para virtualização de recursos de hardware em servidores. Ele é uma camada de software fina que multiplexa os recursos de hardware entre as máquinas virtuais. O projeto deste produto difere em muito ao WMWare Workstation, pois o ESX gerencia diretamente o hardware, permitindo uma melhor performance de I/O e completo controle sobre o gerenciamento de recursos[8].

Este produto também utiliza mecanismo de tradução binária para a virtualização da arquitetura x86 a semelhança do VMWare Workstation, mas que também pode utilizar as extensões de virtualização da Intel e da AMD. O VMM utiliza mecanismos de gerenciamento de memória mais sofisticados, que inclusive permite o compartilhamento de páginas de memória baseadas em seu conteudo entre diferentes máquinas virtuais[8].

4.2 Xen

O projeto Xen nasceu no Laboratório de Computação da Universidade de Cambridge, com o objetivo de implementar um VMM de alta performance para a plataforma x86. Para tanto utiliza uma técnica chamada de paravirtualização: ela expõe uma arquitetura virtual que é um pouco diferente da arquitetura física. A conseqüência disso é que quebra a compatibilidade com SOs existentes. Ou seja, somente é possível executar numa máquina virtual gerenciada pelo Xen um sistema operacional modificado para suportar os mecanismos de paravirtualização. Portanto o Xen não se enquadra na definição de Popek e Goldberg de VMM. Por essa razão e também para adotar a nomeclatura dos desenvolvedores, usa-se o termo *hypervisor* para identificar o artefato de software do Xen equivalente ao VMM. Assim o mesmo é denominado por estar sendo executado num privilégio maior que o código em modo supervisor dos sistemas operacionais convidados. Também será utilizado o termo domínio para para se referir a uma máquina virtual em execução onde está rodando um sistema operacional convidado.[9]. A figura 4 permite visualizar a arquitetura do Xen.

A interface de máquina virtual define a forma como os recursos são disponibilizados pelo hypervisor aos domínios. Em relação a memória os sistemas operacionais convidados são responsáveis por gerenciar suas próprias tabelas de páginas em hardware com a intervenção mínima do Xen; o Xen existe na sessão de 64MB

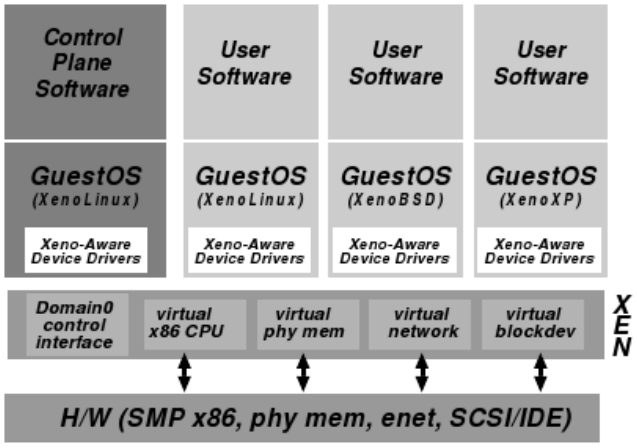

Figura 4. Arquitetura do Xen [9]

de todo espaço de endereçamento. Cada vez que o OS convidado requer uma nova página de memória, aloca e inicia a mesma a partir de seu própria reserva de memória e registra a mesma no Xen. Esta reserva é criada no momento da criação do domínio.O Xen fatia estaticamente e fisicamente a memória, garantindo forte isolamento. A partir deste momento o convidado abre mão de quaisquer direitos de escrita nela e todos as atualizações subsequentes a esta página precisam ser validadas pelo Xen. Desta maneira o Xen apenas permite a atualização de páginas pelo SO convidado que é proprietário da mesma. A região reservada de 64 MB do Xen não é acessível nem remapeável por SOs convidados.

Sobre as execução das instruções, os SOs convidados devem ser modificados para poderem ser executados um nível de privilégio inferior de modo a proteger o hypervisor. Instruções privilegiadas de um sistema operacional convidado são paravirtualizadas através de hypercalls, de forma análoga a chamadas de sistemas realizadas por um processo para um SO. Estas hypercalls geram *traps* que são tratadas pelo *hypervisor*. Já a comunicação entre o Xen e um domínio é realizada através de eventos, mecanismo assíncrono utilizado para simular interrupções de hardware e traps para o SO convidado.

Versões mais recentes do Xen já possuem suporte as extensões a arquitetura x86 para virtualização e conseguem executar SOs convidados nativamente, sem a necessidade de adaptá-los aos mecanismos de paravirtualização implementados pelo *hypervisor* [10]. Nesta situação, o hypervisor do Xen se enquadra na definição de VMM de Popek e Goldberg.

5 Conclusão

Uma revisão sobre classificação de máquinas virtuais e sobre VMMs é realizada, levantando as características e propriedades que um artefato de software deve ter para poder ser denominado de VMM.

Apesar de a arquitetura x86 não ter sido projetada para ser virtualizada, as implementações de VMM da VMWare utilizando o mecanismo de tradução binária permitem efetivamente virtualizar esta arquitetura. Foi visto também uma alternativa de paravirtualização, utilizada pelo Xen. Todas os produtos analisados também podem tirar proveito de extensões desenvolvidas pela Intel e pela AMD que permitem virtualizar a arquitetura x86. De modo que o Xen passe a suportar a virtualização de sistemas operacionais convidados sem a necessidade de que estes sejam modificados.

Referências

[1] R P Goldberg. Survey of virtual machine research. *Computer*, (7), 1974.

[2] J.E. Smith and Ravi Nair. The architecture of virtual machines. *Computer*, 38(5):32–38, May 2005.

[3] Gerald J. Popek and Robert P. Goldberg. Formal requirements for virtualizable third generation architectures. *Commun. ACM*, 17(7):412–421, 1974.

[4] Keith Adams and Ole Agesen. A comparison of software and hardware techniques for x86 virtualization. *SIGPLAN Not.*, 41(11):2–13, 2006.

[5] Jeremy Sugerman, Ganesh Venkitachalam, and Beng-Hong Lim. Virtualizing i/o devices on vmware workstation's hosted virtual machine monitor. In *Proceedings of the General Track: 2002 USENIX Annual Technical Conference*, pages 1–14, Berkeley, CA, USA, 2001. USENIX Association.

[6] Vmware workstation. Disponível em: http://www.vmware.com/products/ws/ . Acesso em: agosto 2008.

[7] Vmware esx. Disponível em: http://www.vmware.com/products/vi/esx/ . Acesso em: agosto 2008.

[8] Carl A. Waldspurger. Memory resource management in vmware esx server. *SIGOPS Oper. Syst. Rev.*, 36(SI):181–194, 2002.

[9] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM.

[10] Xiantao Zhang and Yaozu Dong. Optimizing xen vmm based on intel®virtualization technology. In *ICICSE '08: Proceedings of the 2008 International Conference on Internet Computing in Science and Engineering*, pages 367–374, Washington, DC, USA, 2008. IEEE Computer Society.