Virtualização

Tópicos de Informática para Automação

Mário Antunes

October 06, 2025

Universidade de Aveiro

Executar um Computador Dentro de Outro Computador

A Ideia Central: O que é a Virtualização?

A **Virtualização** cria uma versão baseada em software, ou "virtual", de um computador. Esta Máquina Virtual (VM) corre como uma aplicação no seu computador físico, mas comporta-se como uma máquina completamente separada.

- **Anfitrião (Host):** A sua máquina física e o seu Sistema Operativo (SO).
- **Convidado (Guest):** A máquina virtual e o SO que ela executa.
- **Hipervisor:** O software que cria e gere as VMs.

O Desafio: Gerir Instruções Privilegiadas

Uma aplicação normal não pode aceder diretamente ao hardware; tem de pedir ao SO Anfitrião. Mas um SO Convidado *espera* ter controlo total. Como resolvemos este conflito de forma segura?

O principal trabalho do hipervisor é intercetar e gerir de forma segura os pedidos do convidado para acesso privilegiado ao hardware. A forma como o faz define a diferença entre emulação e virtualização.

Emulação: Definição e Caso de Uso

Definição: A emulação envolve o uso de software para imitar o hardware de um sistema *diferente*. O hipervisor atua como um tradutor, convertendo as instruções da arquitetura da CPU do convidado para a arquitetura da CPU do anfitrião.

Exemplo de Caso de Uso: Executar um videojogo clássico concebido para uma consola baseada em ARM (como a Nintendo Switch) ou uma consola baseada em PowerPC (como a GameCube) no seu PC x86. O emulador (p. ex., Yuzu ou Dolphin) traduz o código do jogo em tempo real.

Emulação: O Caminho de uma Instrução

O hipervisor (emulador) tem de inspecionar e traduzir cada instrução em software antes que esta possa ser executada pelo hardware do anfitrião.

Emulação: Vantagens e Desvantagens

Vantagens 👍

- **Compatibilidade entre Arquiteturas:** É a sua maior força. Permite que software desenhado para um tipo de CPU (p. ex., ARM) corra num tipo completamente diferente (p. ex., x86).

Desvantagens 👎

- **Muito Lenta:** O passo de tradução de software para cada instrução cria uma sobrecarga de desempenho significativa, tornando-a muito mais lenta do que a execução de código nativo.
- **Elevado Uso de Recursos:** O processo de tradução em si é computacionalmente caro e consome muitos ciclos de CPU do anfitrião.

Virtualização Completa: Definição e Caso de Uso

Definição: A Virtualização Completa executa um SO convidado *não modificado* num ambiente de hardware simulado que corresponde à arquitetura do anfitrião. Baseia-se em **assistência de hardware da CPU** (Intel VT-x / AMD-V) para executar o código de forma eficiente. O SO convidado não tem consciência de que está a ser virtualizado.

Exemplo de Caso de Uso: Um utilizador de macOS a executar uma versão completa do Windows 11 no VirtualBox para usar um software específico que não está disponível para macOS, como um programa de CAD ou um jogo de PC específico.

Virtualização Completa: O Caminho de uma Instrução

As instruções não privilegiadas são executadas diretamente na CPU do anfitrião a toda a velocidade. Quando o convidado tenta executar uma instrução privilegiada, o hardware da CPU automaticamente a **interceta ("trap")** e entrega o controlo de forma transparente ao hipervisor para que este a trate de forma segura.

Virtualização Completa: Vantagens e Desvantagens

Vantagens 👍

- **Elevada Compatibilidade:** Pode executar qualquer sistema operativo padrão sem modificações.
- **Bom Desempenho:** A assistência de hardware torna-a significativamente mais rápida do que a emulação.
- **Forte Isolamento:** Os convidados estão isolados de forma segura do anfitrião e uns dos outros pelo hardware.

Desvantagens 👎

- **Sobrecarga do "Trap":** O ciclo "trap-and-emulate" para instruções privilegiadas ainda introduz alguma sobrecarga de desempenho, que pode ser significativa em cargas de trabalho intensivas em I/O (Entrada/Saída).

Paravirtualização: Definição e Caso de Uso

Definição: Na Paravirtualização, o SO convidado está *ciente* de que é uma VM e foi modificado com drivers especiais. Em vez de realizar ações que seriam intercetadas, comunica diretamente com o hipervisor através de uma API eficiente.

Exemplo de Caso de Uso: Esta é a base da computação em nuvem moderna. Um servidor web de alto desempenho a correr numa instância EC2 da Amazon Web Services (AWS) usa drivers paravirtualizados **VirtIO** para os seus dispositivos de disco e rede, para maximizar o débito e a baixa latência.

Paravirtualização: O Caminho de uma Instrução

O SO convidado sabe que não pode aceder diretamente ao hardware, por isso o seu driver "consciente" faz uma **"Hypercall"** — uma chamada de função direta e altamente eficiente ao hipervisor, evitando completamente o mecanismo de "trap".

Paravirtualização: Vantagens e Desvantagens

Vantagens 👍

- **Desempenho Mais Elevado:** Ao evitar a sobrecarga do "trap", oferece o melhor desempenho, especialmente para tarefas intensivas de disco e rede.
- **Eficiente:** Menor sobrecarga de CPU em comparação com a virtualização completa, porque o caminho de comunicação é otimizado.

Desvantagens 👎

- **Requer Modificação do SO Convidado:** Não pode executar um SO padrão não modificado. O SO precisa de ter os drivers de paravirtualização específicos instalados (embora a maioria das versões modernas de Linux e Windows já os inclua).

Resumo Comparativo

Característica	Emulação	Virtualização Completa	Paravirtualização
Conceito Central	Imitar hardware diferente	Isolar um SO não modificado	Cooperar com um SO consciente
Desempenho	Muito Baixo	Bom	Excelente
Modificaçãodo SO Convidado	Não	Não	**Sim**
Hardware	Qualquer convidado emqualquer anfitrião	Convidado e Anfitrião devempartilhar a mesma arquitetura	Convidado e Anfitrião devempartilhar a mesma arquitetura
MecanismoPrimário	Tradução por Software	“Trap & Emulate” porHardware	Hypercalls
Caso de UsoTípico	Jogos Retro, Desenvolvi-mento entre Arquiteturas	Uso em Desktop, Encapsu-lamento de Sistemas Legados	Computação em Nuvem, DataCenters, Servidores deAlto Desempenho

Caso de Uso 1: Data Centers e Servidores

A virtualização é a espinha dorsal da nuvem moderna.

- **Consolidação de Servidores:** Um único servidor físico potente pode substituir dezenas de servidores mais antigos, executando cada um como uma VM separada, poupando eletricidade, arrefecimento e espaço físico.
- **Snapshots e Alta Disponibilidade:** Guarde e restaure instantaneamente o estado de uma VM. As VMs podem até ser migradas entre servidores físicos sem tempo de inatividade.

O Problema: Configuração Repetitiva de VMs

Imagine que precisa de implementar 10 VMs de servidores web idênticas. O processo manual para *cada uma* seria:

1. Arrancar a VM e fazer login.
2. Definir um hostname único.
3. Configurar a rede.
4. Criar contas de utilizador e configurar chaves SSH.
5. Executar atualizações de segurança (`apt update && apt upgrade`).
6. Instalar o software necessário (`nginx`, `ufw`, etc.).
7. Configurar os serviços.

Isto é lento, entediante e propenso a erro humano. Simplesmente não escala para ambientes de nuvem.

A Solução: Cloud-Init ☁️

O **Cloud-Init** é a ferramenta padrão da indústria para automatizar a **configuração inicial** de uma instância na nuvem ou máquina virtual. Foi concebido para ser executado **apenas no primeiro arranque** para provisionar o sistema.

- **Como Funciona:**
  1. A plataforma de nuvem ou o hipervisor fornece dados de configuração (chamados “user data”) à VM no momento da sua criação.
  2. Dentro do SO convidado, um serviço Cloud-Init arranca automaticamente no primeiro boot.
  3. Este serviço encontra os “user data” e executa as instruções contidas neles para configurar o sistema.
- **Analogia:** Pense no Cloud-Init como um script de configuração automatizado que prepara o seu novo servidor antes mesmo de você fazer o primeiro login.

Cloud-Init na Prática: User Data

A configuração para o Cloud-Init é tipicamente escrita num formato de texto simples chamado **YAML**. Este ficheiro, muitas vezes chamado `user-data`, contém um conjunto de diretivas. Com este único ficheiro, uma nova VM pode arrancar como um servidor web totalmente configurado e pronto a usar, sem qualquer intervenção manual.

Aqui está um exemplo prático de um ficheiro `user-data` que configura um servidor web básico:

```yaml
#cloud-config
# Definir o hostname para o servidor
hostname: webserver-01

# Criar um novo utilizador chamado 'admin', dar-lhe direitos de sudo e adicionar uma chave S
users:
  - name: admin
    sudo: ALL=(ALL) NOPASSWD:ALL
    groups: sudo
    shell: /bin/bash
    ssh_authorized_keys:
      - ssh-rsa AAAA ... user@example.com

# Instalar os pacotes do servidor web nginx e da firewall
packages:
  - nginx
  - ufw

# Executar comandos após a instalação dos pacotes para configurar e iniciar os serviços
runcmd:
  - [ ufw, allow, 'WWW Full' ]
  - [ systemctl, enable, --now, nginx ]
```

O Desafio do I/O e a Solução de Alto Desempenho

Uma VM não tem hardware físico. O hipervisor tem de fornecer dispositivos virtuais.

- **Emulação (Lenta):** O hipervisor pode fingir ser um dispositivo de hardware comum e real (como uma placa de rede Intel E1000). Isto é compatível, mas lento.
- **Paravirtualização (Rápida):** Os sistemas modernos usam **VirtIO**. O SO convidado tem um driver `virtio` especial que não emula hardware real, mas usa um canal padronizado e altamente eficiente para comunicar com o hipervisor para tarefas de disco e rede.

Rede Virtual: Modo NAT vs. Modo Bridge

- **Modo NAT (Padrão):** A VM partilha o endereço IP do seu anfitrião. É fácil de configurar e permite que o convidado aceda à internet, mas torna difícil que outros dispositivos na sua rede se conectem ao convidado.
- **Modo Bridge:** A VM obtém o seu próprio endereço IP na sua rede local, aparecendo como um dispositivo físico separado. Isto é ideal para executar servidores.

Acesso a Dispositivos: Passthrough de USB e PCI

Pode conceder a uma VM o controlo exclusivo de um dispositivo físico conectado ao seu anfitrião.

- **Passthrough de USB:** Dá a uma VM acesso direto a um dispositivo USB. Isto é essencial para o desenvolvimento de sistemas embebidos, permitindo que a sua VM Debian programe diretamente uma placa **Arduino ou ESP32**.
- **Passthrough de PCI:** Atribui um dispositivo PCI físico, como uma potente **GPU**, diretamente a uma VM. Isto oferece desempenho quase nativo para tarefas exigentes como jogos ou machine learning.

Como o Passthrough de PCI é Alcançado

Esta funcionalidade avançada requer suporte de hardware da CPU e do chipset da motherboard, especificamente da **IOMMU (Input-Output Memory Management Unit)**.

- **IOMMU da Intel:** VT-d
- **IOMMU da AMD:** AMD-Vi

A IOMMU cria um "sandbox" de memória seguro para o dispositivo, garantindo que este apenas pode aceder à memória da VM à qual está atribuído. Isto impede que o dispositivo interfira com o SO anfitrião ou outras VMs.

Apresentando: Oracle VirtualBox

O VirtualBox é um hipervisor **Tipo-2 (hospedado)** que corre como uma aplicação padrão no seu SO existente. É desenvolvido pela Oracle e é gratuito e de código aberto.

- **Para quem é:** Iniciantes, estudantes e utilizadores de desktop que precisam de uma interface gráfica fácil de usar para executar VMs.
- **Principais Características:**
  - Multi-plataforma (Windows, macOS, Linux).
  - Interface gráfica amigável.
  - "Guest Additions" para integração perfeita do rato, pastas partilhadas e área de transferência.
  - Funcionalidade de snapshots fácil de usar.

Instalar o VirtualBox

O processo envolve a instalação da aplicação principal e de um "Extension Pack" separado para funcionalidades completas (como suporte a USB 2.0/3.0).

1. **Download:** Vá à página oficial de downloads do VirtualBox e descarregue o pacote para o seu SO anfitrião. Descarregue também o **Extension Pack**.
2. **Instalar a Aplicação:** Execute o instalador da aplicação principal.
3. **Segurança no macOS:** No macOS, tem de ir a **Definições do Sistema > Privacidade e Segurança** para **Permitir** a extensão de sistema da Oracle.
4. **Instalar o Extension Pack:** Dê um duplo clique no ficheiro `.vbox-extpack` descarregado. O VirtualBox abrirá e guiá-lo-á na instalação.

Usar o VirtualBox

Criar e executar uma VM é um processo simples, guiado por um assistente.

1. Clique no botão **“Novo”** para iniciar o assistente de nova VM.
2. Atribua um nome, tipo de SO e RAM.
3. Quando solicitado um disco rígido, pode criar um novo ou, para a sua aula, escolher **“Não adicionar um disco rígido virtual”**.
4. Vá às **Definições > Armazenamento** da VM e clique no ícone “Adicionar Disco Rígido” para anexar o seu ficheiro `.vdi` fornecido.
5. Reveja outras definições como **Rede** (`NAT` ou `Bridge`) e **USB** (para ativar o passthrough).
6. Selecione a VM e clique em **“Iniciar”**.

Apresentando: QEMU + KVM

O QEMU é um emulador de máquinas potente, e o KVM (Kernel-based Virtual Machine) é o módulo de virtualização integrado do kernel Linux. Juntos, eles fornecem virtualização **Tipo-1 (bare-metal)** de alto desempenho em Linux.

- **Para quem é:** Administradores de sistemas, programadores e utilizadores avançados que precisam de flexibilidade, desempenho e controlo por linha de comandos. É o motor por trás de muitas plataformas de nuvem em larga escala.
- **Principais Características:**
  - Extremamente flexível e programável (scriptable).
  - Pode emular uma enorme variedade de arquiteturas de CPU (ARM, MIPS, etc.).
  - Desempenho quase nativo quando usado com KVM.

Instalar o QEMU + KVM

Em sistemas baseados em Debian/Ubuntu, a instalação é feita através do gestor de pacotes `apt`.

1. **Instalar Pacotes:** Abra um terminal e execute o seguinte comando. O `virt-manager` é uma ferramenta gráfica altamente recomendada para gerir VMs QEMU/KVM.

```
$ sudo apt update
$ sudo apt install qemu-system-x86 kvm virt-manager libvirt-daemon-system
```

2. **Adicionar Utilizador a Grupos:** Adicione o seu utilizador aos grupos `libvirt` e `kvm` para gerir VMs sem precisar de `sudo` para cada ação. Terá de fazer logout e login novamente para que isto tenha efeito.

```
$ sudo adduser $USER libvirt
$ sudo adduser $USER kvm
```

Usar o QEMU + KVM

Embora o `virt-manager` forneça uma GUI, usar o QEMU a partir da linha de comandos demonstra o seu poder.

1. **Criar um Disco Virtual:** O formato `.qcow2` é recomendado. Este comando cria um disco de 20GB que só cresce à medida que adiciona dados.

```
$ qemu-img create -f qcow2 o_meu_disco_debian.qcow2 20G
```

2. **Lançar uma VM:** Este comando inicia uma VM para instalar um SO a partir de um ficheiro ISO.

```
$ qemu-system-x86_64 -enable-kvm -m 2048 -hda o_meu_disco_debian.qcow2 \
-cdrom debian-13-netinst.iso -boot d
```

   - `-enable-kvm`: Usar KVM para aceleração de hardware (crítico para o desempenho).
   - `-m 2048`: Atribuir 2048 MB de RAM.
   - `-hda`: Especificar o ficheiro do disco rígido primário.
   - `-cdrom`: Anexar um ficheiro ISO como um CD-ROM virtual.
   - `-boot d`: Dizer à VM para arrancar primeiro a partir da unidade de CD-ROM.

Comparação: VirtualBox vs. QEMU/KVM

Característica	VirtualBox	QEMU + KVM
Tipo	Tipo-2 (Hospedado)	Tipo-1 (Bare-Metal, via Kernel Linux)
Plataforma Primária	Multi-Plataforma (Windows, macOS, Linux)	Linux
Facilidade de Uso	**Muito Elevada** (Gráfica, guiada por assistente)	**Média a Baixa** (Focada na linha de comandos)
Desempenho	Bom (Excelente para uso em desktop)	**Excelente** (Velocidade quase nativa)
Flexibilidade	Boa (Opções amigáveis)	**Muito Elevada** (Altamente configurável e programável)
Melhor Para...	Estudantes, utilizadores de desktop, configurações rápidas	Servidores, programadores, soluções personalizadas, emulação

Suporte e Mais Recursos 📚

Guarde estas páginas para referência rápida.

- **Virtual Box:**
  - Manual
  - Rede
- **QEMU:**
  - Manual
  - Rede