Virtualização

a. Emulação
b. Máquinas Virtuais
c. Hypervisors (VMM)
d. Técnicas de Virtualização
e. Soluções
f. Conteineirização

Emulação

- Na computação, um emulador é um software que reproduz as funções de um determinado ambiente, a fim de permitir a execução de outros softwares sobre ele. Pode ser pela transcrição de instruções de um processador alvo para o processador no qual ele está rodando, ou pela interpretação de chamadas para simular o comportamento de um hardware específico. O emulador também é responsável pela simulação dos circuitos integrados ou chips do sistema de hardware em um software.

Technical Information

CPU	• Motorola 68000 (primary CPU) • Zilog Z80 (secondary CPU)
Sound	**Mega Drive/Genesis Model 1** • Yamaha YM2612 with 6 audio channels • Texas Instruments SN76489 with 3 square wave tone generators and 1 noise generator • Mono (Stereo with headphone jack) **Mega Drive/Genesis**
Graphics	512 color palette, 61 colors on-screen
Display	• Progressive: 320x224 (NTSC) or 320x240 (PAL) pixels • Interlaced: 320x448 (NTSC) or 320x480 (PAL)

SEGA Genesis

Technical Specifications

```
- CPU: 65c816 based, named Ricoh 5A22, 16-bit
- PPUs: Ricoh 5c77 (PPU1) and Ricoh 5C78 (PPU2)
- Sound channels: 8, uses compressed wave samples
- APU: Sony 8-bit SPC700 processor at 1.024 MHz
- S-DSP: Outputs at 32 KHz
- DRAM: 128 KBs at 2.58 MHz (?)
- VRAM: 32 KBs two chips
- OAM: 544 Bs
- ARAM: 64 KBs
- WRAM: 1 Mbit (128 Kbyte)
- DMA: 8 channels, H-DMA (Horizontal DMA) also featured, using same channels
- Resolution: 256x224 512 x 448 pixels max hi res and interlaced modes
- Colors Available: 32,768 colors
- Max sprite size: 64 x 64 pixels
- Max sprites: 128
- Min/Max Cart Size: 2 Mbit - 48 Mbit
- Other: 16-bit color, line based interrupts (IRQ and NMI)
```

Super Nintendo Entertainment System

Virtualização - Definição

- Virtualização em Sistemas Computacionais
- Técnica que permite dividir um sistema computacional real (hospedeiro) em diversas máquinas virtuais (hóspedes) isoladas.
- Conceito publicado pela 1ª. Vez por Christopher Strachey, em 1959, e implementado pela IBM na década de 1960, no modelo IBM 7044 e, logo após, no IBM System/360

Definição

- Outros exemplos:
  - Java Virtual Machine (JVM)
  - Memória Virtual
  - Virtualização de Storages
  - Virtualização de Desktops
  - Virtualização em Celulares (Computation Offloading)

Virtualização

- Mainframes:
  - Alto custo
  - Alto poder computacional
  - Difícil acesso

- Solução:
  - Dividir os recursos físicos criando diversas partições lógicas isoladas entre si, permitindo que cada uma possua seu próprio sistema com distintas plataformas
  - Desta forma, os mainframes poderiam executar múltiplos Sistemas Operacionais simultaneamente sem necessidade de alterar os sistemas legados existentes

IBM SYSTEM/360

Virtualização

- Computadores pessoais:
  - Médio custo
  - Baixo poder computacional
  - Fácil acesso

- Solução:
  - Segurança
  - Isolamento
  - Ensino / Aprendizagem
  - Teste de Aplicações
  - Aplicações Legadas

Virtualização

- Servidores com n-núcleos:
  - Baixo custo
  - Alto poder computacional
  - Fácil acesso

- Solução:
  - Maximizar o uso dos recursos
  - Promover compartilhamento de recursos
  - Isolamento (Segurança)
  - Desempenho
  - Transparência

Virtualização - Hypervisor

- Gerenciador de Máquinas Virtuais (Virtual Machine Manager – VMM)

- Camada de software responsável por:
  - Fornecer para cada máquina virtual (VM) uma cópia virtual (abstração) dos recursos físicos do sistema hospedeiro
  - Garantir que várias VMs possam ser executadas simultaneamente sobre um mesmo conjunto de hardware

Virtualização - Recursos

- Processador
  - As instruções despachadas dentro da VM serão executadas diretamente pela CPU real, exceto se forem instruções privilegiadas ou instruções sensíveis
- Memória
  - A tabela de páginas da VM mapeia as páginas físicas do sistema real sendo que o hypervisor faz uma cópia (shadow) desta tabela para controle
- Disco
  - Para acesso ao disco pode ser oferecida uma abstração (um arquivo no sistema real) ou também pode ser oferecida uma partição do disco do sistema real para a VM
- Rede
  - A interface de rede real trabalha em modo promíscuo, de forma a escutar o tráfego destinado a qualquer interface virtual e entregando os pacotes através de uma ponte (bridge)

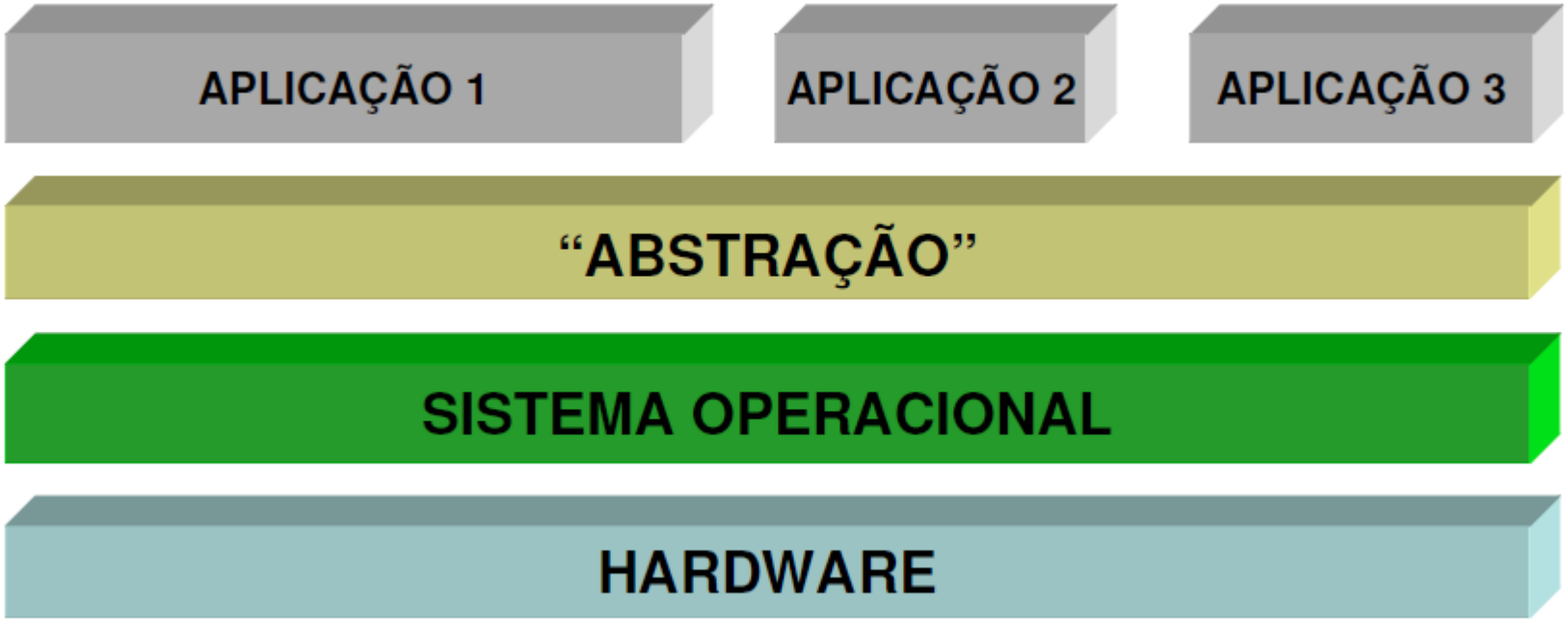
Virtualização

Técnicas de Virtualização

- Virtualização Total ou Completa
- Para-virtualização
- Virtualização Hospedada (Assistida por Hardware)

Técnicas de Virtualização

Sistema sem virtualização:

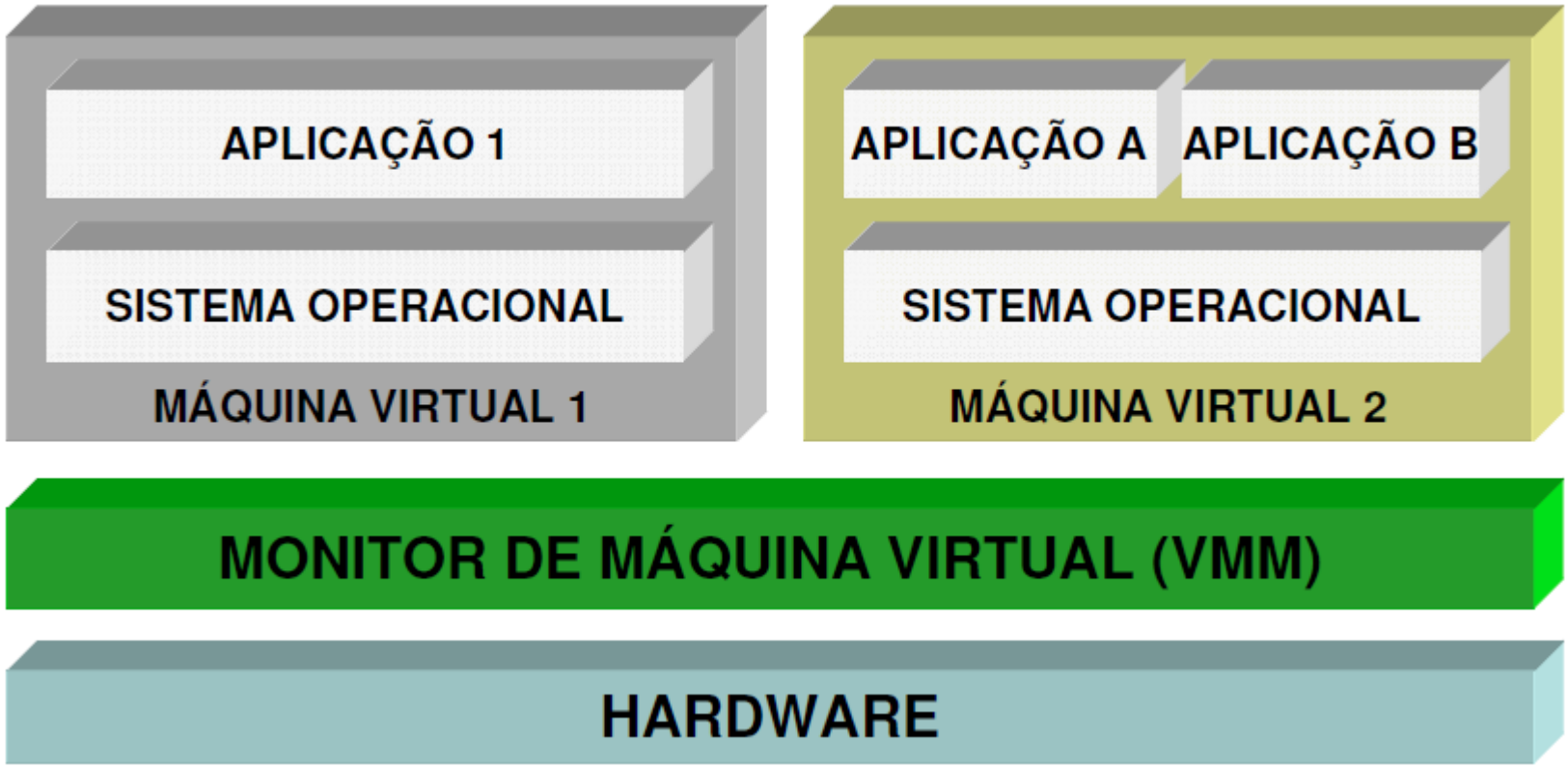


Técnicas de Virtualização

- Virtualização Total ou Completa:
  - O hardware é totalmente virtualizado sendo disponibilizada uma abstração do mesmo para as VMs, gerando independência (portabilidade)
  - Não requer modificações no kernel do S.O. das VMs
- Desvantagens:
  - O hypervisor pode suprimir algumas características do hardware real ao prover uma abstração genérica
  - O hypervisor deve inspecionar as instruções executadas pelas VMs buscando chamadas de instruções sensíveis (gerando queda de desempenho)

Técnicas de Virtualização

Sistema com virtualização total:

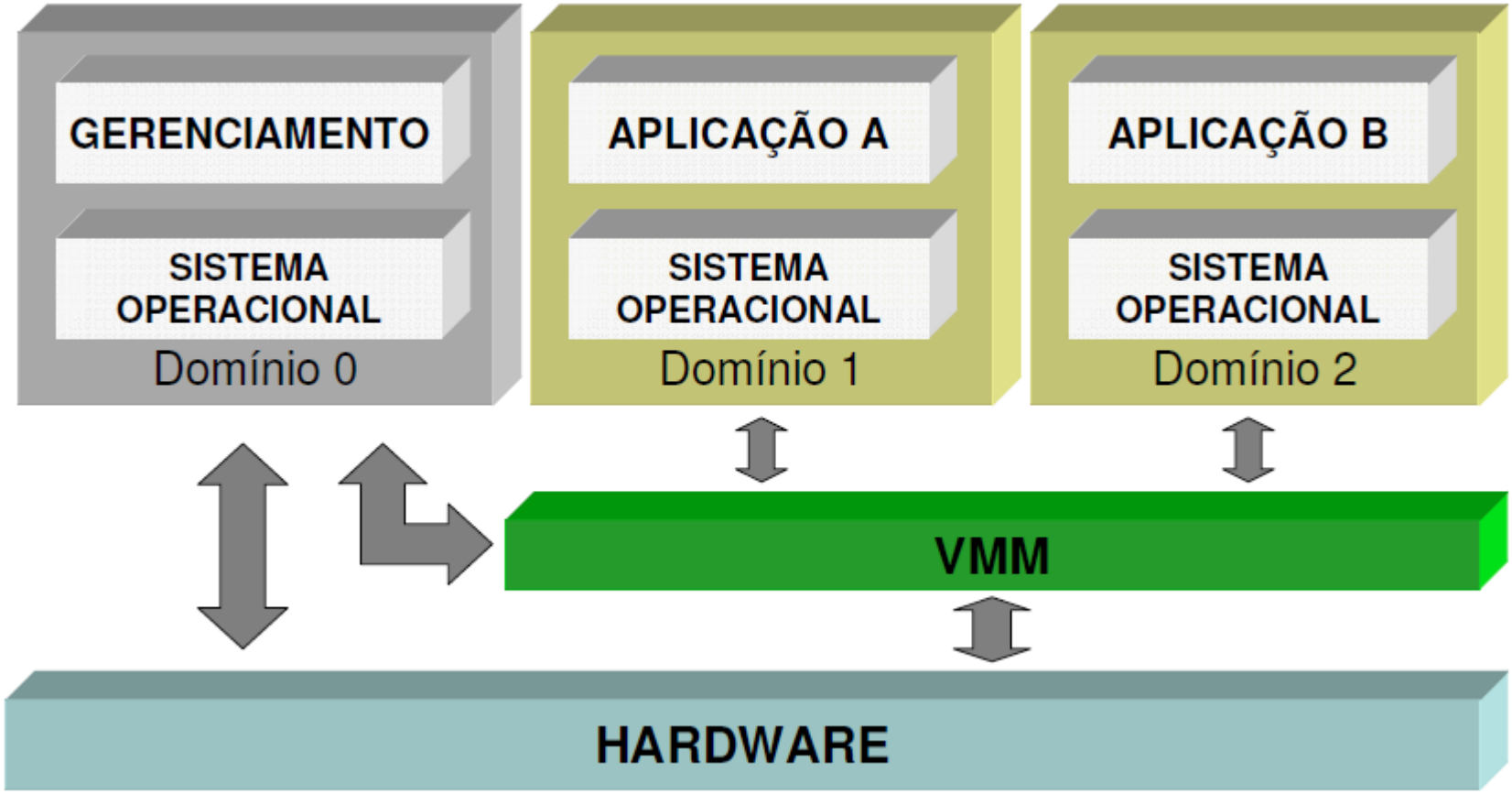


Técnicas de Virtualização

- Para-virtualização:
  - Não requer inspeção das instruções chamadas pelas VMs pois as chamadas de instruções sensíveis são desviadas para o hypervisor (*hypercalls*)
  - As VMs utilizam drivers do próprio hypervisor para acesso aos dispositivos
  - Apresenta melhor desempenho em relação à Virtualização Total
- Desvantagens:
  - Requer modificação no núcleo do sistema das VMs para inserção das hypercalls (nem sempre possível)
  - Os sistemas que são executados dentro das VMs passam a ter conhecimento do hypervisor

Técnicas de Virtualização

Sistema com para-virtualização:



Técnicas de Virtualização

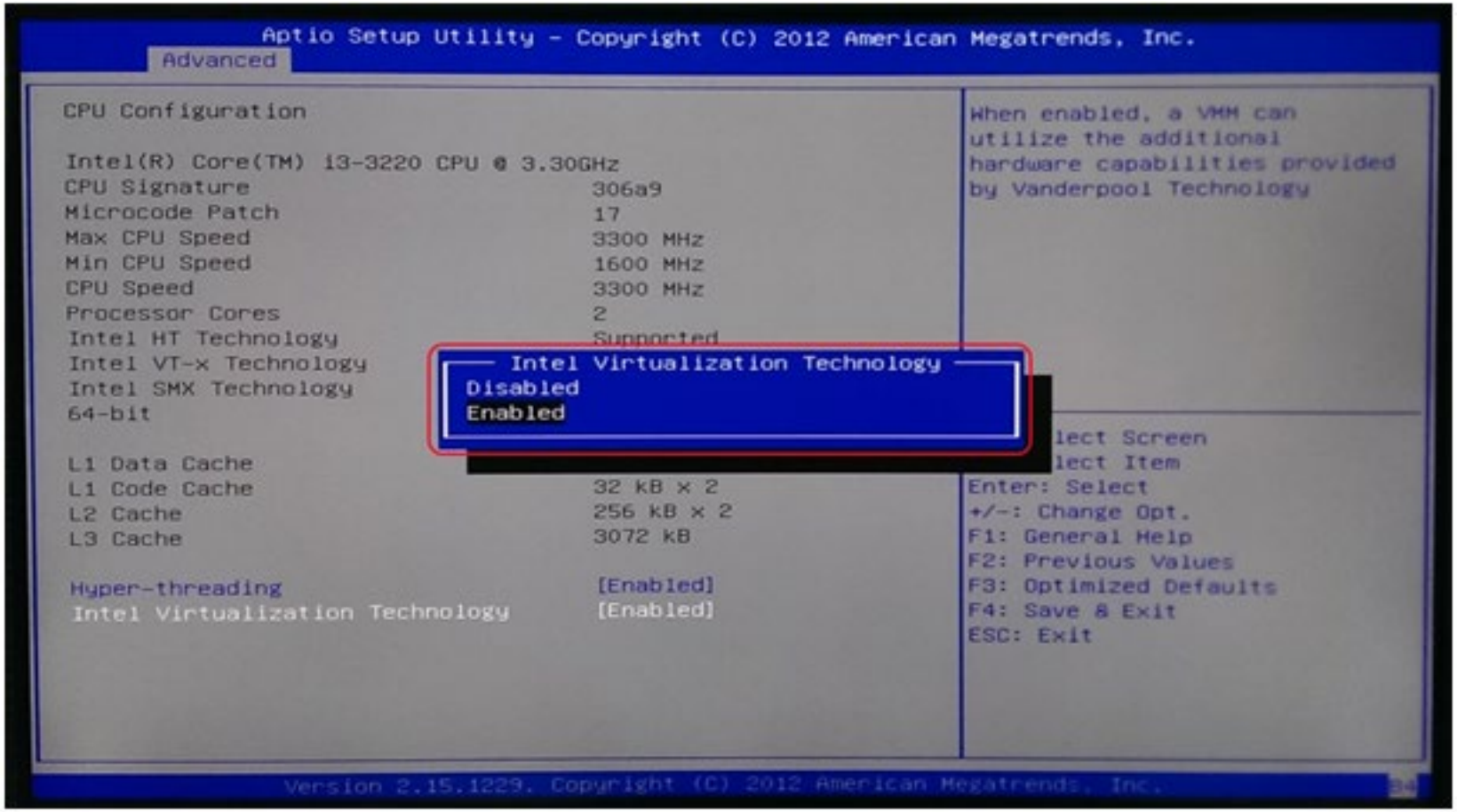
Técnicas de Virtualização

- Virtualização Assistida por Hardware:
  - O desempenho dos sistemas de virtualização no IBM System/360 ficou abaixo do esperado, então a IBM resolveu desenvolver um mainframe com arquitetura específica para suportar a virtualização no hardware (IBM System/370)
  - Intel e AMD passaram a prover suporte, no hardware, para a virtualização de forma a contornar alguns dos problemas anteriores
  - A Intel disponibilizou extensões (Intel VT ou Vanderpool) para a arquitetura x86 complementando o esquema de proteção com a inserção dos modos de operação root e não-root
  - A AMD disponibilizou funções (AMD-V ou Pacífica) no processador para auxiliar no controle do acesso das VMs, auxiliando o hypervisor.

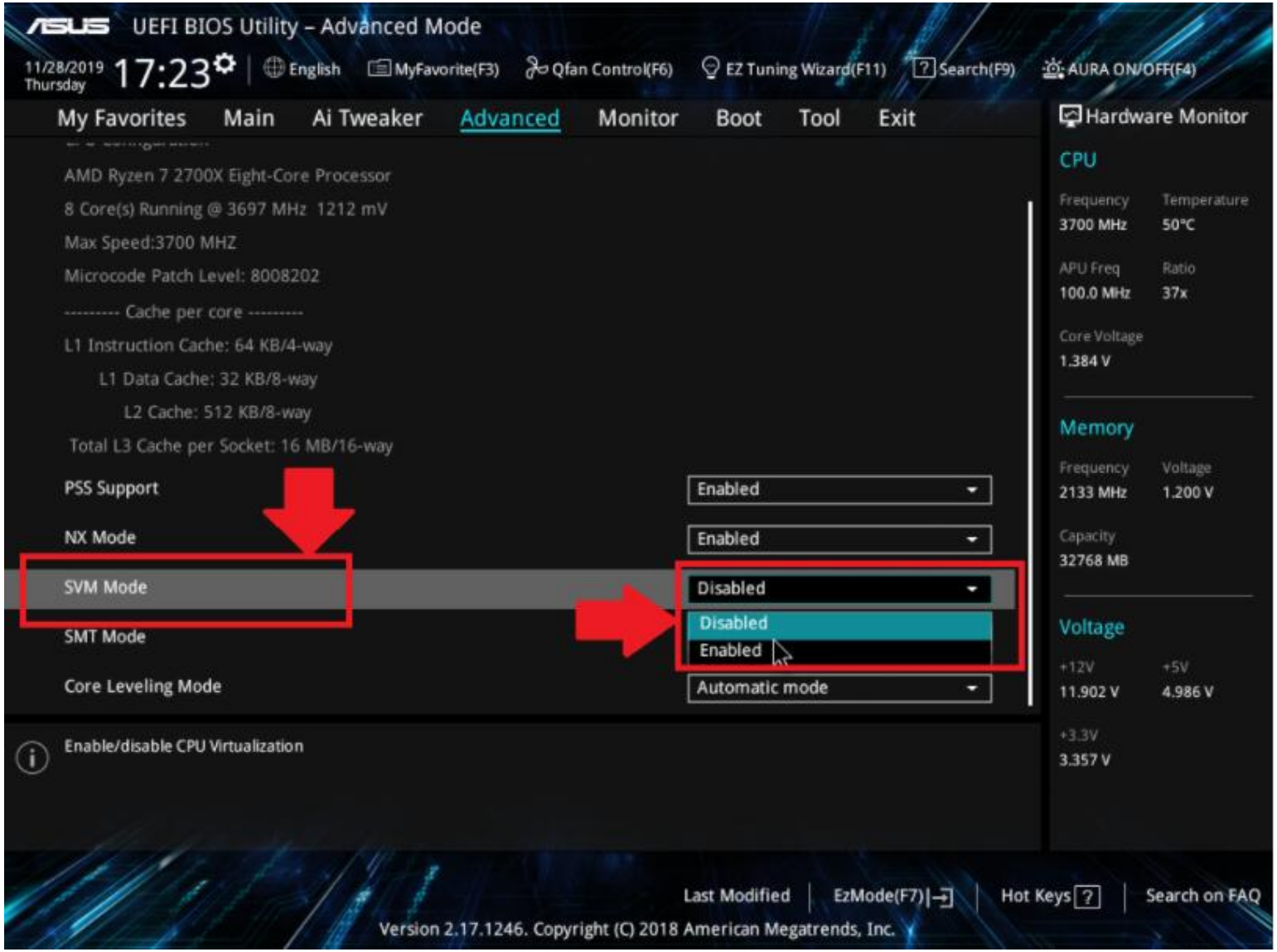
VT-x / AMD-V

- A Intel VT-x (Intel Virtualization Technology for IA-32) é uma tecnologia que se propõe a diminuir a complexidade dos Monitores de Máquina Virtual (VMM), aumentar o desempenho de sistemas virtualizados baseados em software e permitir que sistema operacionais não modificados sejam executados, em máquinas virtuais, com desempenho igual ou superior ao alcançado pela paravirtualização, na qual os sistemas operacionais são modificados ou sofrem translações binárias para serem executados sobre um VMM.
- A virtualização assistida por hardware, caso da Intel VT-x, muda a forma de acesso ao sistema operacional. Sistemas operacionais para a plataforma IA-32 são projetados para terem acesso direto aos recursos do sistema para executarem. Na virtualização por software, o VMM emula o hardware requerido pelo SO visitante. Na virtualização por hardware, o SO tem acesso direto aos recursos, sem emulação ou qualquer modificação no SO visitante.
- As extensões de virtualização para os processadores oferecem novas instruções para controlar a virtualização. Vale lembrar que a arquitetura IA-32 fornece diferentes níveis de acesso a recursos, os chamados *rings*. O nível de maior privilégio é o *ring* 0 e, também, é esse o nível de privilégio que acessa diretamente o hardware do sistema [3].
- Na arquitetura IA-32, o núcleo do sistema operacional espera ter acesso direto à CPU executando no nível 0. Com a virtualização por software isso não é possível, pois o VMM já está sendo executado no nível 0, sendo assim, o sistema operacional visitante deve ser executado no nível 1 ou nível 3. No entanto, algumas instruções só funcionam no nível 0, então uma possível solução é recompilar o sistema operacional visitante, para evitar essas instruções, essa é a chamada paravirtualização, que muitas vezes não é prática, pois o código do SO nem sempre está disponível. Para evitar esse problema, o VMM captura as instruções que podem vir a falhar na execução e emula a sua execução, o que resulta numa perda significativa de desempenho.

VT-x / AMD-V

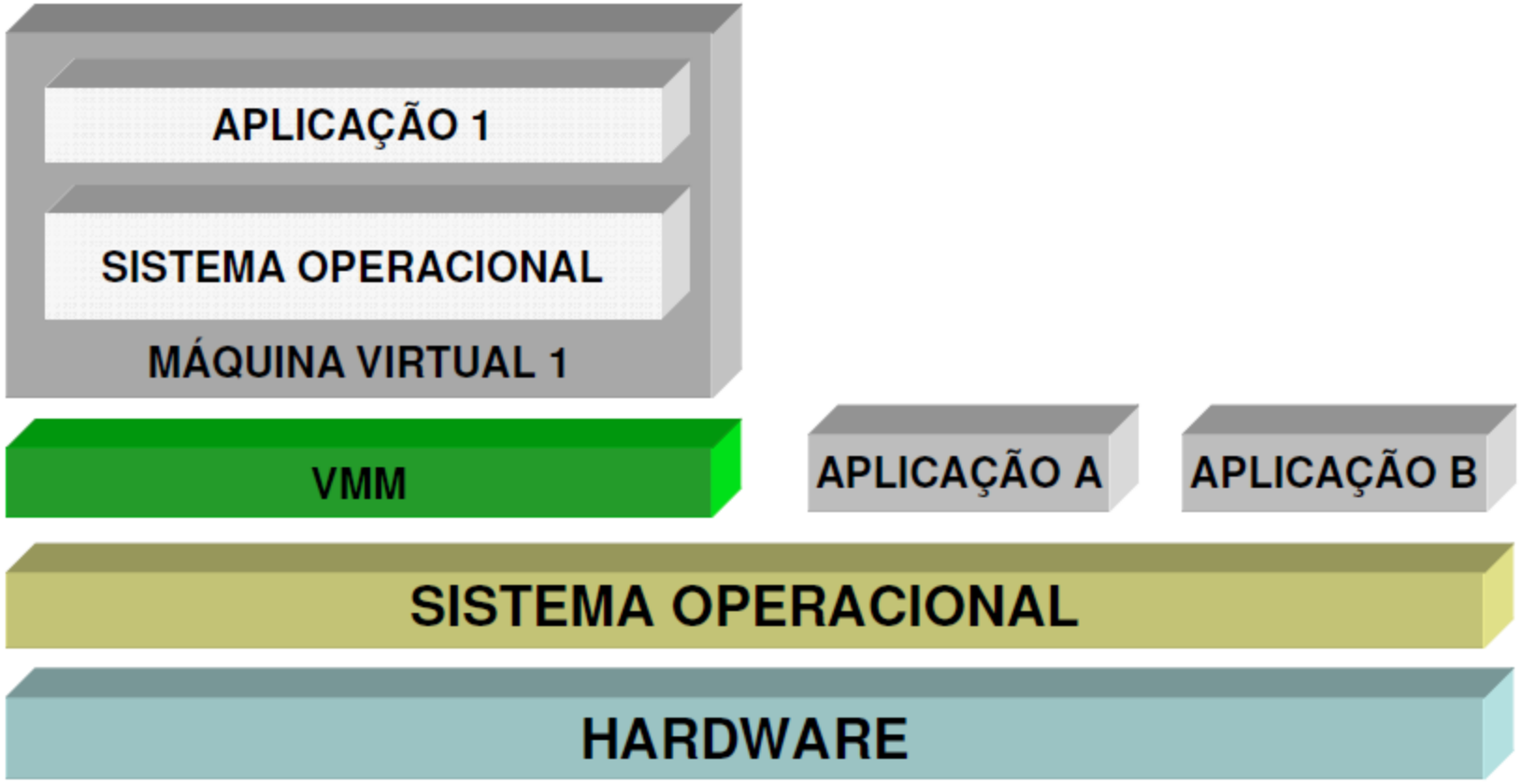

Aptio Setup Utility - Copyright (C) 2012 American Megatrends, Inc.
Advanced
CPU Configuration
Intel(R) Core(TM) i3-3220 CPU @ 3.30GHz
CPU Signature 306a9
Microcode Patch 17
Max CPU Speed 3300 MHz
Min CPU Speed 1600 MHz
CPU Speed 3300 MHz
Processor Cores 2
Intel HT Technology Supported
Intel VT-x Technology
Intel SMX Technology
64-bit
Intel Virtualization Technology
Disabled
Enabled
L1 Data Cache
L1 Code Cache 32 kB x 2
L2 Cache 256 kB x 2
L3 Cache 3072 kB
Hyper-threading [Enabled]
Intel Virtualization Technology [Enabled]
When enabled, a VMM can utilize the additional hardware capabilities provided by Vanderpool Technology
Enter: Select
+/-: Change Opt.
F1: General Help
F2: Previous Values
F3: Optimized Defaults
F4: Save & Exit
ESC: Exit
Version 2.15.1229. Copyright (C) 2012 American Megatrends, Inc.

VT-x / AMD-V


UEFI BIOS Utility – Advanced Mode
11/28/2019 Thursday 17:23
English
MyFavorite(F3)
Qfan Control(F6)
EZ Tuning Wizard(F11)
Search(F9)
AURA ON/OFF(F4)
My Favorites
Main
Ai Tweaker
Advanced
Monitor
Boot
Tool
Exit
AMD Ryzen 7 2700X Eight-Core Processor
8 Core(s) Running @ 3697 MHz 1212 mV
Max Speed:3700 MHZ
Microcode Patch Level: 8008202
--------- Cache per core ---------
L1 Instruction Cache: 64 KB/4-way
L1 Data Cache: 32 KB/8-way
L2 Cache: 512 KB/8-way
Total L3 Cache per Socket: 16 MB/16-way
PSS Support
Enabled
NX Mode
Enabled
SVM Mode
Disabled
Disabled
Enabled
SMT Mode
Core Leveling Mode
Automatic mode
Enable/disable CPU Virtualization
Hardware Monitor
CPU
Frequency
3700 MHz
Temperature
50°C
APU Freq
100.0 MHz
Ratio
37x
Core Voltage
1.384 V
Memory
Frequency
2133 MHz
Voltage
1.200 V
Capacity
32768 MB
Voltage
+12V
11.902 V
+5V
4.986 V
+3.3V
3.357 V
Last Modified
EzMode(F7)
Hot Keys
Search on FAQ
Version 2.17.1246. Copyright (C) 2018 American Megatrends, Inc.

Técnicas de Virtualização

Sistema com virtualização hospedada:



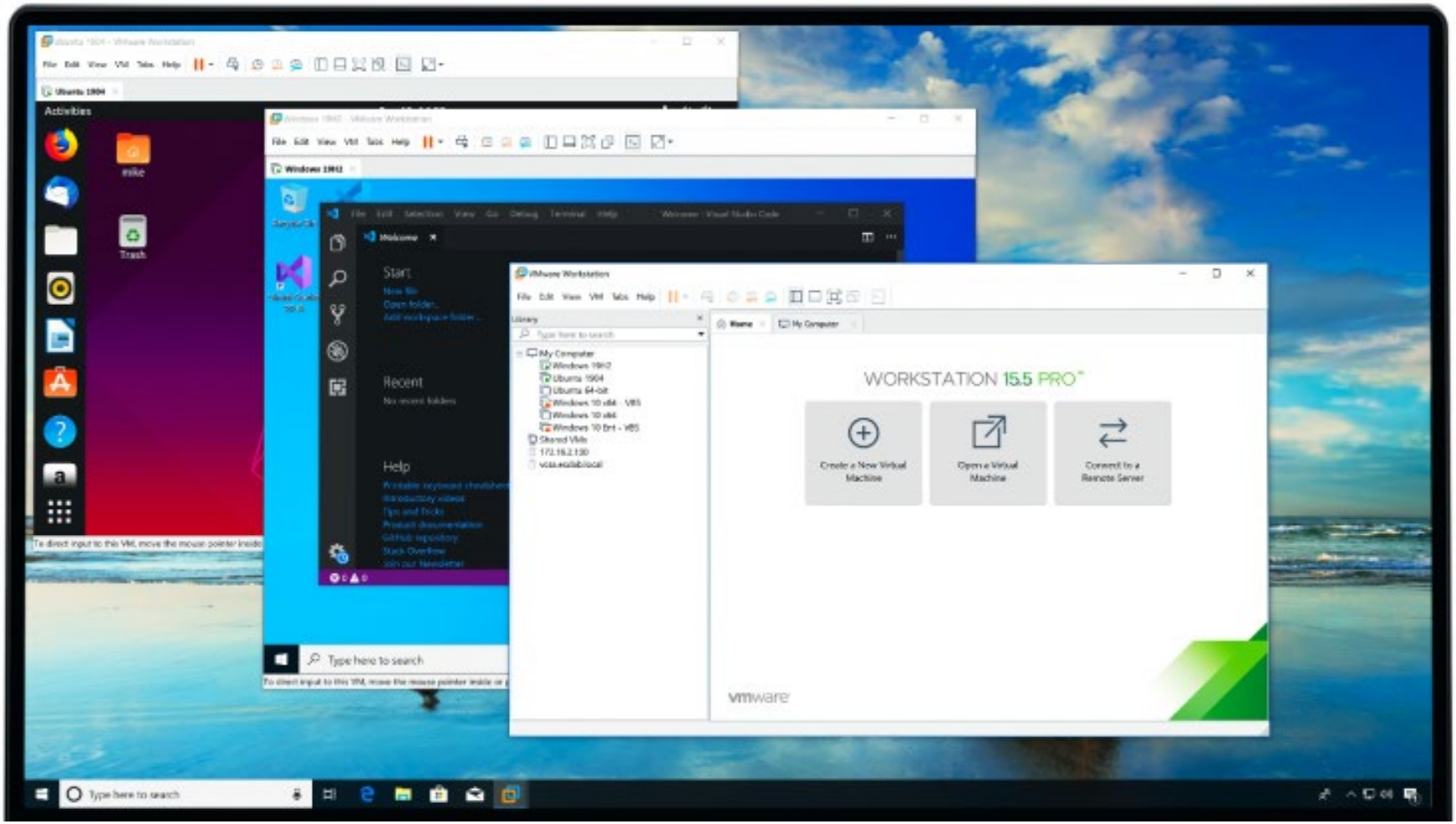
Aplicações

- Execução de aplicações legadas
- Desenvolvimento de sistemas multiplataforma ou distribuídos
- Treinamento (Linux, por exemplo)
- Gerência centralizada de servidores
- Teste e análise de aplicações (malwares por ex.)
- Contingência ou manutenção do hardware sem parada
- Consolidação de servidores (diminuição de espaço e gastos com energia e refrigeração)

Soluções - VMware

- Empresa fundada em 1998, adquirida pela EMC, posteriormente pela Dell e em 2022 passa a ser parte da Broadcom
- Produtos:
  - VMware ESXi
  - vSphere
  - VMware Server
  - VMware Player
  - Vmware Workstation
- Utiliza modelos de virtualização total, para-virtualização e virtualização hospedada
- Suporta Sistemas Operacionais como Windows e Linux (hóspedes ou hospedeiros)
- Ferramentas para migração de backup de VMs
- Soluções de alta disponibilidade (HA) e balanceamento de carga
- Instalação simples por não ter necessidade de alteração do kernel do S.O.

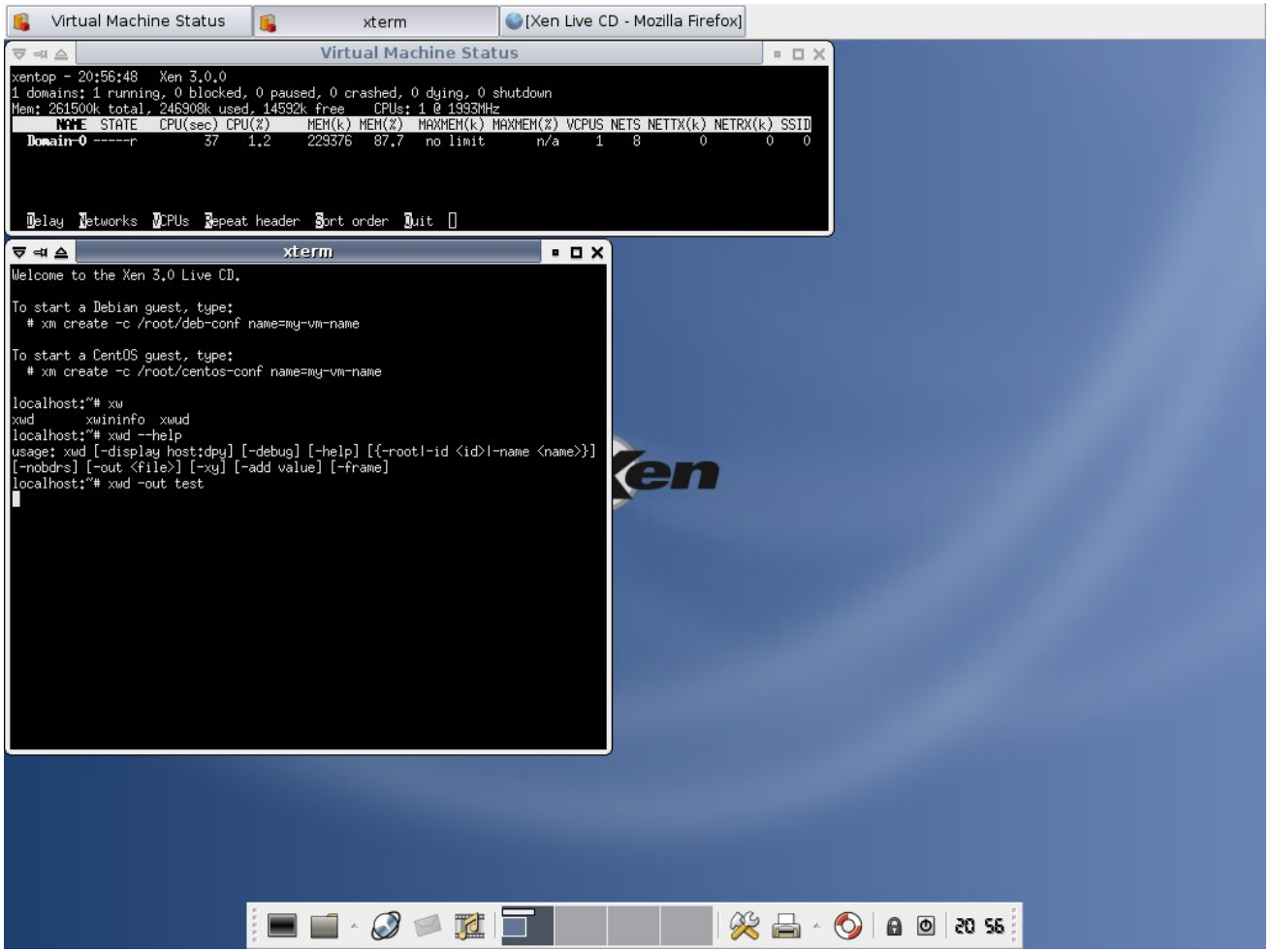
Soluções - VMware



Soluções - Xen

- Solução open source desenvolvida como parte do projeto XenoServers na Universidade de Cambridge (2003)
- Criação da XenSource, empresa adquirida pela Citrix Systems em 2007
- Existem versões comerciais (XenServer até versão 7 – Cytrix Hypervisor a partir da versão 8)
- Utiliza modelo de para-virtualização utilizando um kernel Linux modificado como hospedeiro
- Suporta Windows, Linux, Solaris, dentre outros como hóspedes
- Para suporte do Windows, é necessário suporte de hardware (VT-x ou AMD-V)
- A instalação pode não ser tão simples como o Vmware, já que implica na instalação de um novo kernel
- Existem pacotes prontos disponibilizados para diversas distros (RedHat, Debian, Open SUSE, etc.)

Soluções - XenServer



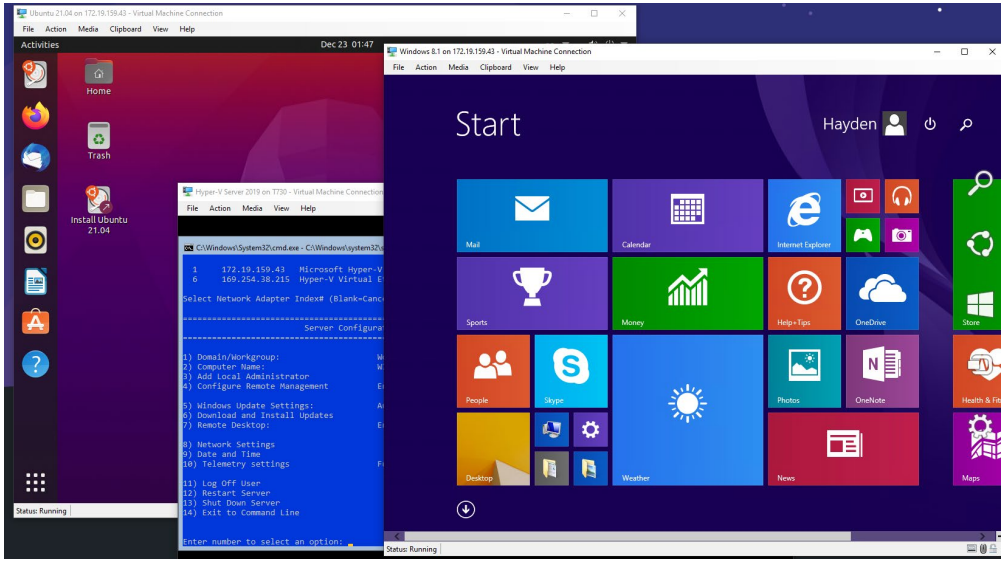
Soluções - Microsoft

- Virtual PC:
  - Uma das primeiras soluções da Microsoft qie permite a criação de instalações virtuais de Windows dentro de estações de trabalho (Mac, inclusive)
- Virtual Server:
  - Solução para uso em servidores. Era executado sobre um Sistema Operacional Windows 2000 ou 2003
- Hyper-V:
  - Solução mais recente que consiste basicamente em um hypervisor e pelo menos uma partição raiz executando Windows Server

Soluções

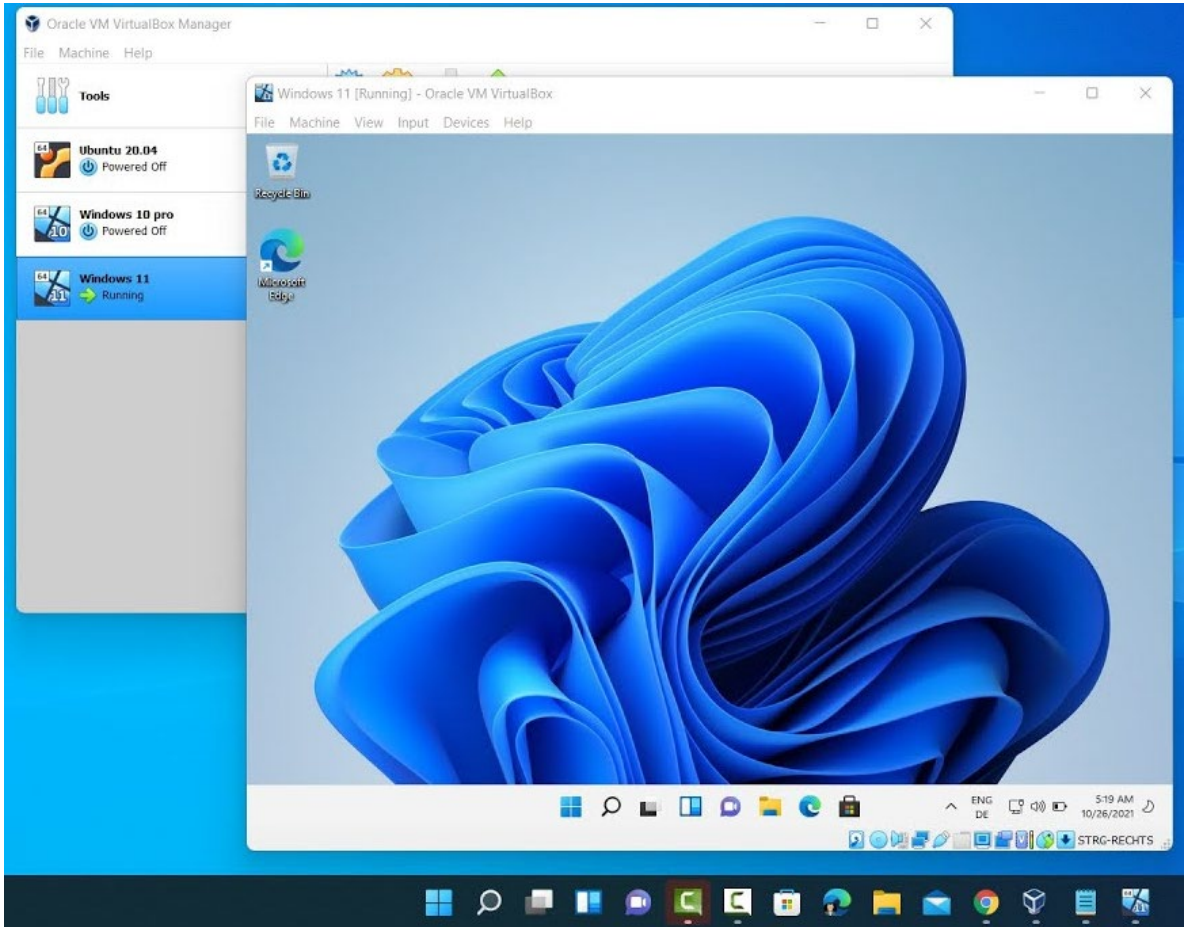
Virtual PC

Hyper-V



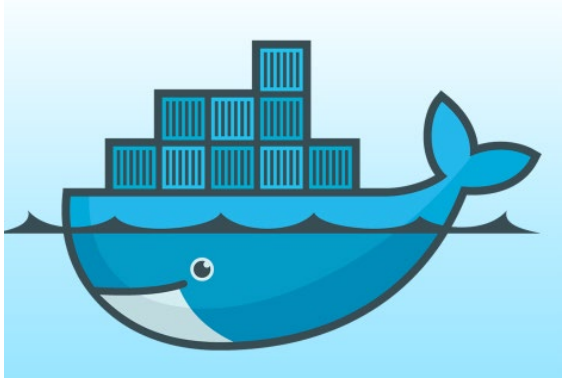
Outras Soluções

- Oracle VirtualBox
- KVM
- OpenVZ
- IBM Lpar
- Linux VServer



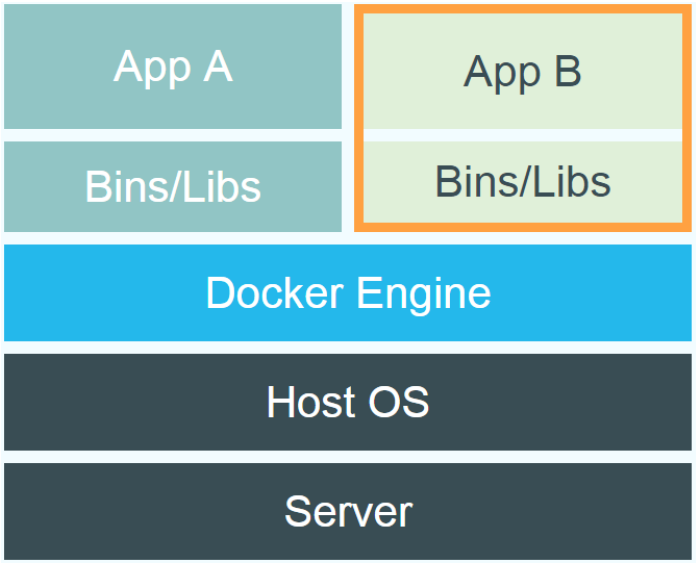
Conteinerização - Docker

- A palavra "Docker" tem várias definições: um projeto da comunidade open source; as ferramentas resultantes desse projeto; a empresa Docker Inc., principal apoiadora do projeto; e as ferramentas compatíveis formalmente com a empresa. O fato da empresa e das tecnologias terem o mesmo nome pode causar alguma confusão.
- O software de TI "Docker" é uma tecnologia de conteinerização para criação e uso de containers Linux®.
- A comunidade open source do Docker trabalha gratuitamente para melhorar essas tecnologias para todos os usuários.
- A empresa Docker Inc. se baseia no trabalho realizado pela comunidade do Docker, tornando-o mais seguro, e compartilha os avanços com a comunidade em geral. Depois, ela oferece aos clientes empresariais o suporte necessário para as tecnologias, que foram aprimoradas e fortalecidas.
- Com o Docker, é possível lidar com os containers como se fossem máquinas virtuais modulares e extremamente lightweight. Além disso, os containers oferecem maior flexibilidade para você criar, implantar, copiar e migrar um container de um ambiente para outro. Isso otimiza as apps na nuvem.

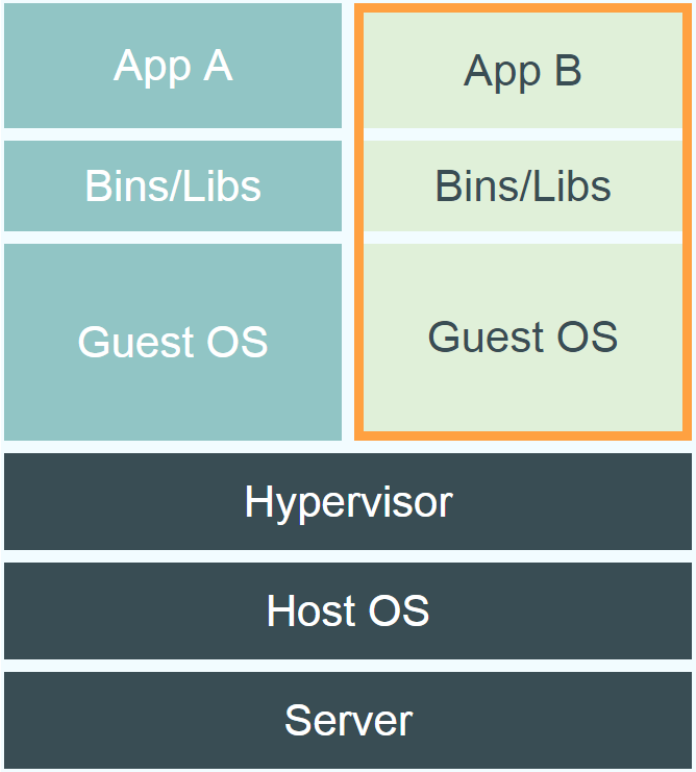


Conteinerização - Docker

- Docker vs. Máquina Virtual



Docker



Máquina Virtual

Conteinerização - Docker

- A tecnologia Docker usa o kernel do Linux e funcionalidades do kernel, como cGroups e namespaces, para segregar processos. Assim, eles podem ser executados de maneira independente. O objetivo dos containers é criar independência: a habilidade de executar diversos processos e apps separadamente para utilizar melhor a infraestrutura e, ao mesmo tempo, manter a segurança que você teria em sistemas separados.

- As ferramentas de container, incluindo o Docker, incluem um modelo de implantação com base em imagem. Isso facilita o compartilhamento de uma aplicação ou conjunto de serviços, incluindo todas as dependências deles em vários ambientes. O Docker também automatiza a implantação da aplicação (ou de conjuntos de processos que constituem uma app) dentro desse ambiente de containers.

- Essas ferramentas baseadas nos containers Linux fazem com que o Docker seja exclusivo e fácil de usar. Elas também oferecem aos usuários acesso sem precedentes a apps e total controle sobre as versões e distribuição, além da habilidade de implantar com rapidez.

- Embora isso possa causar confusão, o Docker não é o mesmo que um container Linux tradicional. A tecnologia Docker foi desenvolvida inicialmente com base na tecnologia LXC, que a maioria das pessoas associa aos containers Linux "tradicionais". No entanto, desde então, essa tecnologia tornou-se independente. O LXC era útil como uma virtualização lightweight, mas não oferecia uma boa experiência para usuários e desenvolvedores. A tecnologia Docker oferece mais do que a habilidade de executar containers: ela também facilita o processo de criação e construção de containers, o envio e o controle de versão de imagens, entre outros.

- Os containers Linux tradicionais usam um sistema init capaz de gerenciar vários processos. Isso significa que aplicações inteiras são executadas como uma. A tecnologia Docker incentiva a segregação de aplicações em processos separados e oferece as ferramentas para fazer isso. Essa abordagem granular tem algumas vantagens..

Conteinerização - Vantagens

- Modularidade
  - A abordagem do Docker para a conteinerização se concentra na habilidade de desativar um pedaço de uma aplicação, seja para reparo ou atualização, sem interrompê-la totalmente. Além dessa abordagem baseada em microsserviços, é possível compartilhar processos entre várias apps da mesma maneira como na arquitetura orientada a serviço (SOA).

- Camadas e controle de versão de imagens
  - Cada arquivo de imagem Docker é composto por uma série de camadas combinadas. Quando a imagem muda, uma camada é criada. Assim como sempre que alguém especificar um comando, como run ou copy.
  - O Docker reutiliza essas camadas para criar novos containers, o que acelera o processo. Para otimizar a velocidade, o tamanho e a eficiência, ocorrem mudanças intermediárias nas imagens. Algo que também é inerente à criação de camadas no controle de versão: sempre que ocorre uma nova alteração, é gerado um changelog integrado, o que oferece controle total sobre as imagens do container.

- Reversão
  - Talvez a melhor parte da criação de camadas seja a reversão. Toda imagem tem camadas. Não gostou da iteração atual de uma imagem? Basta revertê-la para a versão anterior. Esse processo é compatível com uma abordagem de desenvolvimento ágil e possibilita as práticas de integração e implantação contínuas (CI/CD) em relação às ferramentas.

- Implantação rápida
  - Executar, provisionar e disponibilizar um novo hardware costumava levar dias, um processo muito cansativo. Com os containers Docker, a implantação leva alguns segundos. Crie um container para cada processo para agilizar o compartilhamento desses processos com novas apps. Não é necessário inicializar o sistema operacional para adicionar ou mover um container, o que reduz drasticamente o tempo de implantação. Além disso, é possível criar dados e destruir os criados pelos containers de forma fácil e econômica sem nenhuma preocupação.

Docker - Comandos


```
docker pull imagem                      => Puxa uma imagem do docker hub
docker images                           => Mostra imagens baixadas
docker run imagem                       => Cria um conteiner
docker ps                               => Mostra os contêineres que estão rodando
docker ps -a                            => Mostra os contêineres criados (rodando ou não)
docker rm nome_conteiner                => Exclui o contêiner (Não pode estar rodando)
docker rmi nome_imagem                  => Exclui a imagem (Não pode ter contêiner criado a partir dela)
docker stop nome_conteiner              => Para o contêiner
docker start nome_conteiner             => Inicia o contêiner
docker run -d -t --name nome imagem     => Cria um contêiner (Se a imagem não existeir, ele faz o download antes de criar)
docker exec -i -t nome_conteiner proc   => Faz com que o contêiner rode o processo solicitado

Cláusulas:
-d                                      => Rodar a aplicação em background (detached)
-i                                      => Modo interativo, permite comunicação com o processo solicitado
-t                                      => Terminal ficar ativo
--name                                  => Dá um nome contêiner
-v                                      => Mapeia uma pasta (volume) do S.O hospedeiro no contêiner
-P                                      => Criar uma comunicação direta entre host e contêiner a partir de uma porta que esteja disponível
-p                                      => Criar uma comunicação direta entre host e contêiner a partir de uma porta específica
-e                                      => Cria ou modifica alguma configuração das variáveis do ambiente do conteiner
--restart=always                        => Reinicia o contêiner automaticamente se ele cair

Dentro de um contêiner:
ctrl + p & ctrl + q                     => sai do contêiner
exit                                    => sai do contêiner (Pode para a execução do contêiner)
```

Dockerfile - Comandos

- DOCKERFILE – Arquivo que contêm as configurações da geração de uma imagem docker customizada

- FROM – Determina qual imagem será chamada para servir como base da criação da imagem
  - Exemplo:
    - FROM alpine:3.9
    - FROM ubuntu:20.04
    - FROM httpd:latest

- RUN – Permite executar comandos na montagem da imagem
  - Exemplo:
    - RUN apt update
    - RUN apt install -y nano
    - RUN apt-get install -y nano

- COPY x ADD – Permitem enviar arquivos para uma pasta específica da imagem
- COPY . <path>
- ADD – Utilizado para copiar arquivos compactados já descompactando na cópia
  - Exemplo:
    - COPY . /tmp
    - ADD /path/arquivo.tar.bz2 /tmp
    - ADD http://site.com/arquivo.tar.gz /tmp

- CMD x ENTRYPOINT – A partir do momento da criação de um contêiner baseado na imagem, irá rodar o comando definido TODA VEZ que o contêiner inicializar
  - Exemplo:
    - java -jar app.jar
    - CMD [“java”,”-jar”,”app.jar”]
    - ENTRYPOINT [“java”,”-jar”,”app.jar”]

- EXPOSE – Informa qual porta deverá ser mapeada
  - Exemplo:
    - EXPOSE 8080
    - EXPOSE 3306
    - EXPOSE 33060

- VOLUME – Determina o ponto de montagem de uma pasta que ficará disponível para mapeamento host <-> contêiner
  - Exemplo:
    - VOLUME /tmp
    - VOLUME /usr/src

- WORKDIR – Diretório padrão do contêiner.
  - Exemplo:
    - WORKDIR /usr/app

- USER – Define o usuário que executa comandos
  - Exemplo:
    - USER node

- LABEL – Apresenta o criador da imagem
  - LABEL maintainer = “Leandro Colevati”

Dockerfile – Criar Imagem

1. Salvar o arquivo como Dockerfile
2. Rodar o comando:
   - docker build -t nome_imagem .
     - -t - Permite nomear a imagem
3. Para ver a execução do background do contêiner:
   - docker logs nome_contêiner
4. Para disponibilizar no docker hub:
   - docker login
   - docker tag imagem:tag usuario/imagem:tag
   - docker push usuario/imagem:tag