Habilitando Aplicações Nativas de Nuvem

Introdução a Contêineres e Kubernetes

1. Conceitos Básicos e Virtualização

Sergio Rio
www.sergiorio.tech

Refrescando nossa memória

- Na aula passada estudamos a motivação e o programa desse curso.
- Iniciamos o curso com a revisão de conceitos básicos como abstração, definição de aplicativos nativos de nuvem e sua importância para os sistemas distribuídos modernos.
- Atividades preparatórias para a aula de hoje:
  - Por que o sistema binário (base 2) foi escolhido para ser usado em computadores digitais?
  - Assistir palestra “Visão geral sobre aplicativos nativos de nuvem e Kubernetes”
  - Capitulo 1 do livro “Kubernetes Up & Running”, Brendan Burns, Joe Beda, Kelsey Hightower, and Lachlan Evenson – cópia cortesia da VMware Inc.

Por que o sistema binário (base 2) foi escolhido para ser usado em computadores digitais?

- Na pratica
  - A única maneira de diferenciar os estados é pela tensão ou intensidade da corrente elétrica que por um circuito elétrico.
  - Em suma, é mais simples determinar se um sinal está ou não ativado ou desativado (1 ou 0) do que se ele corresponde a um dos muitos estados em uma escala mais granular. Na base 2 a eletricidade varia um pouco ao passar por um circuito, podemos dizer que “1” é qualquer coisa acima de x, e tratar como “0” qualquer coisa abaixo de x, mesmo que haja alguma corrente passando.
  - Mas digamos que você tenha um sistema de base 3 ou qualquer coisa acima da base 2. Você não pode mais confiar em apenas um limite de tensão ou corrente. É preciso intervalos adicionais de valores que represente estados intermediários. Na base 3, um sinal deve ser modulado para estar desligado, meia potência ou potência total. Cada transistor precisa ser capaz de identificar o nível de potência do sinal e produzir um resultado apropriado. À medida que você aumenta a base, a complexidade exigida aumenta exponencialmente.
- Na teoria
  - Para quantificar os custos relativos do uso de diferentes raízes ou bases na representação de números, especialmente em sistemas computacionais, devemos observar que matematicamente a base mais eficiente (medida pela economia radix) é a base "e".
  - Como as bases inteiras são muito mais fáceis de trabalhar, isso deixa as bases 2 e 3 como as escolhas mais próximas que podemos obter da maior eficiência.
- Leia mais sobre o assunto em Teoria Matemática da Informação.

Dúvidas, questões ou comentários?

Programa: Introdução a Contêineres e Kubernetes

1. Conceitos Básicos

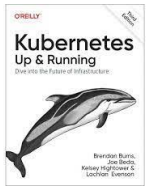
- ✓ Abstrações em Ciência da Computação
- ➢ Virtualização de Computadores
- ➢ MicroVMs e Unikernels
- ▪ Aplicativos Modernos

2. Containers

- ▪ Origem
- ▪ Fundamentos
- ▪ Criação e execução
- ▪ Registro e reuso
- ▪ Infraestrutura como Código

3. Kubernetes

- ▪ Origem
- ▪ Arquitetura
- ▪ Pods
- ▪ Abstrações de Recursos
- ▪ Descoberta de Serviços
- ▪ Serviços de Rede
- ▪ Instalação e administração básica
- ▪ Implantação de um caso de uso (exemplo)



Kubernetes Up & Running
Brendan Burns, Joe Beda, Kelsey Hightower, and Lachlan Evenson
Cortesia da VMware Inc.

Conceitos Básicos

Virtualização de Computadores

“The datacenter is the computer.”

Luiz Andre Barroso, Google (2007)

Association for Computing Machinery

ACM recognizes excellence

ACM Home | ACM A.M. Turing Award | Turing 50 | Digital Library | CACM | Queue | TechNews

Home | Award Recipients | Contact Us | Search

ACM AWARDS | ADVANCED MEMBER GRADES | SIG AWARDS | REGIONAL AWARDS | NOMINATIONS | HONORS AND ETHICS | AWARDS COMMITTEES | ESTABLISHING AN AWARD

Home > Award Recipients > Luiz Andre Barroso

Luiz Andre Barroso

ACM-IEEE CS Eckert-Mauchly Award

Brazil - 2020

CITATION

For pioneering the design of warehouse-scale computing and driving it from concept to industry.

Press Release

ACM Fellows

Brazil - 2010

CITATION

For contributions to multi-core computing, warehouse scale data-center architectures, and energy proportional computing.

2020 ACM - IEEE CS Eckert-Mauchly Award

ACM and IEEE Computer Society named **Luiz André Barroso**, Vice President of Engineering at Google, recipient of the 2020 **Eckert-Mauchly Award** for pioneering the design of warehouse-scale computing and driving it from concept to industry. Today's datacenters contain hundreds of thousands of servers and millions of disk drives, and make possible the most prevalent applications used by the public today, including cloud computing, powerful search engines, and internet services.

Barroso is widely recognized as the foremost architect of the design of these new ultra-scale datacenters. The cornerstone of his architectural vision was to think of a system holistically, weaving together the individual compute, storage, and networking components into an overall design across large-scale distributed systems.

Barroso has been a thought leader in the field, writing seminal papers and books which reconsidered every aspect of data center and system design. At the same time, he has also guided industry efforts in this area. He was the lead architect of Google's first custom-built data centers and has been the primary technical leader steering the development of Google's computing infrastructure for much of the last two decades. This work has been replicated by other large companies. Virtually all the hardware architectures that power today's internet services and cloud computing systems feature elements introduced by Barroso and his team at Google.

Warehouse-scale computing

Barroso proposed the idea that a datacenter should be designed as a single, massive warehouse-scale computer, popularizing the phrase "the datacenter is the computer." The workloads of these computers are internet services that run on thousands of CPUs across high-bandwidth networks and require specialized storage systems. Barroso's designs paired inexpensive hardware with powerful distributed

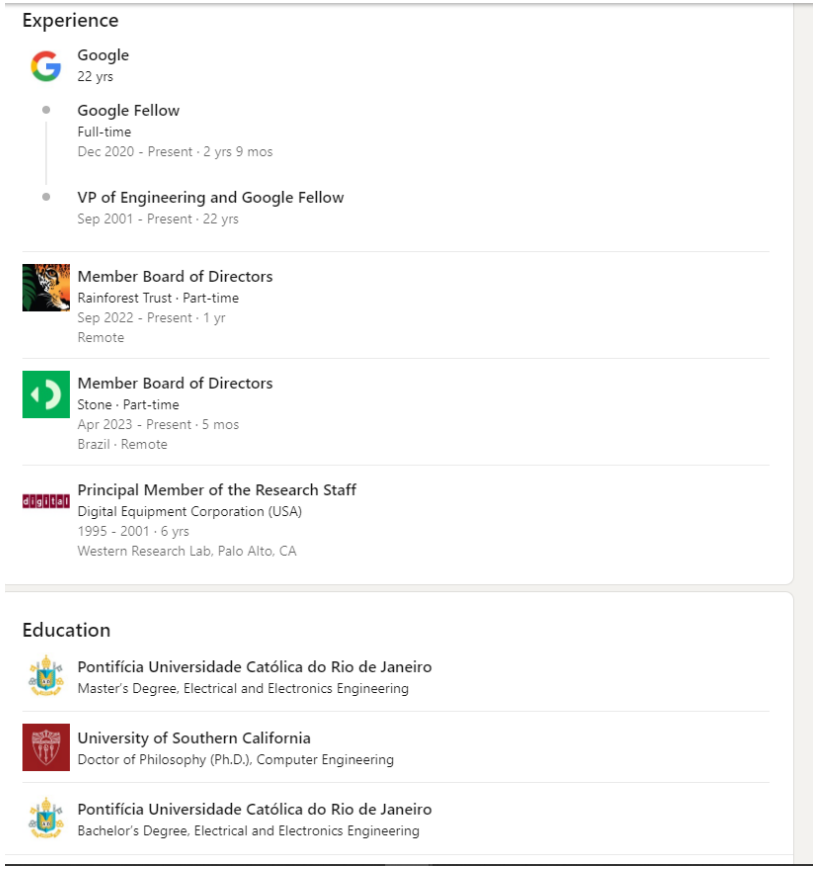
AWARD RECIPIENT

Luiz Andre Barroso

ACM-IEEE CS Eckert-Mauchly Award (2020)

ACM Fellows (2010)

2020 ACM - IEEE CS Eckert-Mauchly Award



https://awards.acm.org/award-recipients/barroso_UJ31885
https://www.linkedin.com/in/luiz-andr%C3%A9-barroso-00255a71/

Um ponto de vista econômico

- A nuvem e’ um serviço que tem os seguintes atributos[1]:

1. Infraestrutura comum – recursos padronizados e agrupados com benefícios gerados pela multiplexação estatística.
2. Localização independente – disponibilidade generalizada atendendo a requerimentos de desempenho, com benefícios derivados da redução de latência e melhora na experiencia dos usuários.
3. Conectividade on-line – habilitador de outros atributos que asseguram acesso ao serviço.
4. Preço utilitário – sensível ao uso, precificado pelo uso com benefícios em ambientes com níveis de demanda variável.
5. Recursos baixo demanda – recursos elásticos e escaláveis provisionados ou descontinuados sem demora ou custos associados à mudança.

A nuvem demanda recursos compartilhados!

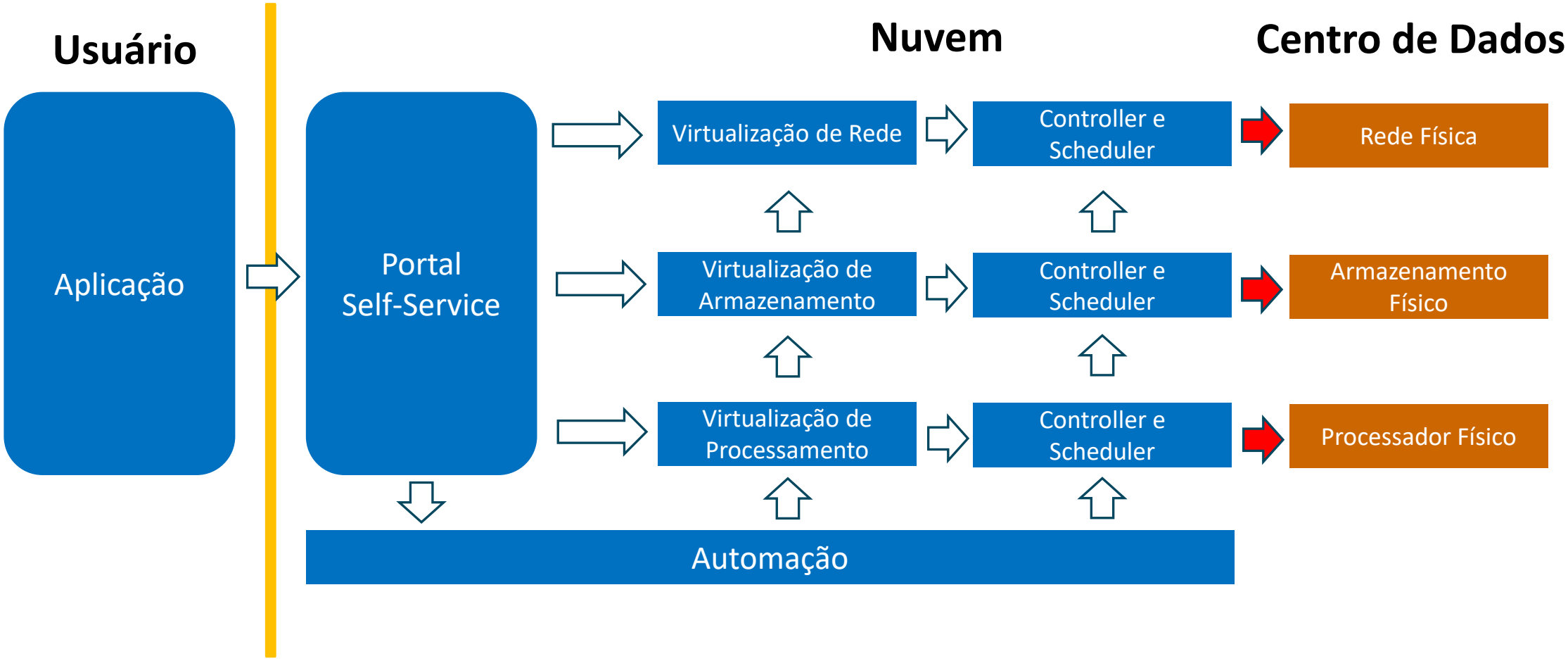
1. Cloudonomics: A Rigorous Approach to Cloud Benefit Quantification

Como compartilhar um computador físico entre vários usuários?

Abstração!

- Introduzir um modelo abstrato de recurso genérico computacional.
- O computador físico proporciona então esse modelo abstrato aos usuários.
- A virtualização evita criar dependências dos recursos físicos.
- As nuvens são baseadas na virtualização:
  - Oferecem serviços basicamente em maquinas virtuais, chamadas remotas a processos (RPCs) e clientes/servidores.
  - Proveem vários servidores a múltiplos clientes.
- A simplicidade no uso e a facilidade na programação requer paradigmas cliente-servidor para a construção de serviços a partir de outros recursos.

Centro de Dados Definido por Software (SDDC)



Mas como chegamos aqui?

- Os computadores digitais originais executavam somente um programa a cada vez:
  - Um único programa de usuário com acesso a todo o hardware disponível.
  - O sistema operacional era uma camada muito simples em volta do BIOS.
  - Não existia o conceito de “processo”.
  - Exemplos: primeiros mainframes nos anos 1950 e o MS-DOS nos 1990.
- A habilitação de computadores multiusuários e multitarefas requer:
  - Isolar programas.
  - Isolar usuários.
  - Abstração de “processo”: um programa em execução e seu contexto.

Dúvidas, questões ou comentários?

Definição de “processo”

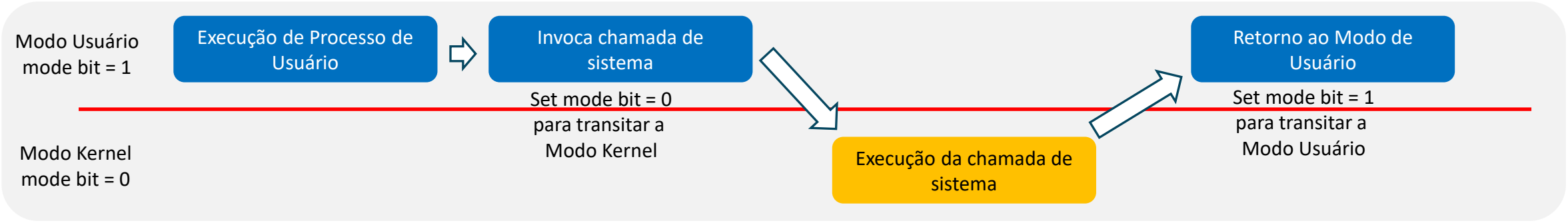
- Imagem de código executável em linguagem de máquina.
- Uso de memória:
  - Espaço de endereçamento virtual→ paginação → uma página de memória virtual carregada na memória física quando o processo tenta usá-la → administrado pelo Sistema Operacional.
  - Dados (entrada/saída) específicos do processo.
  - Pilha dedicada – dados temporários como por exemplo parâmetros de função, variáveis locais, endereços de retorno e pilha de chamadas de função.
- Descritores de recursos de Sistema Operacional: como por exemplo descritores de arquivos, fontes de dados, etc.
- Atributos de segurança: como por exemplo dono do processo e conjunto de atributos de permissões do processo (operações permitidas).
- Estado do processador (contexto):
  - Contador de programa
  - Conteúdo dos registradores e endereçamento de memória física.

Necessidade de isolamento de processos

- Processos não podem ser capazes de se comunicar entre si exceto através de mecanismos seguros administrados pelo núcleo (kernel) do sistema operacional.
- Uso restrito de instruções críticas como por exemplo carga de tabelas de mapeamento de memória e acesso direto a dispositivos físicos de entrada e saída.
- Uso de uma máquina idealizada virtual.
- Imagine o que poderia acontecer se uma aplicativo normal pudesse escrever em posições arbitrarias memória ou acessar diretamente dispositivos de armazenamento (discos rígidos ou memória de estado sólido).
- A CPU e o sistema operacional devem trabalhar juntos para assegurar o isolamento, havendo dois modos de operação:
  - Modo Usuário
  - Modo Kernel

Modos de Usuário e de Kernel

- Modo Usuário
  - Processos de usuário são executados nesse modo.
  - Quando o processo de usuário solicita um serviço do sistema operacional, ou ocorre uma interrupção, a execução forçosamente transita para Modo Kernel.
- Modo Kernel
  - Na inicialização do sistema (boot), o hardware inicia o Modo Kernel.
  - Instruções privilegiadas são executadas somente em Modo Kernel. Qualquer solicitação de usuário para executa-las são capturadas (trap) pela CPU/BIOS.
  - Gestão de interrupções feita pelo Modo Kernel.



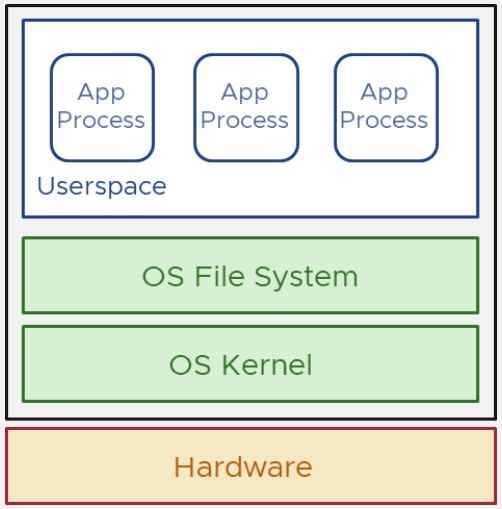
Instruções Protegidas e Não-Protegidas

- Goldberg e Popek[2] estabeleceram critérios para um sistema (hardware + sistema operacional) ser virtualizável:
  - Capacidade de *Trap* e *Emulate*.
  - Dois tipos de instruções: seguras e não-seguras.
  - Capacidade de separar (Trap) instruções não-seguras.
  - Instruções protegidas são um subconjunto de instruções não-seguras.
- Arquitetura original da família X86 não é virtualizável. Lacunas foram resolvidas na seguinte geração Intel VT-x e AMD-V.

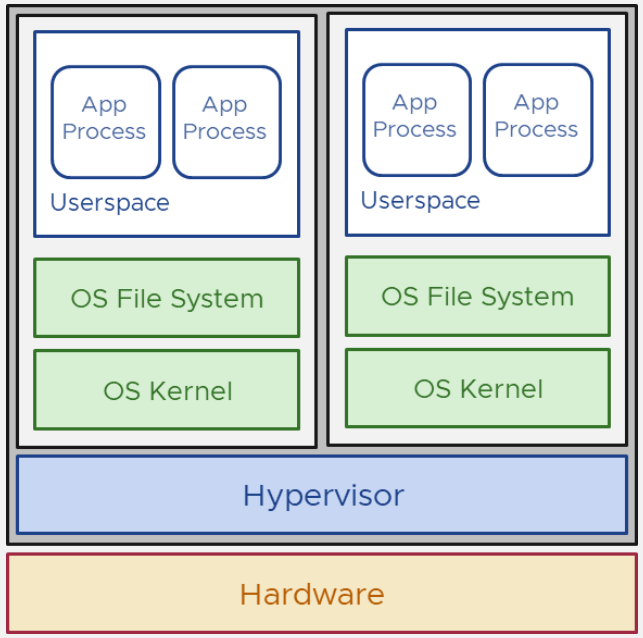
2. Formal requirements for virtualizable third generation architectures, Goldberg and Popek 1974, Communications of the ACM Volume 17 Issue 7, July 1974 pp 412–421

Dúvidas, questões ou comentários?

Hypervisors



Máquina Física



Máquinas Virtuais

- O Hypervisor (gerenciador de máquina virtual) simula ter hardware suficiente para cada sistema operacional.
- Sistemas operacionais isolados.
- Estratégia de *Trap* e *Emulate* foi usada nos primeiros Hypervisors (IBM System 370) e retomada atualmente nas arquiteturas Intel e AMD de 64 bits.
- Exemplos:
  - VMware ESX
  - VirtualBox
  - QEMU

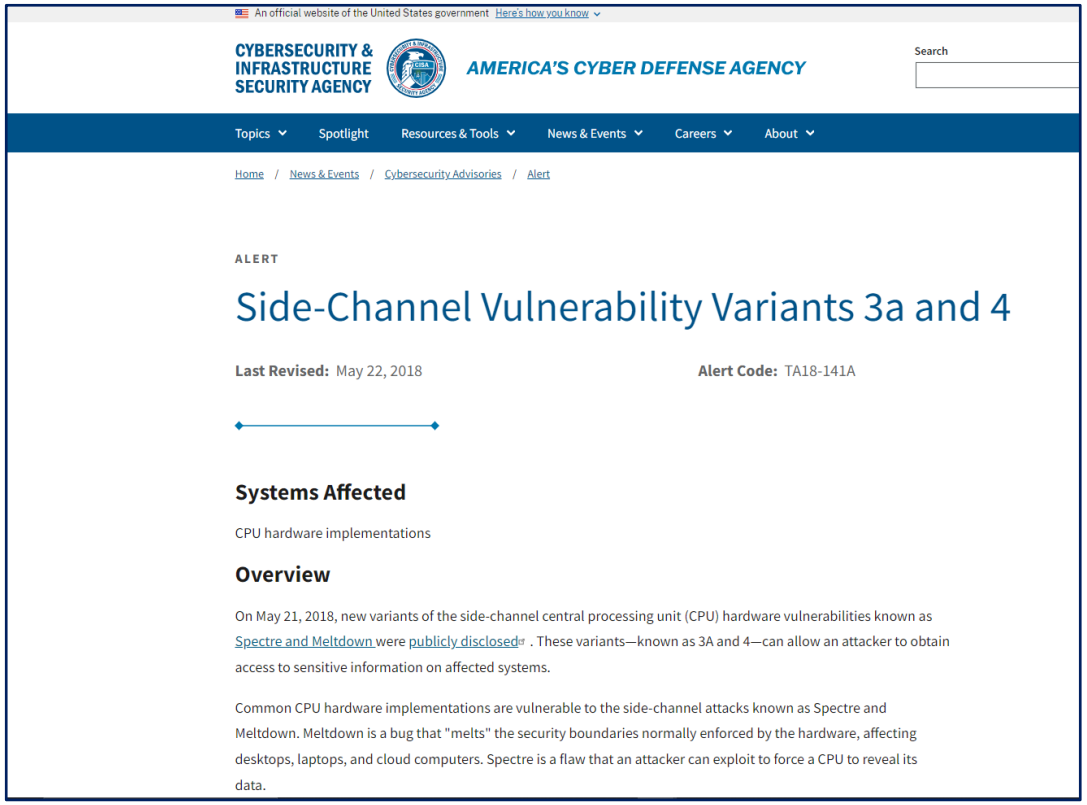
Taxonomia de tipos de virtualização

- Software
  - *Tradução de código binário*: hypervisor traduz código binário de instruções protegidas em tempo de execução e não requer modificações nos sistemas operacionais (exemplo VMware Workstation original de 1999, suporte suspenso em 2016!).
  - *Para-virtualização*: virtualização exclusiva por software para CPUs que atendem aos critérios de Goldberg e Popek. Hypervisor não simula o hardware mas oferece APIs para sistemas operacionais modificados (exemplo Xen/Linux 3).
- Suportado por Hardware
  - Grande impulso na terceira geração baseada na microarquitetura Haswell[3] (anunciada em 2013).

3. List of Intel CPU microarchitectures

Vulnerabilidades conhecidas

- As vulnerabilidades ligadas à execução especulativa de instruções têm sido objeto de muita pesquisa por mais de cinco anos[4].
- De maneira simplificada:
  1. CPU é forçada a ler dados aos quais o usuário não deveria ter acesso. Imagine um cenário teórico onde o programa invasor não tem acesso à chave de criptografia usada para proteger dados confidenciais. Se instruirmos a CPU a “ler a chave de criptografia em um determinado endereço”, a instrução simplesmente não será seguida.
  2. A ajuda chega (ao invasor) na forma de execução especulativa de instruções – uma característica importante das CPUs modernas, que existe há quase três décadas: para acelerar, em vez de esperar que uma instrução termine, o processador executa a seguinte em paralelo.



4. New hardware vulnerability in Intel processors

Dúvidas, questões ou comentários?

Conceitos Básicos

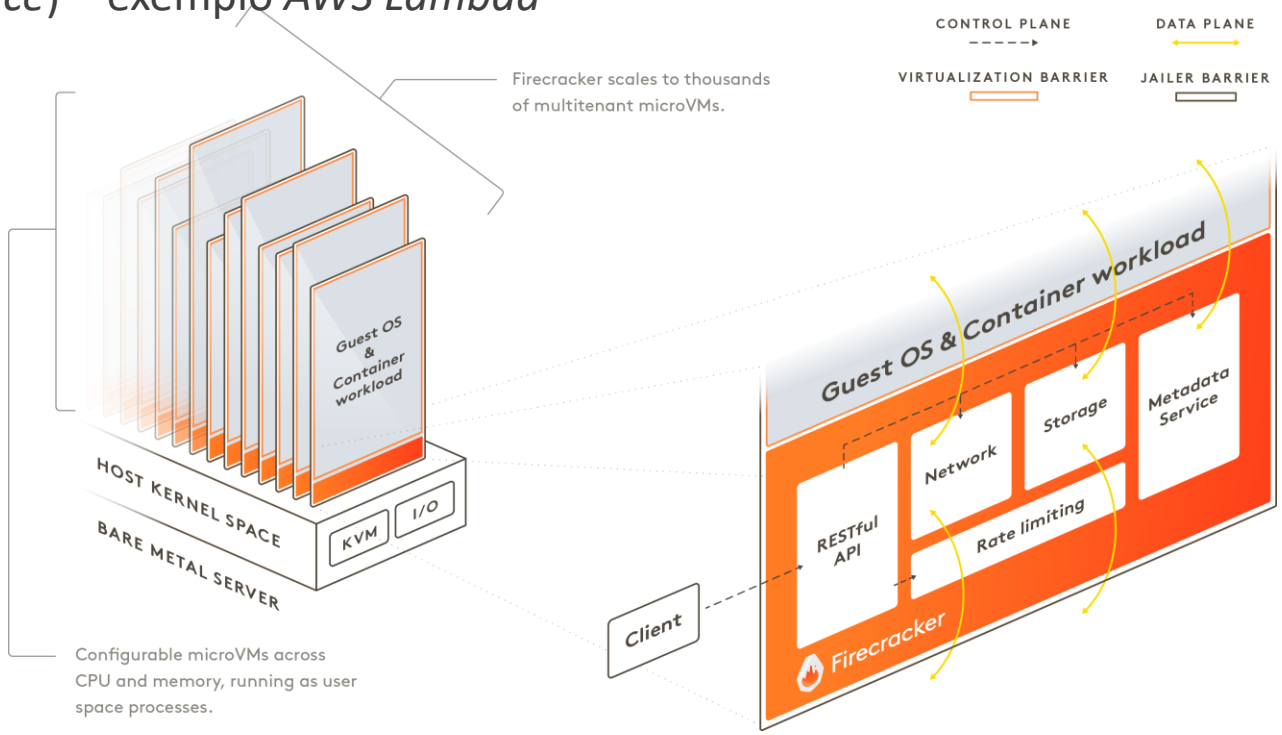
MicroVMs e Unikernels

MicroVMs 1/2

- Máquinas virtuais são soluções “pesadas” que requerem recursos de computação significativos
- Em geral os tempos de inicialização são longos, limitando a velocidade com que podem ser criadas.
- Um aplicativo nativo de nuvem precisa apenas de alguns dispositivos de hardware, como rede e armazenamento. Não há necessidade de dispositivos como teclados completos e monitores de vídeo. Por que executar o aplicativo em uma máquina virtual que fornece várias funcionalidades desnecessárias?
- Os hypervisors microVM podem ser extremamente rápidos com baixa sobrecarga:
  - Tempos de inicialização em milissegundos (ao contrário de minutos para máquinas virtuais tradicionais)
  - O sobre uso de memória pode ser de apenas 5 MB de RAM, possibilitando a execução de milhares de microVMs em um único servidor bare metal.

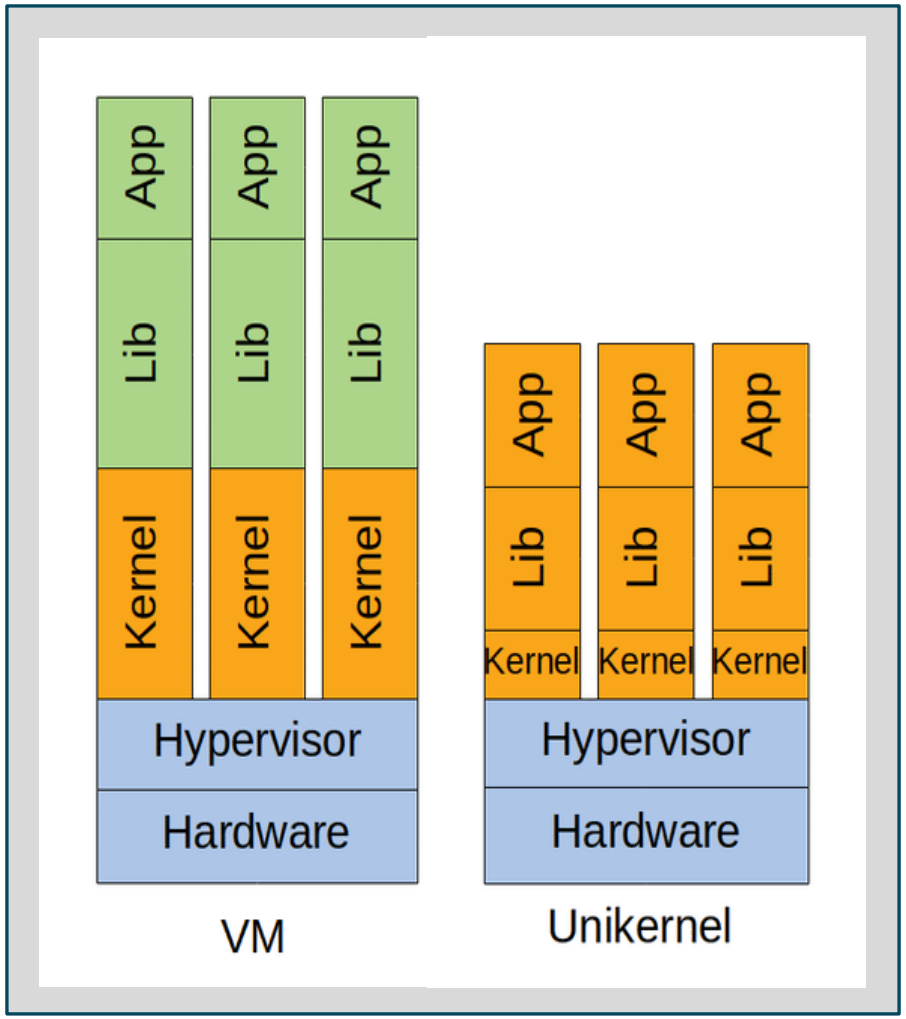
MicroVMs 2/2

- MicroVMs são projetadas para aplicativos nativos de nuvem
  - Casos de uso *Serverless Computing*
  - *Serverless Containers* – exemplo *AWS Fargate*
  - Função como Serviços (*Function as a Service*) – exemplo *AWS Lambda*
- Projeto de Código Aberto *Firecraker*



Unikernels[5]

- Como as micros, os Unikernels permitem executar aplicativos nativos da nuvem com alto desempenho e baixa sobrecarga, ao mesmo tempo em que fornecem uma forte postura de segurança.
- Embora os Unikernels abordem os mesmos problemas que as microVMs, eles o fazem de maneira radicalmente diferente.
- Um Unikernel é um sistema operacional leve e imutável, **compilado** especificamente para executar um único aplicativo.
- Durante a compilação, o código-fonte do aplicativo é combinado com os drivers de dispositivo mínimos e as bibliotecas do sistema operacional necessárias para dar suporte ao aplicativo. O resultado é uma imagem de máquina que pode ser executada sem a necessidade de um sistema operacional host.
- Unikernels alcançam seus benefícios de desempenho e segurança colocando severas restrições na execução.
- Unikernels só podem ter um único processo. Quando empacotado como um Unikernel, o aplicativo não pode gerar subprocessos. Sem outros processos em execução, há menos área de superfície para vulnerabilidades de segurança.



5. Unikernels vs Containers: An In-Depth Benchmarking Study in the Context of Microservice Applications

Dúvidas, questões ou comentários?

Referências adicionais

- The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines. Second Edition, Luiz André Barroso, Jimmy Clidaras and Urs Hölzle
- A Brief History of Warehouse-Scale Computing, Reflections Upon Receiving the 2020 Eckert-Mauchly Award, 2021
- Cloudonomics: A Rigorous Approach to Cloud Benefit Quantification
- OpenStack is an open-source cloud computing infrastructure software project and is one of the three most active open-source projects in the world.
- Formal requirements for virtualizable third generation architectures, Goldberg and Popek 1974, Communications of the ACM Volume 17 Issue 7, July 1974, pp 412–421.
- Secure and fast microVMs for serverless computing
- Unikernels vs Containers: An In-Depth Benchmarking Study in the Context of Microservice Applications

Atividades para a próxima aula

- Assistir vídeos:
  - Tour no data center IBX SP3 da Equinix em São Paulo (15 minutos)
  - Why does Microsoft have underwater data centers? (5 minutos)
  - Over 200,000 Servers in One Place! Visiting Hetzner in Falkenstein (Germany) (20 minutos)

- Ler Unikernels vs Containers: An In-Depth Benchmarking Study in the Context of Microservice Applications e responder as seguintes perguntas:
  1. Quais tipos de aplicações são apropriadas para utilizaram Unikernels? Por que? Mencionar exemplos.
  2. Na sua opinião quais as principais limitações de uso de Unikernels?
  3. Conforme os resultados reportados no artigo, quais são as diferenças de desempenho entre Unikernels e Contêiners?
  4. Na sua opinião, quais os critérios para uma aplicação utilizar Unikernel ou Contêiner? Explique com exemplos.