Sistemas de Computação em Cloud

First generation batch processing: Map-reduce

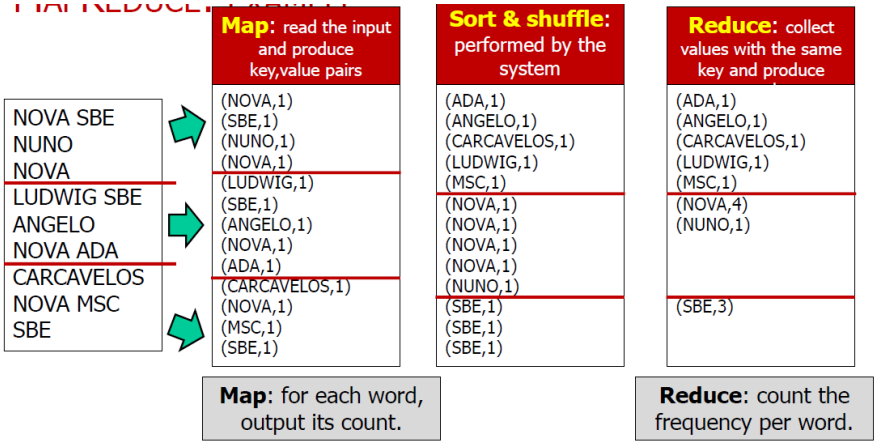
Permite expressar cálculos simples que estávamos tentando realizar, mas esconde os detalhes confusos de:

- paralelização
- tolerância a falhas,
- distribuição de dados
- balanceamento de carga em uma biblioteca

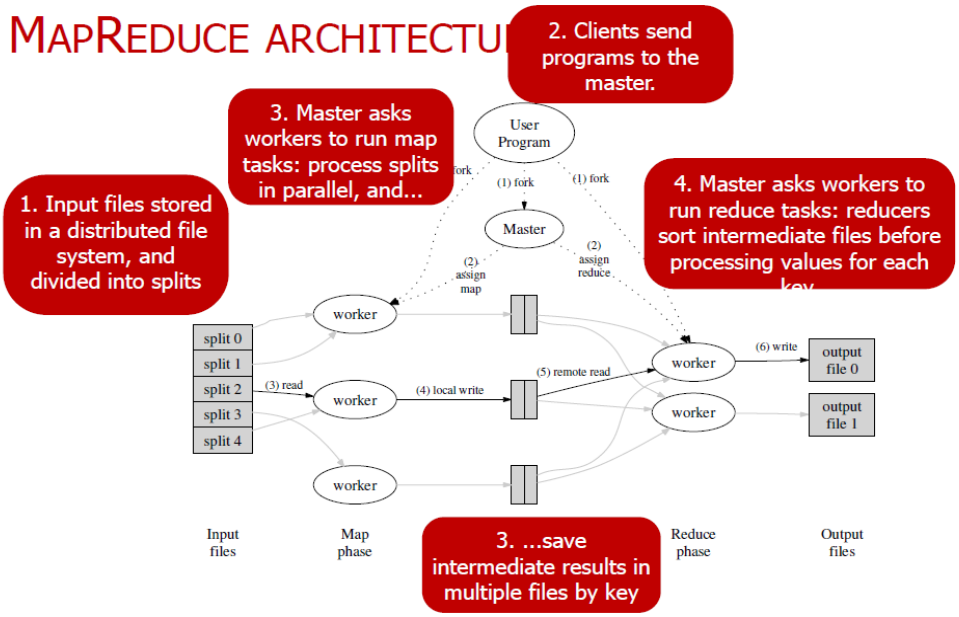
Um modelo de programação e uma implementação associada para processar grandes conjuntos de dados

OverView

1. Lê sequêncialmente muita informação
2. Map phase – extrai a info. importante
3. Agrupa por chave (Sort and Shuffle do output da Map phase – por frequência da palavra)
4. Reduce phase - Aggregate, summarize, filter or transform
5. Dá o resultado



Arquitetura



Master Node

O **Master node** coordena a execução:

- Task status: (idle, in-progress, completed)
- Idle Tasks: as tarefas ociosas são agendadas à medida que os trabalhadores ficam disponíveis
- Quando uma tarefa de mapa é concluída, ela envia ao mestre a localização e os tamanhos de seus arquivos intermediários, um para cada redutor
- O master envia essas informações para os redutores
- O mestre pinga os trabalhadores periodicamente para detectar falhas

Worker

O nó do trabalhador executa tarefas de mapeamento ou redução, conforme solicitado pelo coordenador.

Handling Faults

Map worker failure

- Após a detecção da falha de um trabalhador, as tarefas do mapa são reiniciadas em diferentes trabalhadores

Reduce worker failure

- A tarefa de redução é reiniciada em outro trabalhador

Stragglers (slow workers)

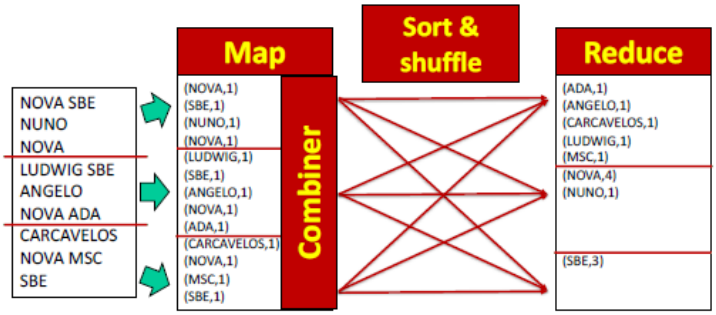
- Se uma tarefa estiver demorando muito para ser concluída, ela será iniciada em outro trabalhador. E usado o primeiro resultado.

Falha mestre

- A tarefa MapReduce é abortada e o cliente é notificado

Map reduce Improvement – Combiner

Combiner permite pré-agregar valores no mapper. Normalmente o mesmo que a função de redução



Desvantagens

O MapReduce requer cálculos complexos para serem divididos em sucessivos MapReduce jobs

Apresentam alta latência devido a:

- precisa ler e gravar arquivos
- replicação do sistema de arquivos fundamental (para gravações)
- um trabalho deve terminar antes que o próximo possa ser iniciado

O Apache Spark aborda essas limitações.

Second generation batch processing: Spark

Apache Spark fornece:

- processamento distribuído na memória
- tolerante a falhas
- múltiplas transformações de dados encadeadas
- modelo de programação funcional de alto nível

Data Models e API's

RDDs

- são tuplos de dados imutáveis
- é dividido em partições no cluster podendo ser operado em paralelo em diferentes nós do cluster
- particionado em vários nós;
- após uma falha os RDDs podem ser automáticamente recriados e de forma eficientemente, a partir das dependências.

Quadros de dados do Spark

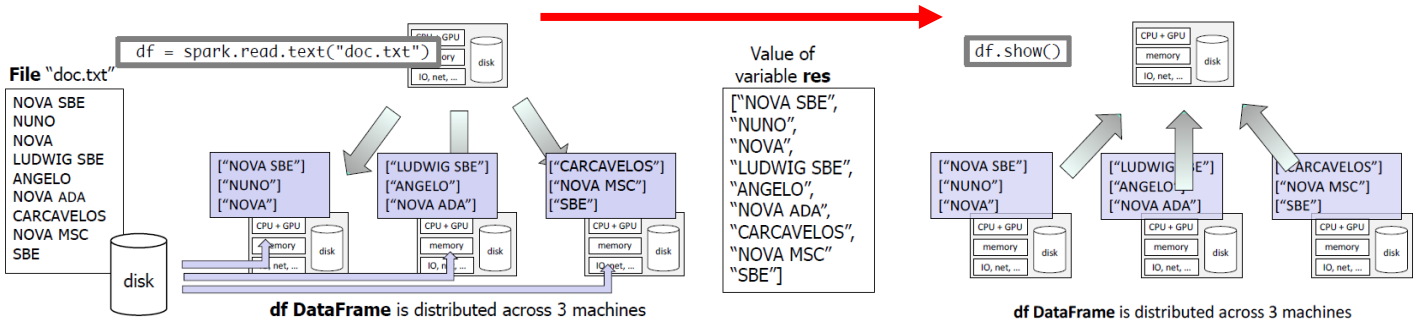
DataFrames - coleções distribuídas de dados agrupadas em colunas com um nome. Podem ser vistas como RDD's com um esquema que nomeia os campos dos tuplos fundamentais.

Spark SQL

Programas com SQL/DataFrames → convertidos em programas **Spark**.
Spark SQL usa técnicas de sistemas de BD.

Programas **Spark**:

- otimizados para serem executados com eficiência.
- bons para processar gráficos e aprendizado de máquina.

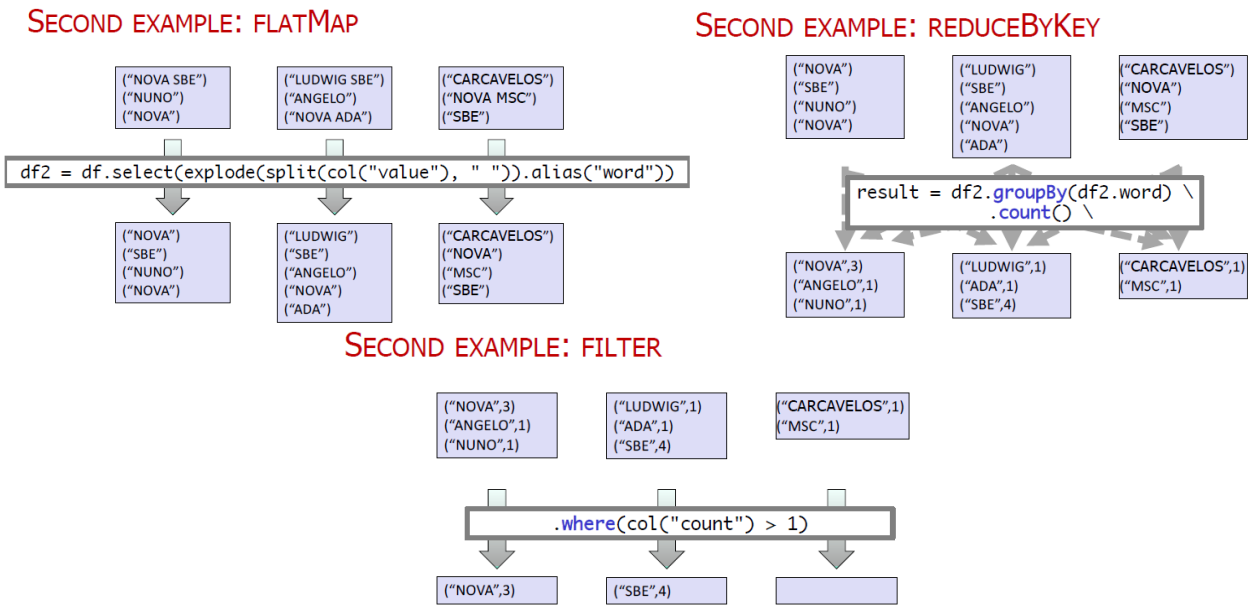


Programming model

Programas Spark Dataframe descrevem o fluxo de transformações que cria uma DataFrame a partir de outra.
Programas Spark codificam as dependências entre os vários DataFrames (e RDDs) - gráfico de linhagem.

São produzidas:
Wide-Dependencies - quando uma **partição RDD** depende de várias partições armazenadas em nós diferentes (alto custo de bandwidth no groupByKey) GroupBy, Join

Narrow-dependencies - quando uma **partição RDD** depende de dados que estão no mesmo nó (melhor desempenho, pois evita embaralhamentos) Filter, Map



Programming and execution model

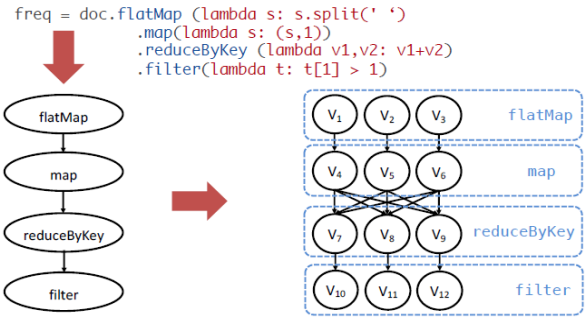
Programas DataFrame → **convertidos** → programas RDD, que envolvem:

- **Transformações**: RDD -> RDD
- **Ações**: RDD -> Resultado (disponível diretamente para o aplicativo cliente)

O **RDD** fornece APIs de baixo nível para processamento de dados distribuídos. Por outro lado, **DataFrame** fornece APIs de alto nível que suportam métodos SQL.

Aplica todas as partições de um RDD em paralelo.

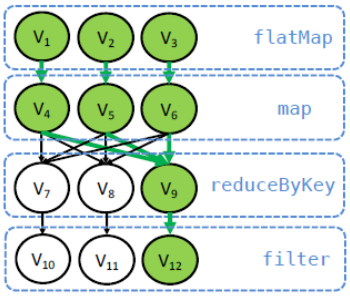
SECOND EXAMPLE: COMPLETE EXECUTION



Fault tolerance

Sparks lida com falhas de nó recalculando partições perdidas, usando informações de linhagem. Otimizado por RDDs intermédias persistentes.

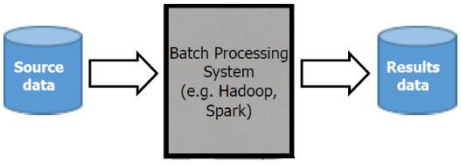
No exemplo, se **V9 for persistente**, muitos recálculos serão salvos.



Stream processing (Big data / batch processing)

Objetivo: Executar computação sobre dados e produzir resultado

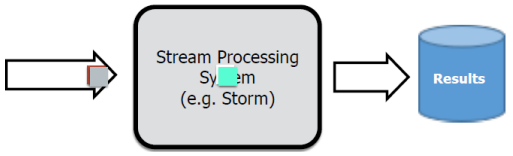
Exemplo: Produzir informação sobre o trânsito com base na informação recolhida dos telemóveis dos users



Processing models

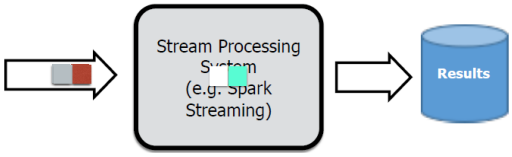
Continuous

- Cada tuplo é processada à medida que chega
- O sistema pode manter o estado para executar **window computation** e **incremental computation**



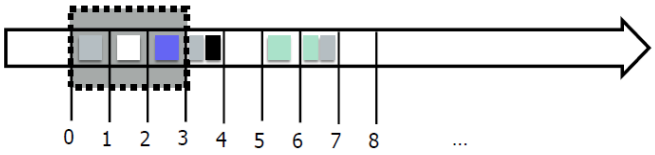
Mini- batches

- Processa tuplos agrupados a cada Xms em um mini-batches
- O sistema pode manter o estado para executar **window computation** e **incremental computation**



Windowing

Calcula-se resultados com base em dados de uma window, mas calcula os resultados mais frequentemente do que o intervalo de tempo (processar dados dos últimos 3 minutos, mas produzir resultados a cada minuto)



Virtualization

Criação de recursos computacionais simulados (**recuros** - rede, armazenamento, CPU+memória).

Vantagens

- Uma VM fornece a abstração de uma máquina para cada cliente numa máquina física.
- Permite compartilhar os mesmos recursos físicos entre vários users.
- Recursos físicos e virtuais estão desacopulados.

Defenições

Máquina Virtual (VM)

- simula a máquina real com um hardware virtual criado e gerido pela VMM
- o software é executado numa VM é isolamento de outras VMs (apenas com reduções na velocidade pequenas)

Virtual Machine Monitor (VMM)

- fornece a ilusão de uma máquina idêntica à real que controla o hardware

Hypervisor

É um SO executa os processos (VM/VMM),

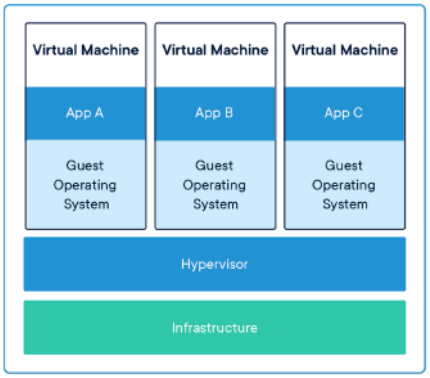
- Um par (VM+VMM) constitui um “processo” de hypervisor

Language VM

Fornece um ambiente com instruções e bibliotecas padrão para aceder os recursos do computador.

- Executa como uma app em um SO, e os programas são compilados e executados respetivamente com o código e linguagem da VM.

Virtualização



A virtualização permite executar um SO dentro de outro

- Host OS/Hypervisor → rodando na máquina real
- SO convidado → corre dentro do SO host

O **VMM** (ou hypervisor) deve fornecer:

- Segurança: a VM não deve comprometer o VMM, que deve ter controle total dos recursos (reais e virtuais).

- Fidelidade: durante a execução, o comportamento do software na VM deve ser idêntico ao comportamento do hardware real.

- Eficiência: o VMM deve se interpor o mínimo possível enquanto o VM está executando seu código.

O **emulador (VMM)** pode manter estado da memória, do processador e IO, e intrepretar cada instrução da VM atualizando esse estado, mas é muito lento. Para que isto se torne rápido é preciso que o software da VM rode no processador fisico todo, mas sem comprometer o sistema.

VM garante

1. Isolamento de memória entre processos;
2. **Operações privilegiadas** – processo acede a recursos do computador mas no modo S (código do SO é o único que é executado no modo supervisor)

VMM garante

O Guest OS da VM . . .

1. . . . precisa pensar que está sendo executado no S-mode (precissa da S flag);
2. . . . precisa ser capaz de emitir instruções privilegiadas, mas sua execução precisa ser controlada pelo VMM.

Implementação

Virtualização com Trap-and-emulate (friendlly)

As **Operações sensíveis** que se comportam de maneira diferente executadas nos modos U e S são preveligiadas

1. A VM executa o código no CPU (sem penalidade de desempenho)
2. Guest OS em U-mode & sinalizadores vCPU no S-mode (Guest OS pense que está a correr em U-mode)
3. Se o **Guest OS executa uma instrução privilegiada, um trap altera a execução para uma função VMM**
4. A VMM interpreta (emula) a instrução e retorna a execução para o Guest OS (em U-mode).

Virtualização com Binary translation (unfriendlly)

Algumas **Operações sensíveis** não são privilegiadas → e.g. PUSHF, POPF

Problema: O Guest OS for executado no modo U (com o sinalizador vCPU S mode), o efeito da execução da operação não será correto.

Ideia: substituir a instrução sensivel por um trap (forçando execução na VMM).

- A VM executa o código no CPU (sem penalidade de desempenho)
- Guest OS em U-mode & sinalizadores vCPU no S-mode (Guest OS pense que está a correr em U-mode)
- O **código binário do Guest OS é alimentado a um tradutor que emite código binário alterado para instruções sensiveis**
- Instruções privilegiadas são executadas como trap-and-emulate.

Desvantagens
- Lento, pois precisa de traduzir o código;
- Várias instruções podem precisar ser capturadas em cada função do Guest OS;

Virtualização com Paravirtualização (unfriendlly)
Problema: mesmo que a tradução de código possa ser rápida, ter várias instruções de cada função do SO convidado bloqueadas retarda a execução.

Ideia: Uma única trap seria necessária para chamar a função VMM. Além disso, a função VMM pode acessar os recursos diretamente (em vez de acessar uma versão virtualizada).

- O código-fonte do **Guest OS é alterado para que as partes que usam instruções sensiveis não privilegiadas e privilegiadas sejam substituídas por chamadas de hipervisor** que executam a mesma funcionalidade (por exemplo, uma função de Guest OS que acederia ao controlador de memória virtual é substituída por uma chamada VMM/hypervisor que acede ao controlador de memória virtual diretamente)
- A VM executa o código na CPU (sem penalidade de desempenho).
- O SO convidado é executado no modo U (mas os sinalizadores vCPU indicam o modo S).
- As chamadas do hipervisor são implementadas como traps.

Vantagens
Melhor performance de um caso não virtualizado.

Desvantagens
Requer alterações no SO convidado: ok quando a fonte estiver disponível (por exemplo, Linux).

Hoje em dia
A **paravirtualização** resolveu o problema de desempenho quando a tradução binária foi usada pois não é necessário com o suporte de hardware para virtualização.

Técnicas de virtualização de CPU's
Virtualização de back-end
A pilha do GPU é executada dentro da VM com o limite de virtualização entre a pilha e o hardware do GPU

GPU PCI passthrough technic
Permite uma VM tem acesso direto à GPU mapeando as suas regiões de memória no espaço de endereço da VM.
- **Vantagem:** Desempenho quase nativo para GPUs
- **Problemas:** Sem multiplexação

Virtualização de front-end

Introduz um limite de virtualização em um nível relativamente alto na pilha e executa o GPU no host/hipervisor

Front-end technic

A GPU é controlada pelo host/hipervisor e oferece suporte à multiplexação.
Na **emulação** de dispositivos, um GPU é emulado e a emulação sintetiza as operações gráficas do host em resposta às ações dos drivers do dispositivo convidado. Isto permite que o GPU seja executado em uma máquina diferente.

IaaS – Azure VM

- Suporta a criação de VM rodando Linux e Windows.
- VMs podem ser adaptadas para a execução de software específico
- Utilizador seleciona as VMs que precisa.

Type of each VM

Type	Sizes	Description
General purpose	B, Dsv3, Dv3, Dasv3, Dav3, DSv2, Dv2, Av2, DC	Balanced CPU-to-memory ratio. Ideal for testing and development, small to medium databases, and low to medium traffic web servers.
Compute optimized	Fsv2	High CPU-to-memory ratio. Good for medium traffic web servers, network appliances, batch processes, and application servers.
Memory optimized	Esv3, Ev3, Easv3, Eav3, Mv2, M, DSv2, Dv2	High memory-to-CPU ratio. Great for relational database servers, medium to large caches, and in-memory analytics.
Storage optimized	Lsv2	High disk throughput and IO ideal for Big Data, SQL, NoSQL databases, data warehousing and large transactional databases.
GPU	NC, NCv2, NCv3, ND, NDv2 (Preview), NV, NVv3	Specialized virtual machines targeted for heavy graphic rendering and video editing, as well as model training and inferencing (ND) with deep learning. Available with single or multiple GPUs.
High performance compute	HB, HC, H	Our fastest and most powerful CPU virtual machines with optional high-throughput network interfaces (RDMA).

VM and faults

Unplanned Hardware Maintenance Event

Azure prevê que o harware/componente associado a uma máquina está prestes a falhar. Para isto a plataforma emite um evento de manutenção de hardware não planejado (e usar **Live Migration**)

Live Migration

Pausa a VM → migra memória, arquivos abertos e conexões de rede → retoma a VM.
Porem, isto pode ser feito enquanto a VM está em execução, parando apenas por um curto periodo no fim para migrar o que foi alterado entretanto.

An Unexpected Downtime

O harware/componente associado a uma máquina falha inesperadamente (falhas de rede local, falhas de disco local ou outras falhas no nível do rack).
Quando detectadas, a cloud platform migra automaticamente a MV para uma máquina física saudável no mesmo datacenter.

Durante a recuperação, as MV's sofrem um tempo de inatividade (reinicialização) e, em alguns casos, dá-se perda da unidade temporária.

Contúdo:

- O SO anexado e os discos de dados são sempre preservados.
- As máquinas virtuais também podem sofrer inatividade no caso improvável de uma interrupção ou desastre que afete um datacenter inteiro ou até mesmo uma região inteira.

Planned Maintenance events

Atualizações periódicas feitas na cloud platform fundamental para melhorar: confiabilidade, desempenho, segurança, gerais da infraestrutura da plataforma.

As plataformas de nuvem tentam usar a **manutenção de preservação de VM** sempre que possivél e mantendo as VMs em execução (migrando-as para outras máquinas).

Select the availability options

Cada **zona de disponibilidade** tem uma fonte de energia, rede e refrigeração distintas. Um aplicativo deve ter replicação da app e data em diferentes zonas para tolerar a perda de um datacenter.

Fault domains

Grupo de harware básico que compartilha uma fonte de energia e um switch de rede comuns em um datacenter local.

Update domains

Grupo de harware básico que pode passar por manutenção ou ser reinicializado ao mesmo tempo.

Scale set

O Azure permitem criar e gerir um grupo de load balance VMs, onde o número de instâncias de VM pode aumentar ou diminuir automaticamente em resposta à demanda ou a um hóriario definido.

Nota: as VMs de um grupo podem ser implantadas em várias regiões (**availability** and **fault-tolerance**)

Availability set

Agrupamento de VMs em um datacenter que fornece **redundancy** e **availabitily**.

Com isto é possivél prevenir:

- falhas hardware
- interrupções de rede
- interrupções de energia

Disks and network options

Managed disk é um disco rígido virtual armazenado como blobs de páginas.

Tipos de disco

- Disco ultra
- SSD premium
- SSD padrão
- HDD padrão

O sistema provisiona por:

- disco IOPS
- largura de banda
- a rede para acesso remoto

Localmente, o servidor implementa o provisionamento SSD para dados armazenados no SDD do servidor, incluindo discos com cache, e locais/temporários.

VM's Vantagens vs. Desvantagens

Vantagens

Eficiência - uso eficiente de recursos e fornece isolamento.
Flexibilidade - recursos podem ser alocados conforme necessário.
Restaurar e recuperar - podem ser armazenadas como arquivo único que pode ser copiado noutra fonte.
Liberdade do sistema operacional - diferentes OSs convidados podem existir no mesmo hypervisor.
Desempenho e movimento - os hipervisores suportam movimentação de uma VM entre hosts em caso de degradação do desempenho na máq. host.

Desvantagens

Sobrecarga de desempenho - uma pilha de VM's inclui: o SO convidado, o hipervisor e potencialmente o SO host
Utilização eficiente de recursos - o uso de vários SO's no mesmo hipervisor duplica os recursos usados.

Containers

Os contêineres fornecem:

- virtualização no nível do OS.
- namespace privado
- interface de rede e endereço IP
- sistemas de arquivos
- portablidade entre máquinas
- gestão fácil de dependências

- Os contêineres compartilham o kernel do sistema host com outros contêineres.
- O código da app precisa usar a API do host OS.

Nota: antes o **chroot** - especificava uma dir como root, e isto permitia um app aceder a ficheiros de outra, ou I/O op.

Kernel namespaces

Os namespaces do kernel dividem os recursos do kernel (processos, usuários, pilhas de rede). Um processo visualiza apenas os recursos em seu namespace mas partilha o kernel sub-jacente com outros contêineres.

Implementação

System calls:

- **clone()** - cria um novo processo e um novo namespace
- **unshare()** - cria um novo namespace e anexa o processo atual para ele.
- **setns()** - permite ingressar em um namespace existente.

Namespaces

UTS - nome de host e de domínio do namespace.
Network - cópia da pilha de rede, com suas próprias rotas, regras de firewall e dispositivos de rede (cada Network Namespace tem IP próprio)
Mount - Na criação, a árvore do sistema de arquivos é copiada para um novo espaço, com todas as montagens anteriores visíveis.
PID - Processos em diferentes namespaces PID podem ter o mesmo ID de processo.
User - Um processo terá um conjunto distinto de UIDs, GIDs e capacidades
IPC - Cada namespace obtém seus próprios objetos IPC e filas de mensagens POSIX.

Cgroups

Mecanismo para aplicar limites de:

- recursos de hardware:
  - consumo de memória
  - CPU dos contêineres
- acesso a um processo ou uma coleção

Nota: Um contêiner pode ser redimensionado simplesmente alterando os limites de seu cgroup correspondente.

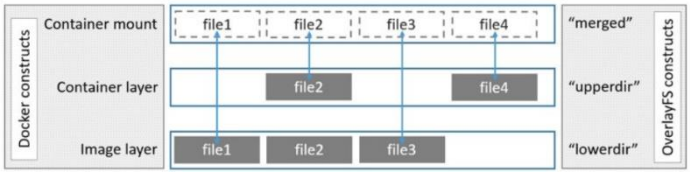
Implementação

A implementação de cgroups requer mexer no kernel: na inicialização, processo de criação e destruição. Todas as operações em cgroups são executadas usando operações em um VFS (sistema de arquivo virtual)

Copy-on-write File system

UnionFS - sistema de arquivos copy-on-write que fornece:

- visão unificada do sistema de arquivos
- combina os sistemas de arquivos empilhados
- as gravações de um contêiner não afetam as leituras noutro
- vários contêineres podem partilhar dados comuns
- cada camada é armazenada apenas uma vez



Docker

Construído sobre:

- namespaces do kernel → cgroups → unionFS

Cada contêiner:

- tem seu próprio conjunto de namespaces e Cgroups
- é isolado de outros através dos namespaces (não vê a lista de processos de outro)
- Cgroups permitem que o administrador isole os recursos usados por cada contêiner e seus filhos

Outros:

- executar daemon docker requer privilégios de root
- Docker fornece uma lista de permissões de recursos para usuários root num contêiner

Docker engine

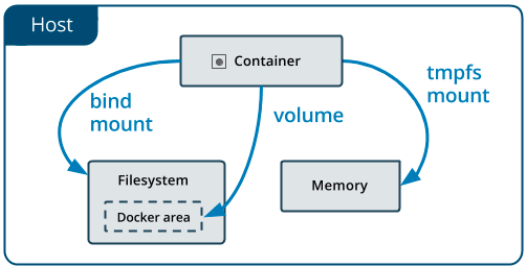
Daemon Docker - gere objetos Docker como imagens, contêineres, redes e volumes
Docker Register - armazena imagens do Docker
Image – Somente para leitura com instruções para criar um contêiner

Docker networking

A **Bridge network** conecta vários contêineres docker em execução no mesmo host docker. Uma rede pode ser criada usando o docker network create. O **overlay network driver** cria uma rede distribuída entre Deamon hosts.

Docker Storage

Os arquivos criados dentro de um contêiner são armazenados em uma camada de um contêiner que possa ser escrito e não persistente. O Docker tem duas opções para contêineres armazenarem arquivos na máquina host (para que os arquivos sejam persistidos mesmo após o container parar): volumes e montagens de ligação.



Os **Volumes** são armazenados em uma parte do sistema de arquivos host gerido pelo Docker (usados parapersistir dados no Docker).

As **Bind mounts** são armazenadas em qualquer lugar no sistema host (são arquivos ou diretórios importantes do sistema). Os processos não Docker no host Docker ou em um contêiner Docker podem modificá-los a qualquer momento.

As **tmpfs mounts** são armazenadas apenas na memória do sistema host e nunca são gravadas no sistema de arquivos do sistema host.

Docker compose

O Docker Compose permite definir e executar aplicativos Docker de vários contêineres.

Docker Swarm

O Swarm é um cluster de instâncias do Docker, e consiste em vários hosts do Docker que são executados no modo Swarm e atuam como managers e trabalhadores.

Funcionamento
Ao criar um serviço, defenimos:

- o nº réplicas
- a rede
- recursos de armazenamento disponíveis
- portas que o serviço expõe ao mundo externo

Por exemplo, se um Worker node ficar indisponível, o Docker agendará as tarefas desse nó em outros nós.

Nodes

Um nó é uma instância do Docker que participa do Swarm.

Nó manager:

- recebe definições de serviço
- dá tarefas para nós trabalhadores
- executam as funções de orquestração e gerenciamento de cluster necessárias para manter o estado do swarm.

Nós do trabalhador:

- recebem e executam tarefas despachadas dos nós do gerenciador.

Service

Defenições das tarefas a serem executadas nos manager or worker nodes.
Um serviço é a raiz primária da interação do cliente com o Swarm. Para serviços globais, o enxame executa uma tarefa para o serviço em cada nó disponível no cluster.

Um serviço:

- especifica qual imagem de contêiner usar
- especifica quais comandos executar
- tem uma entrada DNS no Swarm

Load Balancing

Swarm manager

- usa **ingress load balancing** para expor os serviços externamente ao swarm
- usa **internal load balancing** para distribuir requests entre serviços dentro do cluster (com base no nome DNS do serviço).
- Ambos atribuiem um PublishedPort ao serviço (ou nós podemos configurar um)

Componentes externos como load balancing em nuvem, podem acessar o serviço no PublishedPort de qualquer nó no cluster. Os nós no swarm roteiam conexões de entrada para uma instância de tarefa em execução.

Swarm services vs. standalone container

É possível modificar a configuração de um serviço (redes e volumes aos quais está conectado), sem a necessidade de reiniciar manualmente o serviço.

O Docker atualizará a configuração, interromperá as tarefas de serviço com a configuração desatualizada e criará novas que correspondam à configuração desejada.

Kuebernetes

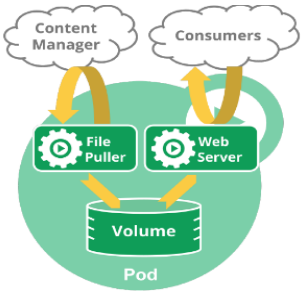
O Kubernetes é uma plataforma de código aberto para automatizar a implantação, dimensionamento e operações de contêineres de aplicativos.

Init containers

Executados e concluídos antes que os contêineres do aplicativo sejam iniciados. Se um init contêiner falhar, o pod será reinicializado.

Objects

Pod Resources



Encapsula:

- contêiner de uma app
- recursos de armazenamento
- IP de rede exclusivo
- opções que controlam como o(s) contêiner(es) deve(m) ser executado(s)

Networking

- Cada pod recebe um endereço IP exclusivo.
- Cada contêiner em um pod compartilha o namespace da rede, incluindo o endereço IP e as portas de rede.
- Os contêineres dentro de um pod se comunicam usando localhost.
- Quando os contêineres em um pod se comunicam com entidades fora do pod, eles devem coordenar como usam os recursos de rede compartilhados (como portas).

Storage

- Um pod pode especificar um conjunto de volumes de armazenamento compartilhado.
- Todos os contêineres no pod podem acessar os volumes compartilhados, permitindo que esses contêineres compartilhem dados.
- Os volumes também permitem que os dados persistentes em um pod sobrevivam caso um dos contêineres precise ser reiniciado.

Services

Um serviço é uma abstração que define um conjunto lógico de pods e uma política para acessá-los. Por exemplo, considere um back-end de processamento de imagem sem estado que está sendo executado com 3 réplicas.

As APIs do Kubernetes para descoberta de serviço permitem consultar Endpoints, que são atualizados sempre que o conjunto de pods em um serviço muda.

Tipos de serviços

ClusterIP: expõe o serviço em um IP interno do cluster (ServiceType padrão).
NodePort: expõe o Serviço no IP de cada node em uma porta estática.

LoadBalancer: expõe o serviço externamente usando o LoadBalancer de um cloud provider. Os serviços NodePort e ClusterIP, para os quais o balanceador de carga externo roteia, são criados automaticamente.
ExternalName: mapeia o serviço para o conteúdo do campo externalName

Volumes

É um diretório acessível aos containers num Pod.

- quando um container falha, ele será reiniciado, e os arquivos serão perdidos
- ao executar contêineres juntos em um pod, é necessário compartilhar arquivos eles
- um volume do Kubernetes tem o mesmo tempo de vida do pod que o inclui.
- quando um Pod deixa de existir, o volume também deixa de existir.
- alguns volumes são persistentes

Namespace

Namespaces fornecem um escopo para nomes. Os nomes dos recursos precisam ser exclusivos dentro de um namespace, mas não entre namespaces.

- o Kubernetes suporta vários clusters virtuais (namespaces) apoiados pelo mesmo cluster físico
- os namespaces destinam-se ao uso em ambientes com muitos usuários em vários projetos
- os nomes dos recursos precisam ser exclusivos dentro de um namespace, mas não entre namespaces.
- os namespaces não podem ser aninhados uns dentro dos outros e cada recurso do Kubernetes só pode estar em um namespace.

Kubernetes controllers

Deployment

Um deployment fornece atualizações declarativas para pods e ReplicaSets. O deployment maneger altera o estado real → para o estado desejado

ReplicaSet

Um ReplicaSet garante que um nº especificado de réplicas de pod esteja em execução a qualquer momento. Um deployment gerencia ReplicaSets.

DeamonSet

Um DaemonSet garante que todos (ou alguns) os nodes executem uma cópia de um Pod.
Nota: A exclusão de um DaemonSet limpará os pods que ele criou.

StatefulSet

StatefulSet é usado para gerir stateful apps, e é útil para apps que exigem um ou mais dos seguintes itens:

- Persistent, unique network identifiers.
- Persistent, persistent storage.
- Ordered, graceful deployment and scaling.
- Ordered, automated rolling updates.

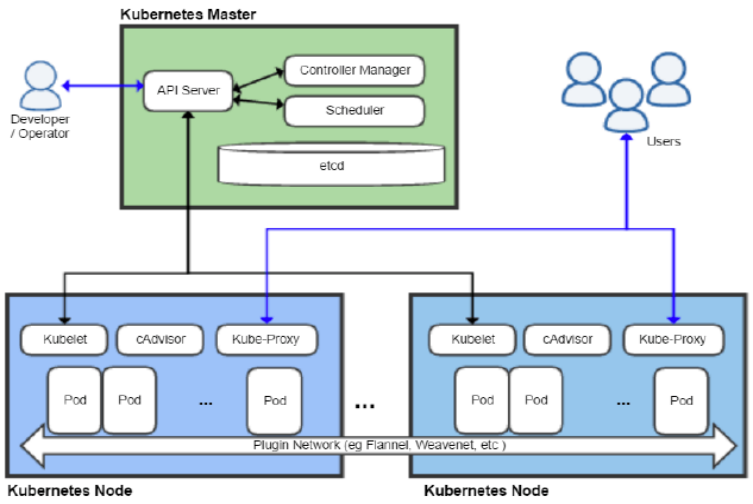
Job

Um trabalho cria um ou mais pods e garante que um número especificado deles seja encerrado com sucesso

Kubernetes Control Plane

O Kubernetes Control Plane faz com que o estado atual do cluster corresponda ao estado desejado

- inicia ou reinicia contêineres
- dimensiona o nº de réplicas de uma app



O Kubernetes master é executado em um único nó de um cluster. E um nó do Kubernetes é:

- kubelet → que se comunica com o Kubernetes Master
- kube-proxy → um proxy de rede que reflete os serviços de rede Kubernetes em cada nó.

Kubernetes aparenta ter melhor escalabilidade do que o Swarm.

Multi cloud computing

Cloud computing platforms podem ter problemas de confiabilidade (reliability) e segurança, porque uma plataforma de nuvem pode falhar ou ter uma violação de segurança. E podemos também ficar presos nos serviços dessa plataforma.

OpenStack

Usado para gerir uma plataforma de nuvem. Controla grandes pools de recursos de computação, armazenamento e rede em um ou vários datacenters.

OpenStack services

Nova: Serviço para provisionar instâncias de computação (VM's; contêineres)

Zun: Fornece uma API para executar contêineres de aplicativos sem a necessidade de gerenciar servidores ou clusters

Cinder: Gere volumes usados pelas Nova VM's, contêineres Zun, e fornece alta disponibilidade e tolerância a falhas, contando com a replicação.

Swift: Armazenamento de objeto/blob altamente disponível, distribuído e eventualmente consistente (DHT, replicação assíncrona)

Vantagens

- Evite vendor lock-in.
- Fornecer maior confiabilidade e segurança combinando recursos em várias plataformas.
- Construir sobre os melhores serviços em diferentes plataformas.

O OpenStack permite dar deploy e executar serviços em vários sites – existe um serviço de orquestração responsável por gerir o deploy da multi-cloud.

Service-level multi-cloud solutions

System/framework that exposes a unique service API for the clients, and then uses the services of several clouds to implement such service API.

Hybrid cloud

Isso combina duas infraestruturas de nuvem diferentes, 1 privada e pelo menos 1 pública.

Vantagens

- Apps e/ou workloads sensiveis com desempenho constante e requisitos de capacidade podem ser executados numa private cloud
- Apps e/ou workloads menos sensiveis podem ser executados na public cloud
- Aproveitar a capacidade da public cloud para acomodar um aumento na demanda por um aplicativo de nuvem privada (**cloud bursting**)

Types of hybrids

Monocloud

Hybrid cloud apenas comum unico cloud provider
Os ambientes são interligados para formar um único ambiente híbrido, gerido apartir da cloud publica com as mesmas ferramentas que a infrastrutura.

Multicloud

Open standards-based stack pode ser deployed em qualquer infraestrutura de nuvem pública.

Multicloud architecture dá a uma organização a flexibilidade de mover workloads de fornecedor para fornecedor e de ambiente para ambiente conforme necessário e para trocar serviços de nuvem e fornecedores por qualquer motivo.

Open standards

O objetivo dos padrões abertos é permitir consistência e repetibilidade na abordagem. O suporte à nuvem híbrida inclui Kubernetes, OpenStack e Cloud Foundry

Integration

A integração entre aplicativos e dados (dentro e fora do local) é fundamental para garantir que os componentes do ecossistema híbrido funcionem juntos de forma rápida e confiável.

Management

O gerenciamento é relativamente direto porque, com um único fornecedor, pode-se usar as mesmas ferramentas para gerenciar ou provisionar toda a infraestrutura.
O Kubernetes pode ajudar com tarefas de gerenciamento, como dimensionar aplicativos em contêineres, lançar novas versões de aplicativos e fornecer monitoramento, registro, depuração etc.

- Monocloud manter o estado da tua app
- Multicloud é mais dificil de manter o estado da tua app

Storage

Um modelo de armazenamento em nuvem híbrida oferece a escolha de quais dados armazenar em qual nuvem.

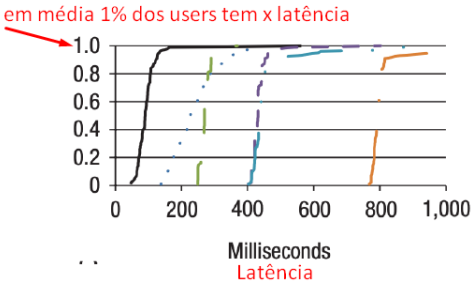
Edge computing

As computations nem sempre vão para a nuvem. Aqui temos recursos de armazenamento mais próximos dos clientes.

Cloudlet pode executar código arbitrário exatamente como na computação em nuvem (encapsulado em uma VM ou em um contêiner mais leve para isolamento, segurança, gerenciamento de recursos e medição)

Benefits

- Serviços em nuvem altamente responsivos
- Escalabilidade via análise de borda/pré-processamento
- Aplicação da política de privacidade
- Mascarar interrupções na nuvem



Edge analytics

Ajude os aplicativos de IoT propagando apenas as informações necessárias e importantes. para a nuvem (por exemplo, a temperatura de um sensor nem sempre precisa ir para a nuvem, apenas quando muda)

Masking cloud outages

Quando a nuvem falha, ela pode mascarar sua falha.
À medida que a dependência da cloud cresce, aumenta também a vulnerabilidade a interrupções na cloud

- Garantir que os dados necessários estejam disponíveis
- Ser capaz de funcionar enquanto estiver desconectado.

Networking 101

Communication between party's

LAN - Os hosts devem se comunicar por meio de endereços IP

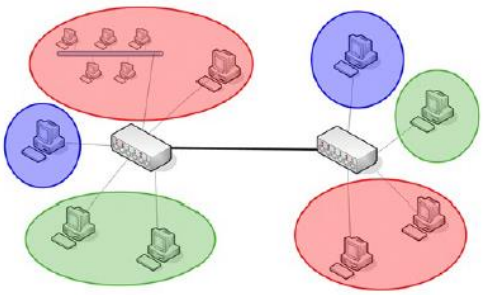
ARP - Preparar um pacote ARP (próprio IP, e o IP do outro) → Coloque o pacote em um quadro (o endereço de origem é seu próprio MAC e o endereço de destino é todos uns) → Aguarde a chegada de uma resposta

- Os sinais representam os bits 0 e 1
- Os bits são compactados em quadros
- Os quadros transportam pacotes IP
- Os pacotes IP transportam segmentos TCP ou datagramas UDP
- Geralmente carregam "dados do usuário"

Em uma rede, cada host tem um sufixo IP diferente e pode se comunicar diretamente com outros hosts nessa rede. Para se comunicar com hosts em uma rede diferente, é necessário um dispositivo roteador.

IP(v4)

10.11.12.13/8	prefix: 10	suffix: 11.12.13
172.16.1.234/16	prefix: 172.16	suffix: 1.234
192.168.1.23/24	prefix: 192.168.1	suffix: 23



Red net	prefix: 192.168.1
	suffixes: 1 - 8
Blue net	prefix: 192.168.2
	suffixes: 1 - 2
Green net	prefix: 172.16
	suffixes: 1.1, 1.2, 1.3

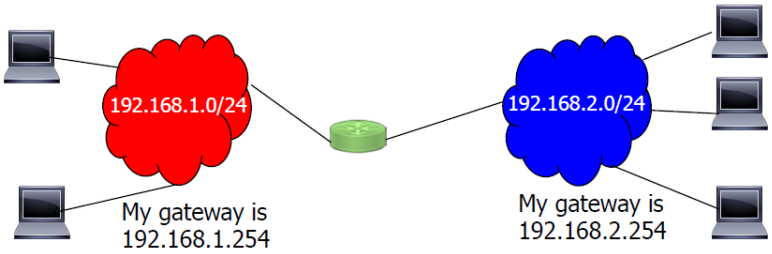
Switches

Permite que as comunicações fluam entre as partes e é um dispositivo "passivo" – não inicia a comunicação

Quando uma porta recebe um quadro, o switch procura os MACs de origem e destino e:

- Se este é o primeiro quadro que ele vê vindo daquele CN, o switch "memoriza" que o CN remetente é alcançável "através daquela porta";
- Se ele sabe qual porta deve ser usada para alcançar o CN de destino, ele encaminha o quadro por essa porta;
- Se não souber qual porta deve ser usada para alcançar o CN de destino, ele transmite o quadro para todos

Passo a passo, ele constrói um FDB (banco de dados de encaminhamento) de portas e MACs acessíveis por essas portas



Quando um host com endereço IP1 quiser se comunicar com um host com IP2, em uma rede diferente, ele irá:

- Prepara um pacote IP com dados e endereço de origem IP1 e endereço de destino IP2;
- Envia o pacote para o roteador (pode exigir um ARP primeiro);
- O roteador recebe na interface X o quadro vindo do IP1, extrai o pacote IP do quadro e o copia para um novo quadro que envia via interface Y para o host IP2 (pode exigir um ARP primeiro).

VLAN

É uma fonte de muitos problemas:

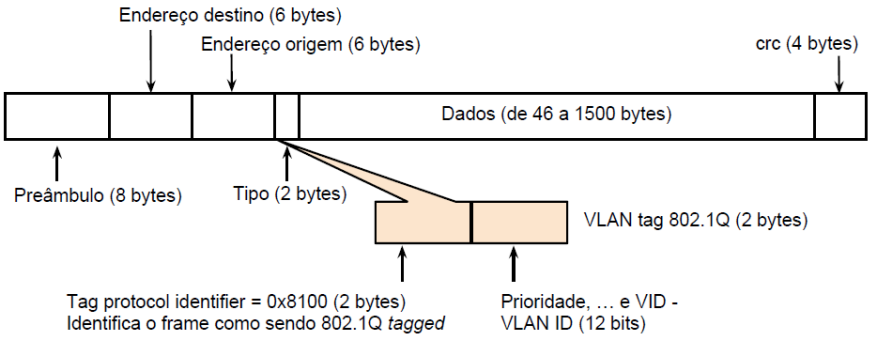
- **Unsecure**: o tráfego pode ser espionado, os IPs alterados e assim todos podem acessar tudo.
- **Load**: As transmissões "roubam" bandwidth valiosa.

Virtual networks in practice

As redes virtuais são usadas para dois objetivos principais:
• Permitir que máquinas/serviços em execução em locais diferentes pertençam à mesma rede.
• Proteger a comunicação entre dispositivos em uma rede virtual.

Switches precisam de tabelas de configuração
• Dizer quais VLANs são acessíveis por meio de quais interfaces

Alterando o cabeçalho Ethernet
• Adicionando um campo para uma tag VLAN
• Implementado nas pontes/switches
• … mas ainda pode interoperar com placas Ethernet antigas



Network in cloud platforms

As plataformas de nuvem tendem a usar internamente protocolos proprietários.

Rationale: ambientes controlados e requisitos específicos são melhor suportados por soluções específicas (do que por padrões projetados para uso geral).

Communication between Azure resources

Os recursos do Azure se comunicam com segurança entre si de uma das seguintes maneiras:

Through a virtual network: Pode-se implantar VMs e vários outros tipos de recursos do Azure em uma rede virtual, como Ambientes de Serviço de Aplicativo do Azure, Serviço de Kubernetes do Azure (AKS) e Conjuntos de Dimensionamento de Máquinas Virtuais do Azure.
Through a virtual network service endpoint: O espaço de endereço privado de rede virtual e a identidade de sua rede virtual para recursos de serviço do Azure, como contas de armazenamento do Azure e bancos de dados SQL do Azure, por meio de uma conexão direta, podem ser extendidos.
Through VNet Peering: Pode-se conectar redes virtuais umas às outras, permitindo que recursos em qualquer uma das redes virtuais se comuniquem entre si, usando emparelhamento de rede virtual. As redes virtuais que se conectam podem estar nas mesmas ou diferentes regiões do Azure.