

Sistemas Operacionais

Virtualização

Sumário

- 1 Histórico
- 2 Interface
- 3 Definição
- 4 Classificação

- 5 Processo
- 6 Hipervisores
- 7 Contêineres
- 8 Emulação
- 9 Bibliografia

Histórico

- Década de 1960: IBM desenvolve o sistema M44/44X e, depois, o OS/370, que oferece suporte à virtualização.
- Cada usuário tinha seu próprio ambiente monousuário, isolado dos demais.

Desinteresse Temporário (1980)

- › Popularização dos PCs baratos reduziu o interesse por virtualização.
- › PCs ofereciam desempenho limitado e pouco suporte à virtualização.
- › Era mais vantajoso fornecer um computador real a cada usuário do que investir em sistemas complexos.

- › Aumento no desempenho do hardware PC reaquece o interesse.
- › Java introduz o conceito de máquina virtual portátil.

- › Virtualização ganha destaque com a consolidação de servidores e surgimento da computação em nuvem.
- › Linguagens como Java e C# frequentemente são compiladas para máquinas virtuais portáteis.
- › Processadores modernos oferecem suporte nativo à virtualização no hardware.

Interface

- › Um sistema real é composto por:
 - › **Hardware** (CPU, chipset, dispositivos);
 - › **Sistema Operacional**;
 - › **Aplicações**.
- › O hardware executa operações solicitadas via sistema operacional.
- › O sistema operacional:
 - › Recebe requisições por chamadas de sistema;
 - › Controla o acesso aos recursos físicos;
 - › Coordena o uso compartilhado de memória e dispositivos.

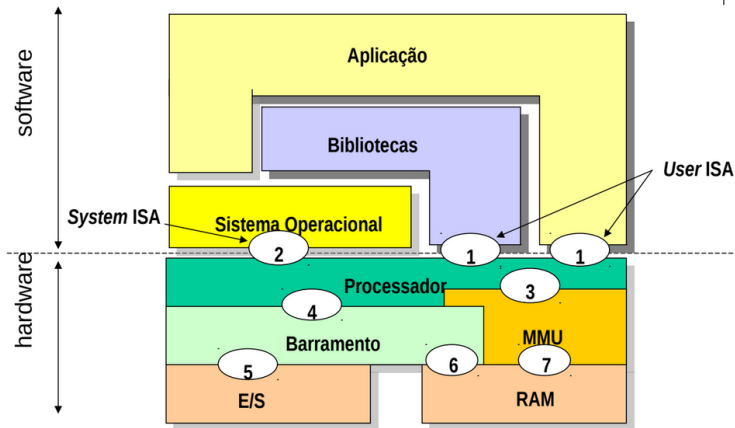


Figura 1: Arquitetura de Computadores (OLIVEIRA; CARISSIMI; TOSCANI, 2001).

- › Sistemas convencionais têm níveis de abstração empilhados.
- › Cada nível possui interfaces **bem definidas** e padronizadas.
- › Interfaces encapsulam os detalhes dos níveis inferiores.
- › Vantagens:
 - ›› Desenvolvimento independente por nível;
 - ›› Facilita manutenção e evolução do sistema.

- › Interface entre hardware e software.
- › Conjunto de instruções aceitas pelo processador.
- › Permite:
 - ›› Acesso à memória física;
 - ›› Manipulação de E/S;



- › Interface entre o **núcleo do SO** e os **processos do usuário**.
- › Permite o acesso controlado a:
 - ›› Dispositivos de entrada/saída;
 - ›› Memória;
 - ›› Instruções privilegiadas da CPU.
- › Exemplo: `read()`, `write()`, `fork()`, `exec()`.

- › Oferecem funções de alto nível para simplificar programas.
- › Muitas libcalls encapsulam syscalls.
- › Interface fornecida pelas bibliotecas é chamada de **API** (Application Programming Interface).
- › Exemplos de bibliotecas:
 - ›› **LibC** (UNIX): `fopen()`, `printf()`.
 - ›› **GTK+**: criação de interfaces gráficas.
 - ›› **SDL**: manipulação de áudio e vídeo.

- O conjunto de instruções (ISA – *Instruction Set Architecture*) é a interface entre o nível de abstração de hardware e o de software.
- Essa interface é composta por todos os códigos de máquina aceitos pelo processador, cada um representando uma instrução.
- Tipicamente, um processador possui dois modos de operação:
 - » **Modo não privilegiado:** usado por programas de usuário.
 - » **Modo privilegiado:** usado pelo núcleo do sistema operacional.
- A interface ISA é dividida em dois subconjuntos:
 - » **User ISA (Instruções de Usuário):**
 - Conjunto de instruções que podem ser executadas diretamente por programas de usuário.
 - Executadas em modo não privilegiado.
 - » **System ISA (Instruções de Sistema):**
 - Instruções que configuram o comportamento do processador ou acessam componentes de hardware diretamente.
 - Executadas exclusivamente pelo núcleo do sistema operacional, em modo privilegiado.

Chamadas de Biblioteca (Libcalls) III

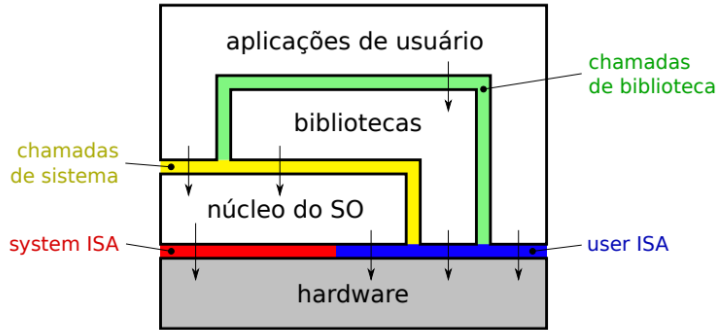


Figura 2: Componente e Interfaces (Maziero, 2019).

Chamadas de Biblioteca (Libcalls) IV

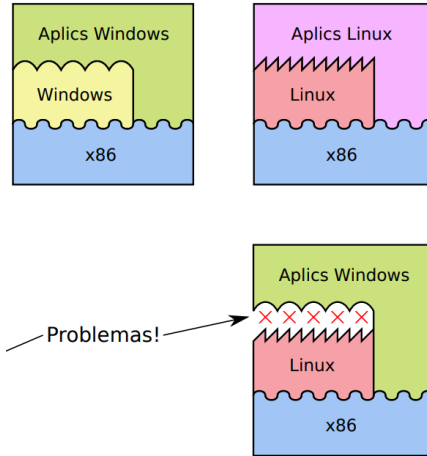


Figura 3: Problemas de Compatibilidade (Maziero, 2019).

Definição

Em sua essência, a virtualização consiste em estender ou substituir um recurso, ou uma interface existente por outro, de modo a imitar um comportamento. Isso é feito por intermédio de uma camada de software responsável por transformar ações de um sistema

(OLIVEIRA; CARISSIMI; TOSCANI, 2001)

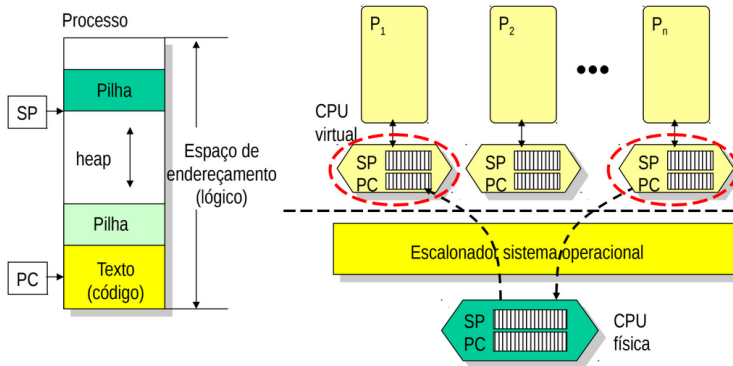


Figura 4: Abstração de Processos (OLIVEIRA; CARISSIMI; TOSCANI, 2001).

Usam processo nada mais é que um tipo de máquina virtual que executa um só programa.

(OLIVEIRA; CARISSIMI; TOSCANI, 2001)

- › **VLANs (Virtual LANs)**: abstração do **segmento de rede**.
- › **VPNs (Virtual Private Networks)**: abstração da **camada de enlace**.
- › **Java Virtual Machine (JVM)**: abstração da **plataforma de execução da aplicação**.
- › **RAID (Redundant Array of Independent Disks)**: abstração de **armazenamento**.

Classificação

Máquinas virtuais podem ser classificadas com base nas características do ambiente virtual que oferecem.

- **Máquina virtual de sistema:**

- » Emula uma plataforma de hardware completa (CPU, periféricos).
- » Suporta sistemas operacionais convidados + aplicações.
- » Exemplos: **KVM, VMware, VirtualBox.**

- **Máquina virtual de sistema operacional (contêineres):**

- » Suporta múltiplos espaços de usuário no mesmo núcleo.
- » Cada espaço tem recursos lógicos isolados (armazenamento, rede, IPC).
- » Exemplos: **Docker, Solaris Containers, FreeBSD Jails.**

- **Máquina virtual de processo:**

- » Suporte à execução de um único processo ou aplicação.
- » Também chamadas de *VMs de aplicação* ou *de linguagem*.
- » Exemplos: **Java Virtual Machine (JVM), Valgrind.**

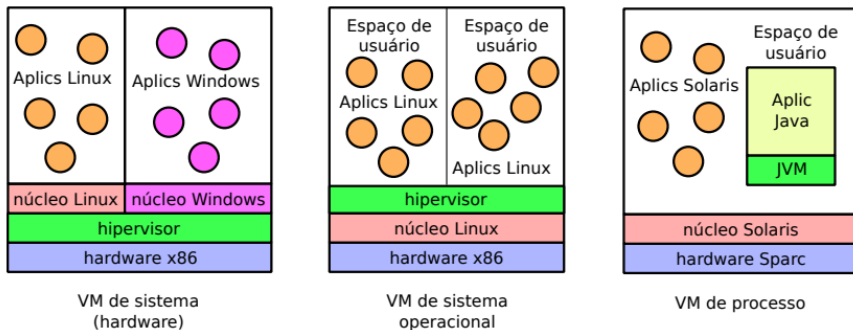


Figura 5: Máquina Virtual de sistema (tipo 1 e 2), container e de processo (Maziero, 2019).

Processo

- Uma máquina virtual de processo, de aplicação ou de linguagem (*Process Virtual Machine*) suporta a execução de um processo ou aplicação individual.
- Criada sob demanda no momento do lançamento da aplicação convidada, e destruída quando a aplicação finaliza.
- Hipervisores que implementam máquinas virtuais de processo permitem interação entre a aplicação convidada e outras aplicações do sistema via:
 - Mecanismos usuais de comunicação e coordenação entre processos, como mensagens, pipes e semáforos.
 - Acesso normal ao sistema de arquivos e outros recursos locais.

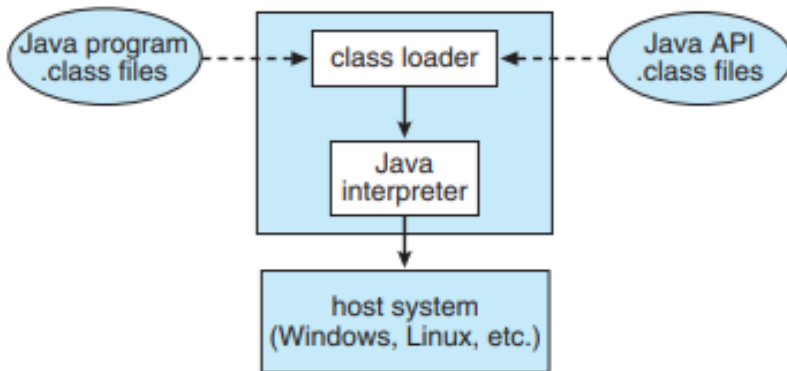


Figura 6: Máquina Virtual Java (STALLINGS, 2009).

Hipervisores

Máquinas virtuais de sistema (ou de hardware):

- › Emulam uma plataforma de hardware completa.
- › Permitem múltiplos **Sistemas Operacionais convidados** com suas aplicações.
- › Cada SO convidado tem a **ilusão de acesso exclusivo** ao hardware.
- › O **hipervisor** fornece interface ISA virtual e gerencia o acesso aos recursos reais.
- › Recursos como memória, disco e rede são virtualizados e isolados.
- › Alguns hipervisores permitem **compartilhamento controlado**, como diretórios entre VM e host.

› Hipervisor nativo (Tipo I):

- › Executa diretamente sobre o **hardware real**.
- › Virtualiza os recursos para as máquinas virtuais.
- › Ex.: **IBM OS/370, VMware ESX, Xen.**

› Hipervisor convidado (Tipo II):

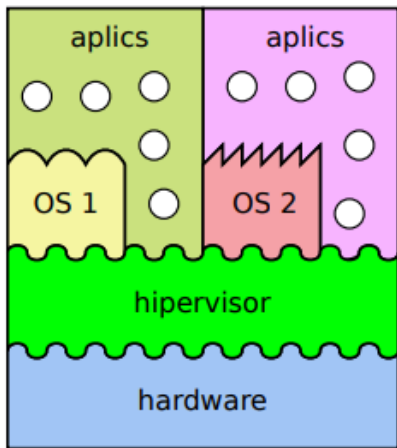
- › Executa como um processo em um SO já instalado.
- › Usa recursos do SO hospedeiro para fornecer VMs.
- › Ex.: **VMware Workstation, KVM, VirtualBox.**

› Tipo I (nativo):

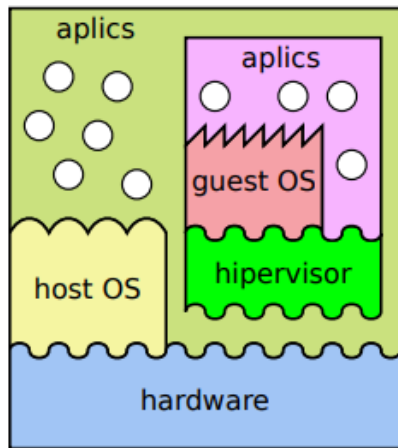
- ›› Maior desempenho: acesso direto ao hardware.
- ›› Requer ambiente dedicado.

› Tipo II (convidado):

- ›› **Mais flexível:** pode ser instalado sob demanda.
- ›› Usa recursos do SO hospedeiro.
- ›› Desempenho inferior ao Tipo I.



hipervisor nativo



hipervisor convidado

Figura 7: Arquiteturas de máquinas virtuais de sistema (Maziero, 2019).

Contêneres

- **Motivação:** isolamento entre subsistemas para maior segurança.
- **Uso típico:** consolidação de servidores (Web, e-mail, DNS etc.) em uma única máquina física.
- **Problema nas VMs de sistema:** alto custo de desempenho por virtualizar hardware e instruções.
- **Solução eficiente:** **virtualizar o espaço de usuário** (userspace) do sistema operacional.

Máquinas virtuais de sistema operacional (ou contêineres):

- › Dividem o espaço de usuário em domínios isolados.
- › Recursos como memória, CPU, disco e rede são **virtualizados por domínio**.
- › Cada domínio possui:
 - » Interface de rede virtual;
 - » Espaço de nomes (usuários, processos, diretórios, IPC etc.).
- › Sem visibilidade entre domínios diferentes.

Exemplo: Estrutura de Domínios Virtuais

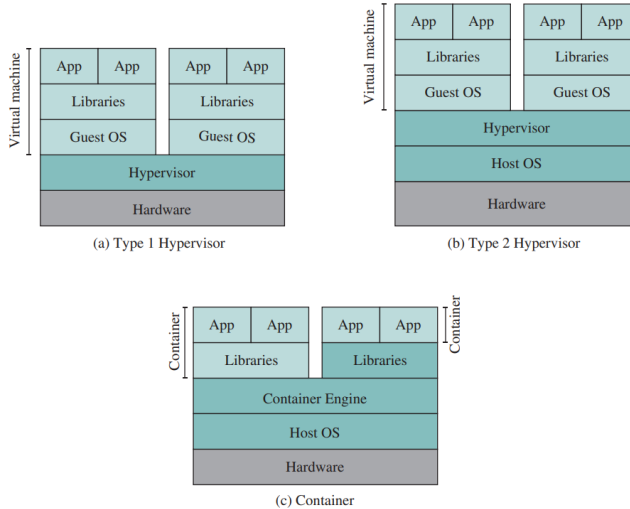


Figura 8: Ambiente com múltiplos domínios isolados (Maziero, 2019).

chroot(): Isolamento do Filesystem

chroot("/caminho"): restringe a visão do processo à subárvore de diretórios:

- Hierarquia a partir do novo diretório torna-se o "/".
- Filhos herdam essa restrição.
- Muito usada em servidores (e.g., DNS, e-mail).

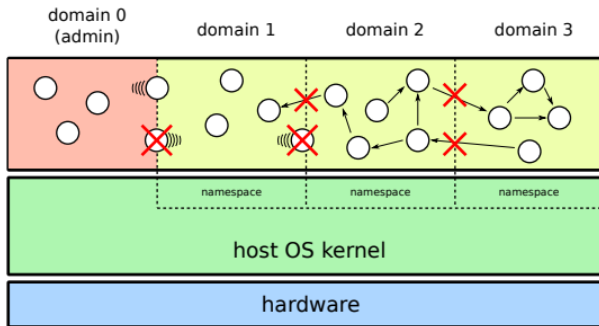


Figura 9: Domínios de containers (Maziero, 2019).

- › **Zonas** no Solaris: similar a jails, com controle de alocação de recursos.
- › **Virtuozzo** e **VServers** (Linux): isolamento leve.
- › **LXC** (Linux Containers): contêineres com namespaces e cgroups.
- › **Docker**: abstração moderna sobre LXC, com foco em portabilidade.

Emulação

- Virtualização de *software* e *hardware*.



Capítulo 11 (OLIVEIRA; CARISSIMI; TOSCANI, 2001)



Bibliografia



OLIVEIRA, R. S.; CARISSIMI, A. S.; TOSCANI, S. S. **Sistemas Operacionais**. 2. ed. Porto Alegre: Sagra-Luzzatto, 2001.



STALLINGS, William. **Operating Systems: Internals and Design Principles**. 6. ed. Upper Saddle River, NJ: Prentice-Hall, 2009.



TANENBAUM, Andrew S. **Sistemas Operacionais Modernos**. 3. ed. São Paulo: Pearson, 2010.



SILBERSCHATZ, Abraham; GALVIN, Peter; GAGNE, Greg. **Sistemas Operacionais: Conceitos e Aplicações**. Rio de Janeiro: LTC, 2001.

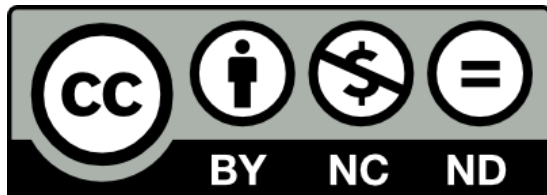


TANENBAUM, Andrew S.; WOODHULL, Albert S. **Sistemas Operacionais: Projeto e Implementação**. 2. ed. Porto Alegre: Bookman, 2000.



MAZIERO, Carlos Alberto. **Sistemas Operacionais: Conceitos e Mecanismos [recurso eletrônico]**. Curitiba: DINF - UFPR, 2019. ISBN 978-85-7335-340-2.

Estes slides estão protegidos por uma licença Creative Commons



Este modelo foi adaptado de Maxime Chupin.

Sistemas Operacionais

Virtualização