Virtualização e a nuvem

Em algumas situações, uma organização tem um multicomputador, mas na realidade não gostaria de tê-lo. Um exemplo comum ocorre quando uma empresa tem um servidor de e-mail, um de internet, um FTP, alguns de e-commerce, e outros. Todos são executados em computadores diferentes no mesmo rack de equipamentos, todos conectados por uma rede de alta velocidade, em outras palavras, um multicomputador. Uma razão para todos esses servidores serem executados em máquinas separadas pode ser que uma máquina não consiga lidar com a carga, mas outra é a confiabilidade: a administração simplesmente não confia que o sistema operacional vá funcionar 24 horas, 365 ou 366 dias sem falhas. Ao colocar cada serviço em um computador diferente, se um dos servidores falhar, pelo menos os outros não serão afetados. Isso é bom para a segurança também. Mesmo que algum invasor maligno comprometa o servidor da internet, ele não terá acesso imediatamente a e-mails importantes também — uma propriedade referida às vezes como caixa de areia (sandboxing). Embora o isolamento e a tolerância a falhas sejam conseguidos dessa maneira, essa solução é cara e difícil de gerenciar, por haver tantas máquinas envolvidas.

A ideia principal é que um Monitor de Máquina Virtual (VMM — Virtual Machine Monitor) cria a ilusão de múltiplas máquinas (virtuais) no mesmo hardware físico. Um VMM também é conhecido como hipervisor.

Hipervisores tipo 1 que são executados diretamente sobre o hardware (bare metal), e os hipervisores tipo 2 que podem fazer uso de todos os serviços e abstrações maravilhosos oferecidos pelo sistema operacional subjacente. De qualquer maneira, a virtualização permite que um único computador seja o hospedeiro de múltiplas máquinas virtuais, cada uma executando potencialmente um sistema operacional completamente diferente.

A vantagem dessa abordagem é que uma falha em uma máquina virtual não derruba nenhuma outra. Em um sistema virtualizado, diferentes servidores podem executar em diferentes máquinas virtuais, mantendo desse modo um modelo de falha parcial que um multicomputador tem, mas a um custo mais baixo e com uma manutenção mais fácil. Além disso, podemos agora executar múltiplos sistemas operacionais diferentes no mesmo hardware, beneficiar-nos do isolamento da máquina virtual diante de ataques e aproveitar outras coisas boas.

É claro, consolidar servidores dessa maneira é como colocar todos os ovos em uma cesta. Se o servidor executando todas as máquinas virtuais falhar, o resultado é ainda mais catastrófico do que a falha de um único servidor dedicado. A razão por que a virtualização funciona, no entanto, é que a maioria das quedas de serviços ocorre não por causa de um hardware defeituoso, mas de um software mal projetado, inconfiável, defeituoso e mal configurado, incluindo enfaticamente os sistemas operacionais.

Quando migrando para uma máquina virtual, tudo o que precisa ser movido são a memória e as imagens de disco, já que todas as tabelas do sistema operacional migram, também.

Outro uso para as máquinas virtuais é executar aplicações legadas em sistemas operacionais (ou versões de sistemas operacionais) que não têm mais suporte, ou que não funcionam no hardware atual.

Talvez o mais importante caso de uso de modismo para a virtualização hoje em dia é encontrado na nuvem. A ideia fundamental de uma nuvem é direta: terceirizar as suas necessidades de computação ou armazenamento para um centro de processamento de dados bem administrado e gerenciado por uma empresa especializada e gerida por experts na área. Como o centro de processamento de dados em geral pertence a outra empresa, você provavelmente terá de pagar pelo uso dos recursos, mas pelo menos não terá de se preocupar com as máquinas físicas, energia, resfriamento e manutenção. Graças ao isolamento oferecido pela virtualização, os provedores da nuvem podem permitir que múltiplos clientes, mesmo concorrentes, compartilhem de uma única máquina física. Cada cliente recebe uma fatia do bolo.

História

Já na década de 1960, a IBM realizou experiências com não apenas um, mas dois hipervisores desenvolvidos independentemente: SIMMON e CP-40.

Em 2000, a IBM lançou a série z, que suportava espaços de endereço virtual de 64 bits, mas por outro lado seguia compatível com o System/360. Todos esses sistemas davam suporte à virtualização décadas antes de ela tornar-se popular no x86.

Em 1974: Gerald Popek e Robert Goldberg publicaram um estudo seminal que listava exatamente quais condições uma arquitetura de computadores deveria satisfazer a fim de dar suporte à virtualização de maneira eficiente.

Os anos de 1970 foram muito produtivos, vendo também o nascimento do UNIX, Ethernet, o Cray-1, Microsoft e Apple.

Na realidade, a verdadeira revolução Disco começou na década de 1990, quando pesquisadores na Universidade de Stanford desenvolveram um novo hipervisor com aquele nome e seguiram para fundar a VMware, um gigante da virtualização que oferece hipervisores tipo 1 e tipo 2.

Em 1999 a VMware introduziu a sua primeira solução de virtualização para o x86 e vieram outros produtos (Xen, KVM, VirtualBox, Hyper-V, Parallels). A virtualização tornou-se popular com as massas, mas não era uma novidade.

Exigências para a virtualização

É importante que as máquinas virtuais atuem como o McCoy real. Em particular, deve ser possível inicializá-las como máquinas reais, assim como instalar sistemas operacionais arbitrários nelas, exatamente como podemos fazer nos hardwares reais. Cabe ao hipervisor

proporcionar essa ilusão e fazê-lo de maneira eficiente. De fato, hipervisores devem se sair bem em três dimensões:

1. Segurança: o hipervisor deve ter o controle completo dos recursos virtualizados.

2. Fidelidade: o comportamento de um programa em uma máquina virtual deve ser idêntico àquele do mesmo programa executando diretamente no hardware.

3. Eficiência: grande parte do código na máquina virtual deve ser executada sem a intervenção do hipervisor.

Uma maneira inquestionavelmente segura de executar as instruções é considerar uma instrução de cada vez em um interpretador (como o Bochs) e realizar exatamente o que é necessário para aquela instrução. Por exemplo, não podemos permitir de fato que o sistema operacional hóspede desabilite interrupções para toda a máquina ou modifique os mapeamentos da tabela de páginas.

Uma máquina será visualizável somente se suas instruções sensíveis forem um subconjunto das instruções privilegiadas.

A ideia básica do VT é criar contêineres nos quais as máquinas virtuais podem executar. Quando um sistema operacional hóspede é inicializado em um contêiner, ele continua a executar ali até causar uma exceção e gerar uma captura que chaveia para o hipervisor, por exemplo, executando uma instrução de E/S. O conjunto de operações que geram capturas é controlado por um mapa de bits do hardware estabelecido pelo hipervisor. Com essas extensões, a abordagem de máquina virtual clássica trap-and-emulate (captura e emulação) torna -se possível.

Antes de 2005, os hipervisores não executavam o sistema operacional hóspede original. Reescrita de parte do código durante a execução para substituir instruções problemáticas com sequências de código seguras que emulavam a instrução original (tradução binária).

Uma forma diferente de virtualização é conhecida como paravirtualização. Ela é bastante diferente da virtualização completa, pois nunca busca apresentar uma máquina virtual que pareça exatamente igual ao hardware subjacente. Em vez disso, apresenta uma interface de software semelhante a uma máquina que expõe explicitamente o fato de que se trata de um ambiente virtualizado. Por exemplo, ela oferece um conjunto de hiperchamadas (hypercalls), que permitem ao hóspede enviar solicitações explícitas ao hipervisor (assim como uma chamada de sistema oferece serviços do núcleo para aplicações). Convidados usam hiperchamadas para operações sensíveis privilegiadas como atualizar tabelas de páginas, mas como elas o fazem explicitamente em cooperação com o hipervisor, o sistema como um todo pode ser mais simples e mais rápido.

Comparada com a virtualização completa, o problema da paravirtualização é que o hóspede tem de estar ciente do API da máquina virtual. Isso significa que ela deve ser customizada explicitamente para o hipervisor.

É importante mencionar que nem toda tecnologia de virtualização tenta fazer o hóspede acreditar que ele tem o sistema inteiro. Às vezes, o objetivo é apenas permitir que um processo execute o que foi originalmente escrito para um sistema operacional e/ou arquitetura diferentes. Portanto, distinguimos entre a virtualização de sistema completa e a virtualização ao nível de processo.

Virtualização ao nível do processo. Ex: Wine (Windows Emulator) do Linux.

Hipervisores tipo 1 e tipo 2

Um tipo de hipervisor, chamado de hipervisor tipo 1. Tecnicamente, ele é como um sistema operacional, já que é o único programa executando no modo mais privilegiado. O seu trabalho é dar suporte a múltiplas cópias do hardware real, chamadas máquinas virtuais, similares aos processos que um sistema operacional normal executa.

Em comparação, um hipervisor tipo 2 é um programa que depende do, digamos, Windows ou Linux para alocar e escalonar recursos, de maneira bastante similar a um processo regular. É claro, o hipervisor tipo 2 ainda finge ser um computador completo com uma CPU e vários dispositivos. Ambos os tipos de hipervisores devem executar o conjunto de instruções da máquina de uma maneira segura. Por exemplo, um sistema operacional executando sobre o hipervisor pode mudar e até bagunçar as suas próprias tabelas de páginas, mas não as dos outros.

O sistema operacional executando sobre o hipervisor em ambos os casos é chamado de sistema operacional hóspede. Para o hipervisor tipo 2, o sistema operacional executando sobre o hardware é chamado de sistema operacional hospedeiro. O primeiro hipervisor tipo 2 no mercado x86 foi o VMware Workstation.

Hipervisores tipo 2, às vezes referidos como hipervisores hospedados, dependem para uma grande parte de sua funcionalidade de um sistema operacional hospedeiro como o Windows, Linux ou OS X. Quando ele inicializa pela primeira vez, age como um computador recentemente inicializado e espera para encontrar um DVD, unidade de USB ou CD-ROM contendo um sistema operacional na unidade. Dessa vez, no entanto, a unidade poderia ser um dispositivo virtual.

Técnicas para virtualização eficiente

Presuma, por ora, que temos um hipervisor tipo 1 dando suporte a uma máquina virtual. Como todos os hipervisores tipo 1, ele executa diretamente no hardware. A máquina virtual executa como um processo do usuário no modo usuário e, como tal, não lhe é permitido executar instruções sensíveis (no sentido Popek-Goldberg). No entanto, a máquina virtual executa um sistema operacional hóspede que acredita que ele está no modo núcleo (embora, é claro, ele não esteja). Chamamos isso de modo núcleo virtual. A máquina virtual

também executa processos do usuário, que acreditam que eles estão no modo usuário (e realmente estão).

O que acontece quando o sistema operacional hóspede (que acredita que ele está em modo núcleo) executa uma instrução que é permitida somente quando a CPU está de fato em modo núcleo? Em geral, em CPUs sem VT, a instrução falha e o sistema operacional cai. Em CPUs com VT, quando o sistema operacional hóspede executa uma instrução sensível, ocorre uma captura para o hipervisor. O hipervisor pode então inspecionar a instrução para ver se ela foi emitida pelo sistema operacional hóspede ou por um programa usuário na máquina virtual. No primeiro caso, ele arranja para que a instrução seja executada; no segundo caso, emula o que o hardware de verdade faria se confrontado com uma instrução sensível executada em modo usuário.

Virtualizando o “invirtualizável”

Os anéis de proteção do processador. Por muitos anos, o x86 deu suporte a quatro modos de proteção ou anéis. O anel 3 é o menos privilegiado. É aí que os processos do usuário normais executam. Nesse anel, você não pode executar instruções privilegiadas. O anel 0 é o mais privilegiado que permite a execução de qualquer instrução. Em uma operação normal, o núcleo executa no anel 0. Os dois anéis restantes não são usados por qualquer sistema operacional atual. Em outras palavras, os hipervisores eram livres para usá-los quando queriam.

Quanto às instruções sensíveis no código núcleo do hóspede: o hipervisor certifica-se de que elas não existem mais. Para fazê-lo, ele reescreve o código, um bloco básico de cada vez. Um bloco básico é uma sequência de instruções curta, em linha reta, que termina com uma ramificação.

Por outro lado, é comum realizar tradução binária em todo o código do sistema operacional hóspede executando no anel 1 e substituir mesmo as instruções sensíveis privilegiadas que, em princípio, poderiam ser obrigadas a gerar capturas também. A razão é que as capturas são muito caras e a tradução binária leva a um desempenho melhor.

Embora hipervisores tipo 2 sejam conceitualmente diferentes dos hipervisores tipo 1, eles usam, como um todo, as mesmas técnicas.

No entanto, para executar o código hóspede nativamente e usar exatamente as mesmas técnicas exige que o hipervisor tipo 2 manipule o hardware no nível mais baixo, o que não pode ser feito do espaço do usuário. Por exemplo, ele tem de estabelecer os descritores do segmento para exatamente o valor correto para o código hóspede. Para uma virtualização fiel, o sistema operacional hóspede também deve ser enganado para pensar que ele é o “rei” do pedaço, com o controle absoluto de todos os recursos da máquina e com acesso ao espaço de endereçamento inteiro.

Portanto, a maioria dos hipervisores modernos do tipo 2 tem um módulo núcleo operando no anel 0 que os permite manipular o hardware com instruções privilegiadas. É claro, não há problema algum em manipular o hardware no nível mais baixo e dar ao hóspede acesso

ao espaço de endereçamento completo, mas em determinado ponto o hipervisor precisa limpá-lo e restaurar o contexto do processador original.

Ir de uma configuração de hardware para o núcleo hospedeiro para uma configuração para o sistema operacional hóspede é conhecido como mudança de mundo.

Deve ficar claro agora por que esses hipervisores funcionam, mesmo em um hardware invirtualizável: instruções sensíveis no núcleo hóspede são substituídas por chamadas a rotinas que emulam essas instruções. Nenhuma instrução sensível emitida pelo sistema operacional hóspede jamais é executada diretamente pelo verdadeiro hardware. Elas são transformadas em chamadas pelo hipervisor, que então as emula.

Custo da virtualização

Alguém poderia imaginar ingenuamente que CPUs com VT teriam um desempenho muito melhor do que técnicas de software que recorrem à tradução, mas as mensurações revelam um quadro difuso (ADAMS e AGESEN, 2006). No fim das contas, aquela abordagem de captura e emulação usada pelo hardware VT gera uma grande quantidade de capturas, e capturas são muito caras em equipamentos modernos, pois elas arruínam caches da CPU, TLBs e tabelas de previsão de ramificações internas à CPU. Em comparação, quando instruções sensíveis são substituídas por chamadas às rotinas do hipervisor dentro do processo em execução, nada dessa sobrecarga de chaveamento de contexto é incorrida. Como Adams e Agesen demonstram, dependendo da carga de trabalho, às vezes o software bate o hardware. Por essa razão, alguns hipervisores tipo 1 (e tipo 2) realizam tradução binária por questões de desempenho, embora o software seja executado corretamente sem elas.

Virtualização da memória

Quase todos os sistemas operacionais modernos dão suporte à memória virtual, o que é basicamente um mapeamento de páginas no espaço de endereçamento virtual para as páginas da memória física. Esse mapeamento é definido por tabelas de páginas (em múltiplos níveis). Em geral, o mapeamento é colocado para funcionar fazendo que o sistema operacional estabeleça um registro de controle na CPU que aponte para a tabela de página no nível mais alto. A virtualização complica muito o gerenciamento de memória. Na realidade, os fabricantes de hardwares precisavam de duas tentativas para acertar.

Suponha, por exemplo, que uma máquina virtual está executando, e o sistema operacional hospedado nela decide mapear suas páginas virtuais 7, 4 e 3 nas páginas físicas 10, 11 e 12, respectivamente. Ele constrói tabelas de páginas contendo esse mapeamento e carrega um registrador de hardware para apontar para a tabela de páginas de alto nível. Essa instrução é sensível. Em uma CPU com VT, ela vai gerar uma captura; com a tradução dinâmica vai provocar uma chamada para uma rotina do hipervisor; em um sistema operacional paravirtualizado, isso vai gerar uma hiperchamada. Para simplificar, vamos presumir que ela gere uma captura para um hipervisor tipo 1, mas o problema é o mesmo em todos os três casos.

O que o hipervisor faz agora? Uma solução é realmente alocar as páginas físicas 10, 11 e 12 para essa máquina virtual e estabelecer as tabelas de páginas reais para mapear as páginas virtuais da máquina virtual 7, 4 e 3 para usá-las. Até aqui, tudo bem.

Agora suponha que uma segunda máquina virtual inicialize e mapeie suas páginas virtuais 4, 5 e 6 nas páginas físicas 10, 11 e 12 e carregue o registro de controle para apontar para suas tabelas de página. O hipervisor realiza a captura, mas o que ele vai fazer? Ele não pode usar esse mapeamento porque as páginas físicas 10, 11 e 12 já estão em uso. Ele pode encontrar algumas páginas livres, digamos 20, 21 e 22, e usá-las, mas primeiro tem de criar novas tabelas mapeando as páginas virtuais 4, 5 e 6 da máquina virtual 2 em 20, 21 e 22. Se outra máquina virtual inicializar e tentar usar as páginas físicas 10, 11 e 12, ele terá de criar um mapeamento para elas. Em geral, para cada máquina virtual o hipervisor precisa criar uma tabela de página sombra (shadow page table) que mapeie as páginas virtuais usadas pela máquina virtual para as páginas reais que o hipervisor lhe deu.

Pior ainda, toda vez que o sistema operacional hóspede mudar suas tabelas de página, o hipervisor tem de mudar as tabelas de páginas sombras também. O problema é que o sistema operacional hóspede pode mudar suas tabelas de página apenas escrevendo para a memória. Nenhuma operação sensível é necessária, então o hipervisor nem faz ideia da mudança e certamente não poderá atualizar as tabelas de páginas sombras usadas pelo hardware real.

Uma solução possível (mas desajeitada) é o hipervisor acompanhar qual página na memória virtual do hóspede contém a tabela de página de alto nível. Ele pode conseguir essa informação na primeira vez que o hóspede tentar carregar o registro do hardware que aponta para ele, pois essa instrução é sensível e gera uma captura. O hipervisor pode criar uma tabela de página sombra a essa altura e também mapear a tabela de página de alto nível e as tabelas de página para as quais ele aponta como somente leitura.

Outra solução tão desajeitada quanto é fazer exatamente o oposto. Nesse caso, o hipervisor simplesmente permite que o hóspede adicione novos mapeamentos às suas tabelas de páginas conforme sua vontade. À medida que isso está acontecendo, nada muda nas tabelas de páginas sombras.

Ambas as técnicas incorrem em muitas faltas de páginas, e tais faltas são caras. Em geral distinguimos entre faltas de páginas “normais” que são causadas por programas hóspedes que acessam uma página que foi paginada da RAM, e faltas de páginas que são relacionadas a assegurar que as tabelas de páginas sombras e as tabelas de páginas do hóspede estejam em sincronia. As primeiras são conhecidas como faltas de páginas induzidas pelo hóspede e, embora sejam interceptadas pelo hipervisor, elas precisam ser injetadas novamente no hóspede. Isso não sai nem um pouco barato. As segundas são conhecidas como faltas de páginas induzidas pelo hipervisor e são tratadas mediante a atualização de tabelas de páginas sombras.

Faltas de páginas são sempre caras, mas isso é especialmente verdadeiro em ambientes virtualizados, pois eles levam às chamadas saídas para VM (VM exit). Uma saída para VM é uma situação na qual o hipervisor recupera o controle. Considere o que a CPU precisa

fazer para que essa saída para VM aconteça. Primeiro, ela registra a causa da saída para VM, de maneira que o hipervisor saiba o que fazer. Ela também registra o endereço da instrução hóspede que causou a saída. Em seguida, é realizado um chaveamento de contexto, que inclui salvar todos os registros. Então, ela carrega o estado de processador do hipervisor. Apenas então o hipervisor pode começar a lidar com a falta de página, que era cara para começo de conversa. E quando tudo isso tiver sido feito, ela pode inverter os passos. Todo o processo pode levar dezenas de milhares de ciclos, ou mais. Não causa espanto que as pessoas façam um esforço para reduzir o número de saídas.

Suporte de hardware para tabelas de páginas aninhadas
O custo de lidar com tabelas de páginas sombras levou os produtores de chips a adicionar suporte de hardware para tabelas de páginas aninhadas. Tabelas de páginas aninhadas é o termo usado pela AMD. A Intel refere-se a elas como EPT

Elas são similares e buscam remover a maior parte da sobrecarga lidando com toda a manipulação de tabelas de páginas adicionais no hardware, tudo isso sem capturas. De maneira interessante, as primeiras extensões de virtualização no hardware x86 da Intel não incluíam nenhum suporte para a virtualização de memória. Embora esses processadores de VT-estendida removessem quaisquer gargalos relativos à virtualização da CPU, remexer nas tabelas de páginas continuava tão caro como sempre. Foram necessários alguns anos para a AMD e a Intel produzirem o hardware para virtualizar a memória de maneira eficiente.

O hardware “caminha” por essas tabelas de páginas para encontrar o endereço físico que corresponda ao endereço virtual. Acrescentar máquinas virtuais simplesmente acrescenta um mapeamento extra.

Em nosso exemplo, o hardware primeiro caminha pelas tabelas de páginas “regulares” para traduzir o endereço virtual do hóspede para um endereço físico do hóspede, da mesma maneira que ele faria sem virtualização. A diferença é que ele também caminha pelas tabelas de páginas estendidas (ou aninhadas) sem a intervenção do software para descobrir o endereço físico do hospedeiro, e ele precisa fazer isso toda vez que um endereço físico do hóspede seja acessado.

Todo nível na hierarquia de paginação incorre em uma busca nas tabelas de páginas aninhadas. Em outras palavras, o número de referências de memória cresce quadraticamente com a profundidade da hierarquia. Mesmo assim, o EPT reduz drasticamente o número de saídas para VM. Os hipervisores não precisam mais mapear a tabela de página do hóspede como somente de leitura e podem se livrar do tratamento de tabela de página sombra. Melhor ainda, quando alterna entre máquinas virtuais, ele apenas muda esse mapeamento, da mesma maneira que um sistema operacional muda o mapeamento quando alterna entre processos.

Recuperando a memória
Ter todas essas máquinas virtuais no mesmo hardware físico com suas próprias páginas de memória e todas achando que mandam é ótimo — até precisarmos da nossa memória de volta. Isso é particularmente importante caso ocorra uma sobrealocação (overcommit) da memória, onde o hipervisor finge que o montante total de memória para todas as máquinas

virtuais combinadas é maior do que o montante total de memória física presente no sistema. Em geral, essa é uma boa ideia, pois ela permite que o hipervisor admita mais e mais máquinas virtuais potentes ao mesmo tempo. Por exemplo, em uma máquina com 32 GB de memória, ele pode executar três máquinas virtuais, cada uma pensando que ela tem 16 GB de memória. Claramente, isso não funciona. No entanto, talvez as três máquinas não precisem de fato da quantidade máxima de memória física ao mesmo tempo. Ou talvez elas compartilhem páginas que têm o mesmo conteúdo (como o núcleo Linux) em diferentes máquinas virtuais em uma otimização conhecida como deduplicação. Nesse caso, as três máquinas virtuais usam um montante total de memória que é menor do que 3 vezes 16 GB.

A questão é: como podemos tirar páginas da memória com segurança se essa memória já foi dada para uma máquina virtual?

Em princípio, poderíamos ver outro nível ainda de paginação. No caso da escassez de memória, o hipervisor paginaria para fora algumas das páginas da máquina virtual, da mesma maneira que um sistema operacional poderia paginar para fora algumas das páginas de aplicação. O problema dessa abordagem é que o hipervisor deveria fazer isso, e ele não faz ideia de quais páginas são as mais valiosas para o hóspede. É muito provável que elimine as páginas erradas.

Uma solução comum é usar um truque conhecido como ballooning, onde um pequeno módulo balão é carregado em cada VM como um pseudodriver de dispositivo para o hipervisor. O módulo balão pode inflar diante da solicitação do hipervisor alocando mais e mais páginas marcadas, e desinflar “desalocando” essas páginas. À medida que o balão infla, a escassez de memória no hóspede diminui. O sistema operacional hóspede responderá paginando para fora o que ele acredita serem as páginas menos valiosas — o que é simplesmente o que queríamos. De maneira contrária, à medida que o balão desinfla, mais memória torna-se disponível para o hóspede alocar. Em outras palavras, o hipervisor induz o sistema operacional a tomar decisões difíceis por ele. Na política, isso é conhecido como empurrar a responsabilidade.

Virtualização de E/S

O sistema operacional hóspede tipicamente começará sondando o hardware para descobrir que tipos de dispositivos de E/S estão ligados. Essas sondas gerarão uma captura para o hipervisor. O que o hipervisor deve fazer? Uma abordagem é ele reportar de volta que os discos, impressoras e assim por diante são os dispositivos que o hardware realmente tem. O hóspede carregará drivers de dispositivos para eles e tentará usá-los. Quando os drivers do dispositivo tentam realizar a E/S real, eles lerão e escreverão nos registros de dispositivos de hardware do dispositivo. Essas instruções são sensíveis e gerarão capturas para o hipervisor, que poderia então copiar os valores necessários para e os registradores do hardware, conforme necessário.

Mas aqui, também, temos um problema. Cada SO hóspede poderia pensar que ele é proprietário de uma partição inteira de disco, e pode haver muitas máquinas virtuais mais (centenas) do que existem partições de disco reais. A solução usual é o hipervisor criar um arquivo ou região no disco real para cada disco físico da máquina virtual.

Também é possível que o disco que o hóspede está usando seja diferente do disco real. Por exemplo, se o disco real é algum disco novo de alto desempenho (ou RAID) com uma interface nova, o hipervisor poderia notificar para o SO hóspede que ele tem um disco IDE antigo simples e deixar que o SO hóspede instale um driver de disco IDE. Quando esse driver emite comandos de disco IDE, o hipervisor os converte em comandos para o novo disco. Essa estratégia pode ser usada para atualizar o hardware sem mudar o software.

MMUs de E/S

Outro problema de E/S que deve ser solucionado de certa maneira é o uso do DMA, que usa endereços de memória absolutos. Como poderia ser esperado, o hipervisor tem de intervir aqui e remapear os endereços antes que o DMA inicie. No entanto, o hardware já existe com uma MMU de E/S, que virtualiza a E/S da mesma maneira que a MMU virtualiza a memória. MMU de E/S existe em formas e formatos diferentes para muitas arquiteturas de processadores. Mesmo que nos limitássemos ao x86, Intel e AMD têm uma tecnologia ligeiramente diferente. Ainda assim, a ideia é a mesma. Esse hardware elimina o problema de DMA.

A MMU de E/S usa tabelas de páginas para mapear um endereço de memória que um dispositivo quer usar (o endereço do dispositivo) para um endereço físico. Em um ambiente virtual, o hipervisor pode estabelecer as tabelas de páginas de tal maneira que um dispositivo realizando DMA não irá pisotear a memória que não pertence à máquina virtual para a qual ele está trabalhando.

MMUs de E/S oferecem vantagens diferentes quando lidando com um dispositivo em um mundo virtualizado. A passagem direta do dispositivo (device pass through) permite que o dispositivo físico seja designado diretamente para uma máquina virtual em particular.

O isolamento de dispositivo assegura que um dispositivo designado a uma máquina virtual possa acessar diretamente aquela máquina virtual sem atrapalhar a integridade dos outros hóspedes. Em outras palavras, a MMU de E/S evita o tráfego de DMA “trapaceiro”, da mesma maneira que a MMU normal mantém acessos de memória “trapaceiros” longe dos processos — em ambos os casos, acessos a páginas não mapeadas resultam em faltas.

DMA e endereços não são toda a história de E/S, infelizmente. Para completar a questão, também precisamos virtualizar as interrupções, de maneira que a interrupção gerada por um dispositivo chegue à máquina virtual certa, com o número de interrupção certo. MMUs de E/S modernos, portanto, suportam o remapeamento de interrupções.

Por fim, ter uma MMU de E/S também ajuda dispositivos de 32 bits a acessar memórias acima de 4 GB. Em geral, tais dispositivos são incapazes de acessar (por exemplo, realizar DMA) para endereços além de 4 GB, mas a MMU de E/S pode facilmente remapear os endereços mais baixos do dispositivo para qualquer endereço no espaço de endereçamento físico maior.

Domínios de dispositivos

Uma abordagem diferente para lidar com E/S é dedicar uma das máquinas virtuais para executar um sistema operacional padrão e refletir todas as chamadas de E/S das outras para ela. Essa abordagem é incrementada quando a paravirtualização é usada, assim o comando emitido para o hipervisor realmente diz o que o SO hóspede quer (por exemplo, ler bloco 1403 do disco 1) em vez de ser uma série de comandos escrevendo para registradores de dispositivos, caso em que o hipervisor tem de dar uma de Sherlock Holmes e descobrir o que ele está tentando fazer. Xen usa essa abordagem para E/S, com a máquina virtual que faz E/S chamada domínio 0.

A virtualização de E/S é uma área na qual hipervisores tipo 2 têm uma vantagem prática sobre hipervisores tipo 1: o sistema operacional hospedeiro contém os drivers de dispositivo para todos os dispositivos de E/S esquisitos e maravilhosos ligados ao computador. Quando um programa aplicativo tenta acessar um dispositivo de E/S estranho, o código traduzido pode chamar o driver de dispositivo existente para realizar o trabalho. Com um hipervisor tipo 1, o hipervisor precisa conter o próprio driver, ou fazer uma chamada para um driver em domínio 0, que é de certa maneira similar a um sistema operacional hospedeiro. À medida que a tecnologia de máquinas virtuais amadurece, é provável que o hardware futuro permita que os programas aplicativos acessem o hardware diretamente de uma maneira segura, significando que os drivers do dispositivo podem ser ligados diretamente com o código da aplicação ou colocados em servidores de modo usuário separados (como no MINIX3), eliminando assim o problema.

Virtualização de E/S de raiz única
Designar diretamente um dispositivo para uma máquina virtual não é muito escalável.

Idealmente, a tecnologia de virtualização ofereceria a equivalência de um passe de dispositivo através de um único dispositivo para múltiplos hipervisores, sem qualquer sobrecarga. Virtualizar um único dispositivo para enganar todas as máquinas virtuais a acreditar que ele tem acesso exclusivo ao seu próprio dispositivo é muito mais fácil se o hardware realmente realiza a virtualização para você. No PCIe, isso é conhecido como virtualização de E/S de raiz única.

A virtualização de E/S de raiz única (SR-IOV — Single root I/O virtualization) nos permite passar ao longo do envolvimento do hipervisor na comunicação entre o driver e o dispositivo. Dispositivos que suportam SR-IOV proporcionam um espaço de memória independente, interrupções e fluxos de DMA para cada máquina virtual que a usa (Intel, 2011). O dispositivo aparece como múltiplos dispositivos separados e cada um pode ser configurado por máquinas virtuais separadas. Por exemplo, cada um terá um registro de endereço base e um espaço de endereçamento separado. Uma máquina virtual mapeia uma dessas áreas de memória (usadas por exemplo para configurar o dispositivo) no seu espaço de endereçamento.

SR-IOV proporciona acesso ao dispositivo em dois sabores: PF (Physical Functions — Funções Físicas) e VF (Virtual Functions — Funções Virtuais). PFs estão cheios de funções PCIe e permitem que o dispositivo seja configurado de qualquer maneira que o administrador considere adequada. Funções físicas não são acessíveis aos sistemas

operacionais hóspedes. VFs são funções PCIe leves que não oferecem tais opções de configuração. Elas são idealmente ajustadas para máquinas virtuais. Em resumo, SR-IOV permite que os dispositivos sejam virtualizados em (até) centenas de funções virtuais que enganam as máquinas virtuais para acreditarem que são as únicas proprietárias de um dispositivo.

Aplicações virtuais

Máquinas virtuais oferecem uma solução interessante para um problema que há muito incomoda os usuários, especialmente usuários de software de código aberto: como instalar novos programas aplicativos. O problema é que muitas aplicações são dependentes de inúmeras outras aplicações e bibliotecas, que em si são dependentes de uma série de outros pacotes de software, e assim por diante. Além disso, pode haver dependências em versões particulares dos compiladores, linguagens de scripts e do sistema operacional.

Com as máquinas virtuais agora disponíveis, um desenvolvedor de software pode construir cuidadosamente uma máquina virtual, carregá-la com o sistema operacional exigido, compiladores, bibliotecas e código de aplicação, e congelar a unidade inteira, pronta para executar. Essa imagem de máquina virtual pode então ser colocada em um CD-ROM ou um website para os clientes instalarem ou baixarem. Tal abordagem significa que apenas o desenvolvedor do software tem de compreender todas as dependências. Os clientes recebem um pacote completo que funciona de verdade, completamente independente de qual sistema operacional eles estejam executando e de que outros softwares, pacotes e bibliotecas eles tenham instalado. Essas máquinas virtuais “compactadas” são muitas vezes chamadas de aplicações virtuais. Como exemplo, a nuvem EC2 da Amazon tem muitas aplicações virtuais pré-elaboradas disponíveis para seus clientes, que ela oferece como serviços de software convenientes (“Software como Serviço”).

Máquinas virtuais em CPUs com múltiplos núcleos

A combinação de máquinas virtuais e CPUs de múltiplos núcleos cria todo um mundo novo no qual o número de CPUs disponíveis pode ser estabelecido pelo software. Se existem, digamos, quatro núcleos, e cada um pode executar, por exemplo, até oito máquinas virtuais, uma única CPU (de mesa) pode ser configurada como um computador de 32 nós se necessário, mas também pode ter menos CPUs, dependendo do software.

Além disso, máquinas virtuais podem compartilhar memória. Um exemplo típico em que isso é útil é um único servidor sendo o hospedeiro de múltiplas instâncias do mesmo sistema operacional. Tudo o que precisa ser feito é mapear as páginas físicas em espaços de endereços de múltiplas máquinas virtuais. O compartilhamento de memória já está disponível nas soluções de deduplicação. A deduplicação faz exatamente o que você pensa que ela faz: evita armazenar o mesmo dado duas vezes. Trata-se de uma técnica relativamente comum em sistemas de armazenamento, mas agora está aparecendo na virtualização também. No Disco, ela ficou conhecida como compartilhamento transparente de páginas (que exige modificações de acordo com o hóspede), enquanto o VMware a chama de compartilhamento de páginas baseado no conteúdo (que não exige modificação

alguma). Em geral, a técnica consiste em varrer a memória de cada uma das máquinas virtuais em um hospedeiro e gerar um código de espalhamento (hash) das páginas da memória. Se algumas páginas produzirem um código de espalhamento idêntico, o sistema primeiro tem de conferir para ver se elas são de fato as mesmas, e se afirmativo, deduplicá-las, criando uma página com o conteúdo real e duas referências para aquela página. Dado que o hipervisor controla as tabelas de páginas aninhadas (ou sombras), esse mapeamento é direto. É claro, quando qualquer um dos hóspedes modificar uma página compartilhada, a mudança não deve ser visível na(s) outra(s) máquina(s) virtual(ais). O truque é usar copy on write (cópia na escrita) de maneira que a página modificada será privada para o escritor.

Questões de licenciamento

Alguns softwares são licenciados em uma base por CPU, especialmente aqueles para empresas. Em outras palavras, quando elas compram um programa, elas têm o direito de executá-lo em apenas uma CPU.

Em alguns casos, vendedores de softwares colocaram uma cláusula explícita na licença proibindo a licença de executar o software em uma máquina virtual ou em uma máquina virtual não autorizada. Para empresas que executam todo o seu software exclusivamente em máquinas virtuais, esse poderia ser um problema de verdade. Se qualquer uma dessas restrições se justificará e como os usuários respondem a elas permanece uma questão em aberto.

Nuvens

A tecnologia de virtualização exerceu um papel crucial no crescimento extraordinário da computação na nuvem. Existem muitas nuvens. Algumas são públicas e estão disponíveis para qualquer um disposto a pagar pelo uso desses recursos; outras são de uma organização. Da mesma maneira, diferentes nuvens oferecem diferentes coisas. Algumas dão ao usuário acesso ao hardware físico, mas a maioria virtualiza seus ambientes. Algumas oferecem diretamente as máquinas, virtuais ou não, e nada mais, mas outras oferecem um software que está pronto para ser usado e pode ser combinado de maneiras interessantes, ou plataformas que facilitam aos usuários desenvolverem novos serviços. Provedores da nuvem costumam oferecer diferentes categorias de recursos, como “máquinas grandes” versus “máquinas pequenas” etc.

Cinco características essenciais:

1. Serviço automático de acordo com a demanda. Usuários devem ser capazes de abastecer-se de recursos automaticamente, sem exigir a interação humana.

2. Acesso amplo pela rede. Todos esses recursos devem estar disponíveis na rede por mecanismos padronizados de maneira que dispositivos heterogêneos possam fazer uso deles.

3. Pooling de recursos. O recurso de computação de propriedade do provedor deve ser colocado à disposição para servir múltiplos usuários e com a capacidade de alocar e realocar os recursos dinamicamente. Os usuários em geral não sabem nem a localização exata dos “seus” recursos ou mesmo em que país eles estão.

4. Elasticidade rápida. Deveria ser possível adquirir e liberar recursos elasticamente, talvez até automaticamente, de modo a escalar de imediato com as demandas do usuário.

5. Serviço mensurado. O provedor da nuvem mensura os recursos usados de uma maneira que casa com o tipo de serviço acordado.

As nuvens como um serviço

Especificamente, consideramos nuvens que oferecem acesso direto a uma máquina virtual, a qual o usuário pode usar da maneira que ele achar melhor. Desse modo, a mesma nuvem pode executar sistemas operacionais diferentes, possivelmente no mesmo hardware. Em termos de nuvem, isso é conhecido como IAAS (Infrastructure As A Service — Infraestrutura como um serviço), em oposição ao PAAS,, SAAS e muitos outros tipos.

• As nuvens como um serviço
– IaaS (Infraestrutura como um serviço)
Não define necessariamente um SO específico
– PaaS (Plataforma como um serviço)
Proporciona um ambiente completo incluindo, por exemplo,
SO, BD, servidor web
– SaaS (Software como um serviço)
Fornece softwares específicos, como aplicativos online

• Algumas nuvens podem se enquadrar em mais de uma
categoria (ex.: Amazon EC2)

As nuvens podem transformar a maneira como as empresas realizam computação. Como um todo, consolidar os recursos de computação em um pequeno número de lugares (convenientemente localizados próximos de uma fonte de energia e resfriamento barato) beneficia-se de uma economia de escala. Terceirizar o seu processamento significa que você não precisa se preocupar tanto com o gerenciamento da sua infraestrutura de TI, backups, manutenção, depreciação, escalabilidade, confiabilidade, desempenho e talvez segurança. Tudo isso é feito em um lugar e, presumindo que o provedor de nuvem seja competente, bem feito.

Migração de máquina virtual

A tecnologia de virtualização não apenas permite que nuvens IAAS executem múltiplos sistemas operacionais diferentes no mesmo hardware ao mesmo tempo, como ela também permite um gerenciamento inteligente.

Hipervisores desacoplam a máquina virtual do hardware físico. Em outras palavras, realmente não importa para a máquina virtual se ela executa nessa ou naquela máquina. Desse modo, o administrador poderia simplesmente derrubar todas as máquinas virtuais e reinicializá-las em uma máquina nova em folha. Fazê-lo, no entanto, resulta em um tempo parado significativo. O desafio é mover a máquina virtual do hardware que precisa de manutenção para a máquina nova sem derrubá-la.

Uma abordagem ligeiramente melhor pode ser pausar a máquina virtual, em vez de desligá-la. Durante a pausa, fazemos a cópia das páginas de memória usadas pela máquina virtual para o hardware novo o mais rápido possível, configuramos as coisas corretamente no novo hipervisor e então retomamos a execução. Além da memória, também precisamos transferir o armazenamento e a conectividade de rede, mas se as máquinas estiverem próximas, isso será feito relativamente rápido.

Em vez disso, o que as soluções de virtualização modernas oferecem é algo conhecido como migração viva (live migration). Em outras palavras, eles movem a máquina virtual enquanto ela ainda é operacional. Por exemplo, eles empregam técnicas como a migração de memória com pré-cópia. Isso significa que eles copiam páginas da memória enquanto a máquina ainda está servindo solicitações. A maioria das páginas de memória não tem muita escrita, então copiá-las é algo seguro. Lembre-se, a máquina virtual ainda está executando, então uma página pode ser modificada após já ter sido copiada. Quando as páginas da memória são modificadas, temos de nos certificar de que a última versão seja copiada para o destino, então as marcamos como sujas. Elas serão recopiadas mais tarde. Quando a maioria das páginas da memória tiver sido copiada, somos deixados com um pequeno número de páginas sujas. Agora fazemos uma pausa muito brevemente para copiar as páginas restantes e retomar a máquina virtual na nova localização. Embora ainda exista uma pausa, ela é tão breve que as aplicações em geral não são afetadas. Quando o tempo de parada não é perceptível, ela é conhecida como uma migração viva sem emendas (seamless live migration).

Checkpointing

O desacoplamento de uma máquina virtual e do hardware físico tem vantagens adicionais. Em particular, mencionamos que podemos pausar uma máquina. Isso em si é útil. Se o estado da máquina pausada (por exemplo, estado da CPU, páginas de memória e estado de armazenamento) está armazenado no disco, temos uma imagem instantânea de uma máquina em execução. Se o software bagunçar uma máquina virtual ainda em execução, é possível apenas retroceder para a imagem instantânea e continuar como se nada tivesse acontecido.

A maneira mais direta de gerar uma imagem instantânea é copiar tudo, incluindo o sistema de arquivos inteiro. No entanto, copiar um disco de múltiplos terabytes pode levar um tempo, mesmo que ele seja um disco rápido. E, de novo, não queremos fazer uma pausa por muito tempo enquanto estamos fazendo isso. A solução é usar soluções cópia na escrita (copy on write), de maneira que o dado é copiado somente quando absolutamente necessário.

Criar uma imagem instantânea funciona muito bem, mas há algumas questões. O que fazer se uma máquina estiver interagindo com um computador remoto? Podemos fazer uma imagem instantânea do sistema e trazê-lo de volta novamente em um estágio posterior, mas a parte que estava se comunicando pode ter partido há tempos. Claramente, esse é um problema que não pode ser solucionado.

Estudo de caso: VMware

Desde 1999, a VMware, Inc., tem sido a provedora comercial líder de soluções de virtualização com produtos para computadores de mesa, servidores, nuvem e agora até telefones celulares. Ela provê não somente hipervisores, mas também o software que gerencia máquinas virtuais em larga escala.