Nota: Este material complementar, disponível em https://prettore.github.io/lectures.html representa uma cópia resumida de conteúdos bibliográficos disponíveis gratuitamente na Internet.

Virtualização em Sistemas Operacionais

Introdução

A virtualização tornou-se uma tecnologia fundamental na computação moderna, transformando a maneira como os recursos de hardware são gerenciados e utilizados. Essencialmente, a virtualização introduz uma camada de abstração entre o hardware físico e os sistemas operacionais ou aplicações que rodam sobre ele, permitindo que múltiplos ambientes isolados coexistam em uma única máquina física (Tanenbaum & Bos, 2015; Silberschatz, Galvin, & Gagne, 2018). Essa capacidade de criar máquinas virtuais (VMs) ou containers oferece inúmeros benefícios, como a consolidação de servidores, a otimização do uso de recursos, maior flexibilidade administrativa, isolamento de ambientes para segurança e desenvolvimento, e a facilidade de provisionamento e migração de sistemas (Silva, Sousa, & Silva, 2017). O sistema operacional hospedeiro, ou um software especializado chamado hipervisor, gerencia essa camada de abstração, alocando recursos de CPU, memória, armazenamento e rede para cada ambiente virtualizado. Este resumo explora os conceitos centrais da virtualização, incluindo seus diferentes tipos, o papel dos hipervisores, a execução em hardware moderno, a comparação com containers e as técnicas de migração.

Tipos de Virtualização

A virtualização pode ser implementada de diferentes maneiras, cada uma com suas características e casos de uso específicos. As abordagens mais comuns são a virtualização total (ou completa) e a paravirtualização.

Na **virtualização total**, o hipervisor emula completamente o hardware subjacente para cada máquina virtual. Isso significa que o sistema operacional convidado (guest OS) não precisa de nenhuma modificação para rodar, pois ele acredita estar interagindo

diretamente com o hardware real. O hipervisor intercepta e traduz as instruções privilegiadas do guest OS, o que pode introduzir alguma sobrecarga de desempenho. Exemplos de tecnologias que utilizam virtualização total incluem VMware Workstation/ESXi e VirtualBox em suas configurações padrão (Tanenbaum & Bos, 2015). A principal vantagem é a compatibilidade, permitindo rodar praticamente qualquer sistema operacional sem alterações.

A **paravirtualização**, por outro lado, requer que o sistema operacional convidado seja modificado (ou 'paravirtualizado') para estar ciente de que está rodando em um ambiente virtualizado. Em vez de emular o hardware, o hipervisor expõe uma API específica que o guest OS utiliza para realizar operações privilegiadas. Essa comunicação direta entre o guest e o hipervisor elimina a necessidade de interceptar e traduzir instruções, resultando, geralmente, em melhor desempenho em comparação com a virtualização total. O Xen é um exemplo clássico de hipervisor que popularizou a paravirtualização, embora hipervisores modernos frequentemente combinem técnicas de virtualização total e paravirtualização (assistida por hardware) para otimizar desempenho e compatibilidade (Silberschatz, Galvin, & Gagne, 2018).

Existem também outras classificações, como a virtualização assistida por hardware, onde extensões específicas da CPU (como Intel VT-x e AMD-V) auxiliam o hipervisor na execução de instruções privilegiadas, melhorando significativamente o desempenho da virtualização total. A virtualização no nível do sistema operacional, exemplificada pelos containers, representa outra abordagem distinta, que será discutida posteriormente.

Hipervisores

O componente central que torna a virtualização possível é o hipervisor, também conhecido como Monitor de Máquina Virtual (VMM - Virtual Machine Monitor). O hipervisor é uma camada de software (ou firmware, ou mesmo hardware) que cria, executa e gerencia as máquinas virtuais ou containers. Ele é responsável por abstrair o hardware físico e alocar os recursos (CPU, memória, E/S) para cada ambiente virtualizado, garantindo o isolamento entre eles (Silberschatz, Galvin, & Gagne, 2018). Existem duas categorias principais de hipervisores:

Hipervisores Tipo 1 (Bare-metal): Estes hipervisores rodam diretamente sobre o hardware físico do servidor, sem a necessidade de um sistema operacional hospedeiro subjacente. Eles funcionam, essencialmente, como um sistema operacional minimalista otimizado para gerenciar VMs. Por terem acesso direto ao hardware, os hipervisores Tipo 1 geralmente oferecem melhor desempenho, escalabilidade e robustez, sendo a escolha predominante em ambientes de data center e computação em nuvem. Exemplos notáveis incluem VMware ESXi, Microsoft Hyper-V Server, Xen e KVM (Kernel-based Virtual Machine), que, embora integrado ao kernel Linux, opera de forma muito similar a um hipervisor bare-metal (Tanenbaum & Bos, 2015).

Hipervisores Tipo 2 (Hosted): Estes hipervisores rodam como uma aplicação sobre um sistema operacional convencional (hospedeiro), como Windows, macOS ou Linux. O sistema

operacional hospedeiro gerencia o hardware, e o hipervisor interage com o SO para obter acesso aos recursos necessários para as VMs. Hipervisores Tipo 2 são mais fáceis de instalar e gerenciar, sendo ideais para usuários finais, desenvolvedores e ambientes de teste que precisam rodar diferentes sistemas operacionais em uma única estação de trabalho. No entanto, a camada adicional do SO hospedeiro pode introduzir latência e sobrecarga de desempenho. Exemplos comuns incluem VMware Workstation, Oracle VirtualBox e Parallels Desktop (Tanenbaum & Bos, 2015; Silva, Sousa, & Silva, 2017).

A escolha entre Tipo 1 e Tipo 2 depende dos requisitos específicos de desempenho, escalabilidade, segurança e gerenciamento do ambiente de virtualização.

Máquinas Virtuais em CPUs Multicore

A proliferação de processadores multicore impactou significativamente a forma como a virtualização é implementada e gerenciada. Os hipervisores modernos são projetados para tirar proveito dessas arquiteturas, distribuindo as máquinas virtuais e seus processos entre os múltiplos núcleos disponíveis. Isso permite que várias VMs executem em paralelo de forma mais eficiente, melhorando o desempenho geral do sistema (Silberschatz, Galvin, & Gagne, 2018). O hipervisor é responsável pelo escalonamento das CPUs virtuais (vCPUs) das VMs nos núcleos físicos (pCPUs). Estratégias de escalonamento sofisticadas são empregadas para garantir a justiça na alocação de tempo de CPU, minimizar a contenção por recursos e otimizar a localidade de cache, por exemplo, tentando manter uma vCPU rodando no mesmo pCPU ou em núcleos dentro do mesmo soquete físico sempre que possível. Além disso, tecnologias como NUMA (Non-Uniform Memory Access), comuns em servidores multicore, exigem que o hipervisor gerencie a alocação de memória de forma inteligente, garantindo que as VMs acessem preferencialmente a memória local ao seu nó NUMA para evitar latências elevadas (Tanenbaum & Bos, 2015). A capacidade de alocar múltiplas vCPUs para uma única VM também permite que aplicações multithreaded dentro da VM se beneficiem diretamente da arquitetura multicore do hospedeiro.

Containers vs. Máquinas Virtuais

Embora tanto containers quanto máquinas virtuais ofereçam formas de isolamento e execução de aplicações em ambientes separados, eles operam em níveis de abstração diferentes e possuem características distintas. A escolha entre eles depende das necessidades específicas de isolamento, desempenho, densidade e gerenciamento.

As **Máquinas Virtuais (VMs)**, como discutido, utilizam um hipervisor para emular um conjunto completo de hardware (CPU, memória, disco, rede) para cada instância. Cada VM executa seu próprio sistema operacional convidado (guest OS) completo, isolado do sistema operacional hospedeiro e das outras VMs. Esse isolamento no nível do hardware e do kernel oferece um alto grau de segurança e compatibilidade, permitindo executar sistemas operacionais diferentes (e versões diferentes do mesmo SO) na mesma máquina física. No entanto, cada VM consome uma quantidade significativa de recursos (disco para o SO guest, memória RAM, sobrecarga do hipervisor), o que limita a densidade (número de instâncias

por host) e pode impactar o desempenho (Tanenbaum & Bos, 2015; Silberschatz, Galvin, & Gagne, 2018).

Os **Containers**, por outro lado, operam em um nível de abstração mais alto, virtualizando o próprio sistema operacional. Em vez de emular hardware, os containers compartilham o kernel do sistema operacional hospedeiro. Cada container executa como um processo isolado no espaço de usuário do host, com sua própria visão do sistema de arquivos, rede e identificadores de processo. Tecnologias como Docker e LXC utilizam recursos do kernel Linux (como namespaces e cgroups) para prover esse isolamento. Como não há um SO guest completo para cada container, eles são extremamente leves e rápidos para iniciar. Isso permite uma densidade muito maior de containers por host em comparação com VMs e oferece desempenho próximo ao nativo, pois não há a sobrecarga da emulação de hardware ou da tradução de instruções pelo hipervisor. A desvantagem principal é o menor nível de isolamento em comparação com as VMs; uma vulnerabilidade no kernel do host pode, teoricamente, afetar todos os containers. Além disso, todos os containers em um host devem ser compatíveis com o kernel do sistema operacional hospedeiro (e.g., containers Linux só rodam em hosts Linux) (Tanenbaum & Bos, 2015; Silva, Sousa, & Silva, 2017).

Em resumo, VMs são ideais quando é necessário um isolamento forte, a execução de sistemas operacionais diferentes ou legados, ou quando se precisa de controle total sobre o ambiente do SO. Containers são preferíveis para empacotar e distribuir aplicações (especialmente microsserviços), para ambientes de desenvolvimento e teste rápidos, e quando alta densidade e desempenho próximo ao nativo são cruciais.

Migração de Máquinas Virtuais

A capacidade de migrar máquinas virtuais entre hosts físicos é uma das vantagens mais significativas da virtualização, especialmente em ambientes de data center e nuvem. A migração permite realizar manutenção em hardware sem interromper os serviços, balancear a carga entre servidores, otimizar o consumo de energia e aumentar a resiliência a falhas. A forma mais avançada é a **migração ao vivo (Live Migration)** (Silberschatz, Galvin, & Gagne, 2018).

Na migração ao vivo, uma máquina virtual em execução é movida de um host físico para outro com interrupção mínima ou imperceptível para o usuário final e as aplicações rodando dentro da VM. O processo geralmente envolve várias etapas: primeiro, o estado da memória da VM é copiado do host de origem para o host de destino através da rede. Como a VM continua rodando durante essa cópia inicial, páginas de memória que são modificadas (páginas sujas ou ‘dirty pages’) precisam ser rastreadas e copiadas novamente em iterações subsequentes. Quando a taxa de modificação de páginas se torna baixa o suficiente, a execução da VM é brevemente pausada no host de origem, as últimas páginas sujas e o estado final da CPU são transferidos para o destino, e a execução da VM é retomada no novo host. O armazenamento da VM, geralmente localizado em uma rede de armazenamento compartilhada (SAN ou NAS), não precisa ser movido, apenas o acesso a ele é transferido para o novo host. Tecnologias como VMware vMotion e KVM Live Migration implementam esse processo (Tanenbaum & Bos, 2015). A migração ao vivo é crucial para a alta

disponibilidade e a flexibilidade operacional exigidas pelos ambientes de computação em nuvem modernos.

Conclusão

A virtualização representa uma mudança paradigmática na forma como os recursos computacionais são gerenciados e consumidos. Ao desacoplar o software do hardware subjacente através de hipervisores ou mecanismos de containerização, ela oferece flexibilidade, eficiência e escalabilidade sem precedentes. Desde a consolidação de servidores em data centers até o provisionamento rápido de ambientes de desenvolvimento e a arquitetura de microsserviços na nuvem, a virtualização tornou-se onipresente. A compreensão dos diferentes tipos de virtualização (total, paravirtualização), o papel dos hipervisores (Tipo 1 e Tipo 2), as nuances da execução em hardware multicore, as diferenças fundamentais entre VMs e containers, e as capacidades de migração ao vivo são essenciais para profissionais de TI e desenvolvedores. Embora traga inúmeros benefícios, a virtualização também introduz novas complexidades em termos de gerenciamento, segurança e otimização de desempenho, exigindo conhecimento especializado para sua implementação e manutenção eficazes. À medida que a tecnologia continua a evoluir, a virtualização permanecerá como um pilar fundamental da infraestrutura de TI moderna (Silberschatz, Galvin, & Gagne, 2018; Tanenbaum & Bos, 2015).

Referências

- Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). *Operating System Concepts* (10th ed.). Wiley.
- Tanenbaum, A. S., & Bos, H. (2015). *Modern Operating Systems* (4th ed.). Pearson Education.
- Silva, G. S., Sousa, J. F., & Silva, E. R. C. (2017). Virtualização de Sistemas Operacionais: Análise Comparativa entre os Ambientes Físico e Virtual. *Anais do Simpósio de Produção Científica*, UEMA. (Nota: Referência adaptada do artigo encontrado em Unibalsas, verificar dados completos se necessário para publicação formal).