Resumos ASCN

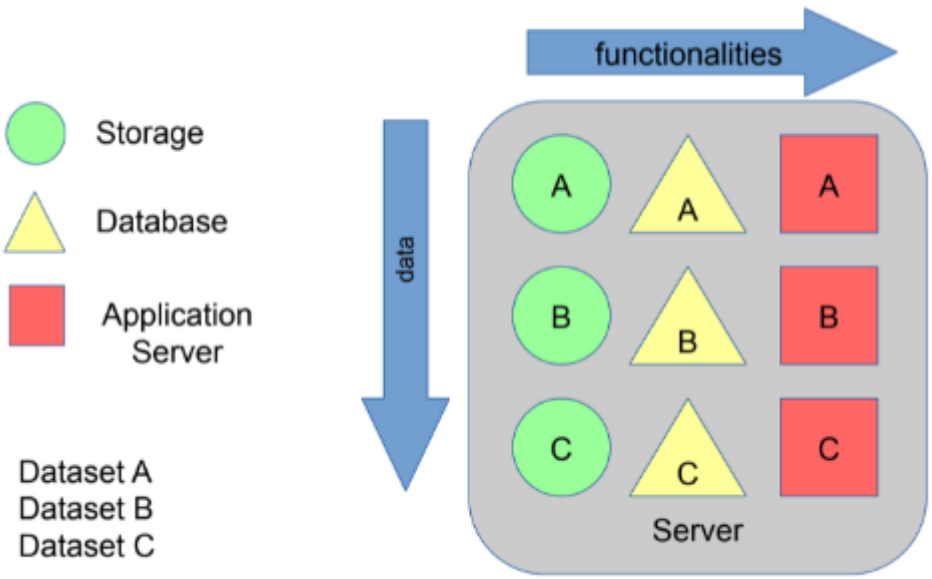
Distributed Aplications

Porquê aplicações distribuídas?

- modularidade, separação de responsabilidades
- performance
- dependências

Sistema monolítico

Multiplos serviços para o mesmo cliente estão no mesmo servidor



Sistema distribuído

Objetivos de distribuição:

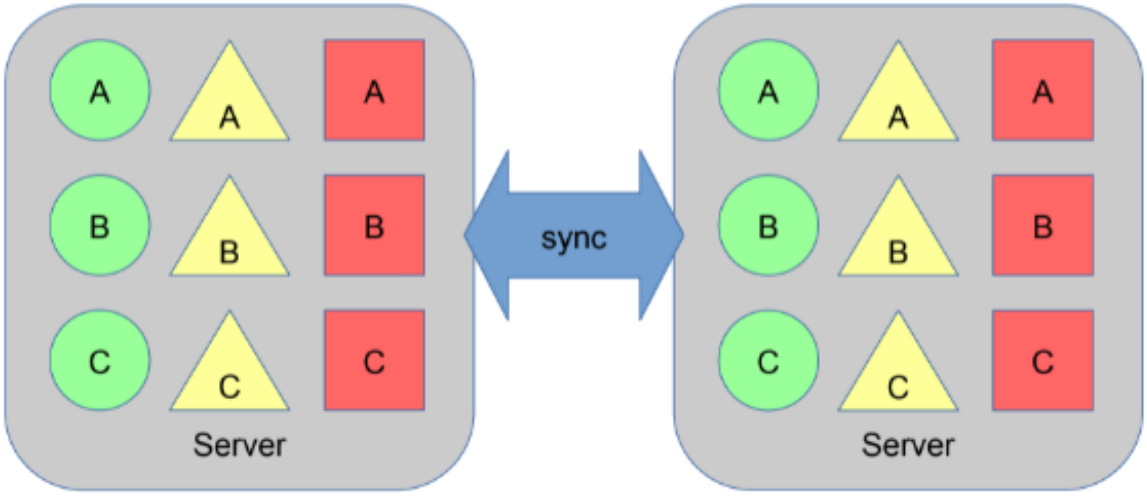
- replicação de componentes
- partitioning
- service-orientation

Estas técnicas procuram melhorar a escalabilidade de um serviço/aplicação

Replicação

Várias cópias dos mesmos dados e funcionalidades

Resiliência de endereços e escalabilidade

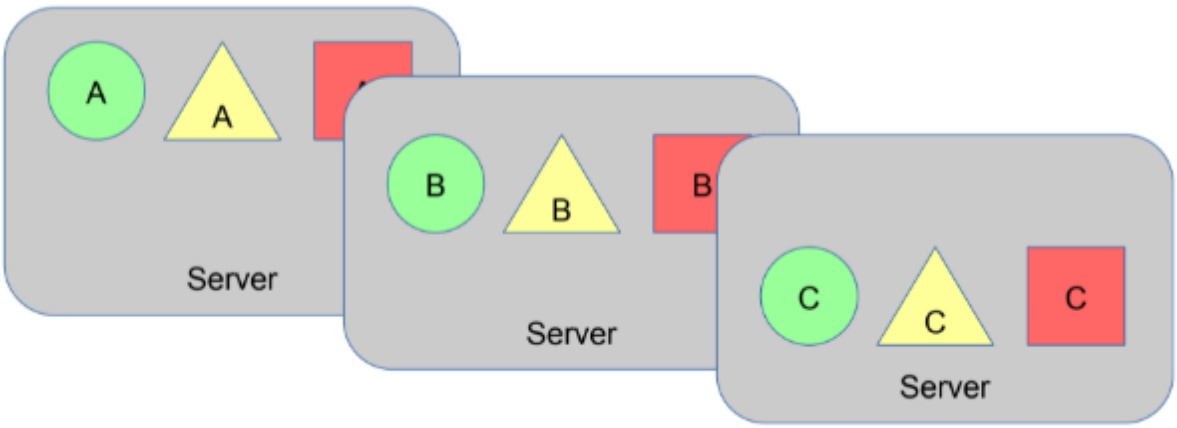


Partitioning

Um servidor é dividido horizontalmente (Sharding)

Endereços são escaláveis

Pode ser usado em computação, dados...

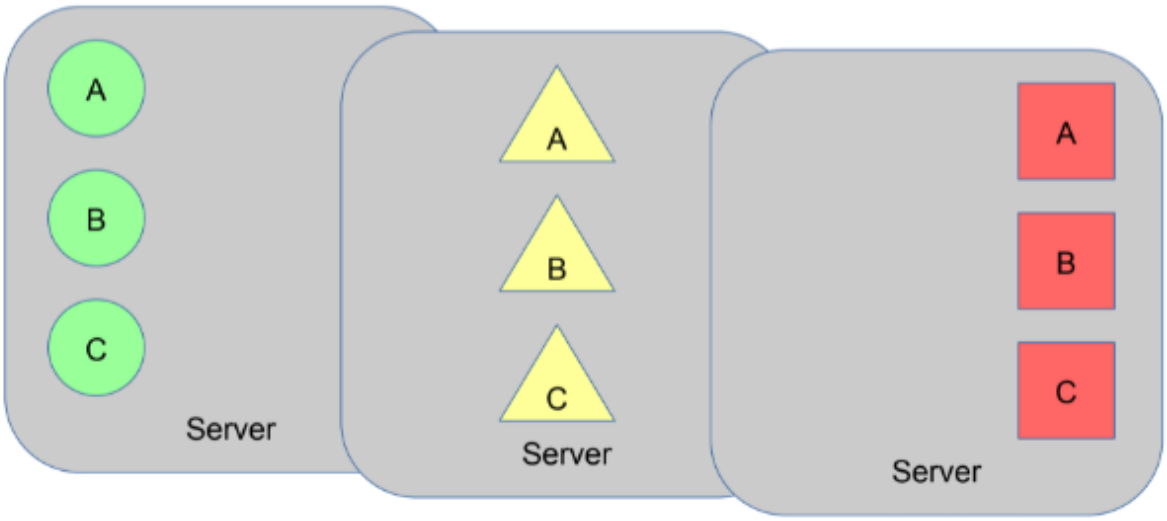


Service-Oriented Architecture (SOA)

Um servidor é separado verticalmente

Endereços escaláveis e modularidade

Exemplo: micro-services



Microservices

- Cada serviço implementa uma funcionalidade específica
- Serviços podem escalar independemente
- A decomposição em microserviços pode ser complexa (quão micro é micro?)
- Consistência
- Deployment e testing complexos

Arquiteturas Distribuídas

Client-Server

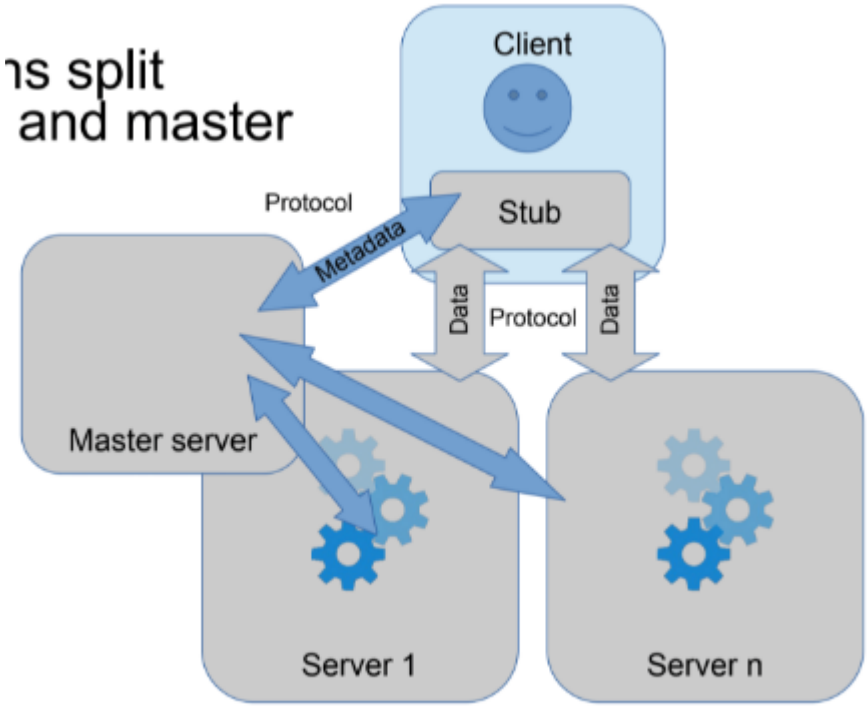
- Funcionalidades e dados estão no servidor
- Um stub corre no cliente
- O stub é parte do package de software do servidor

Proxy-Server

- Múltiplos servidores podem ser usados transparentemente
- O proxy é um bottleneck de performance e availability

Master-Server

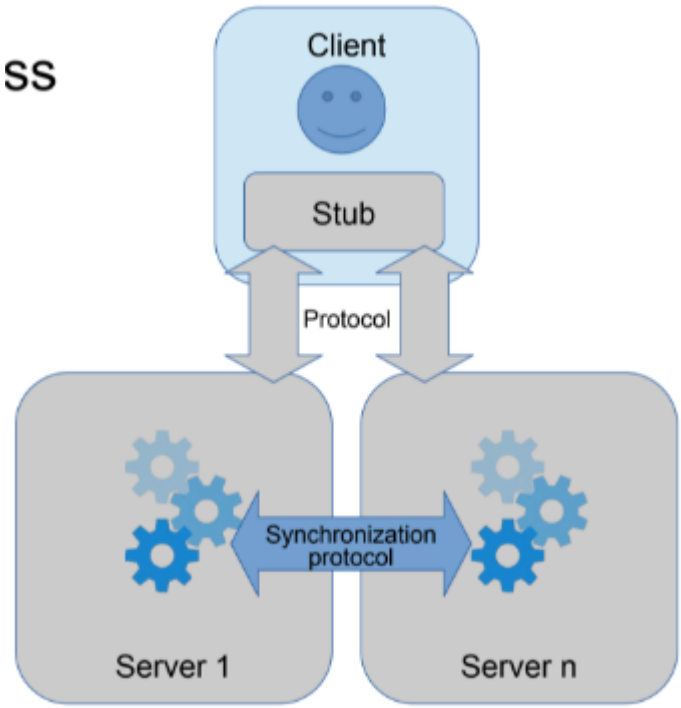
- As funções do proxy estão divididas pelo stub e servidor master
- Escalabilidade



12

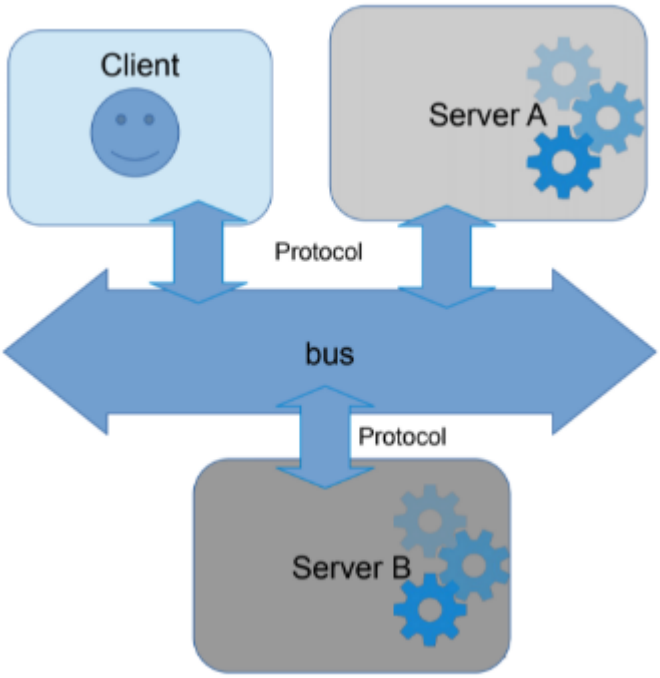
Server group

- Todos os servidores podem processar pedidos
- Pode ser necessária coordenação
- Resiliência



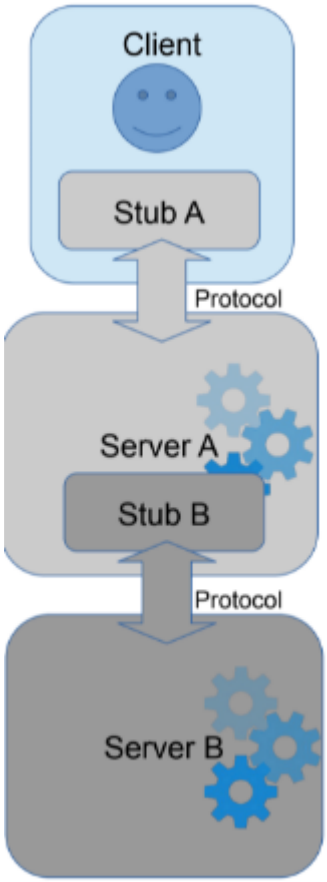
Bus

- O bus reencaminha mensagens
- Os participantes publicam e consomem mensagens para/do bus
- Separa os produtores de consumidores
- Flexibilidade!



Multi-tier

- Cada servidor funciona como um cliente para o próximo tier
- permite deployment independente e escalabilidade de diferentes funcionalidades



State in multi-tier

- Estado persistente é mais complexo de replicar e de fazer shard
- Computação é mais fácil de replicar e fazer shard
- Não há estado nas camadas superiores
- estado transiente/cached nas camadas do meio
- estado persistente nas camadas mais baixas

Provisionamento de Sistemas

Provisionamento é a ação de fornecer algo para utilização

Deployment

Processo de instalação ou upgrade de um serviço/aplucação num servidor

Gestão de configuração

Uma solução para gerir mudanças sistemáticas do sistema, mantendo a integridade durante o seu ciclo de vida

Receitas/Reutilização/Automação

Automatizar tarefas utilizando um conjunto de diretivas expressas numa determinada linguagem

Ansible

- Inventário - grupo de alvos de deployment (hosts)
- Module - Unidade de trabalho reutilizável
- Task - Combinação de um módulo e argumentos para criar uma ação
- Playbook - Descriver políticas para sistemas remotos aplicarem (conjunto de tasks)
- Handlers - Uma task especial que responde a uma dada notificação
- Role - Componente reutilizável que encapsula variáveis, tarefas e handlers (configuráveis)

execução da receita é agentless e feita através de SSH ou localmente

Computação na Cloud

Os serviços da cloud podem ser divididos em 3 níveis principais de abstração:

Infrastructure-as-a-Service (IaaS):

- fornece recursos de hardware virtuais tais como computação, armazenamento e rede
- recursos são alocados on demand e num modo pay-per-use.
- exemplos: Amazon EC2, Google Compute Engine...

Platform-as-a-Service (PaaS):

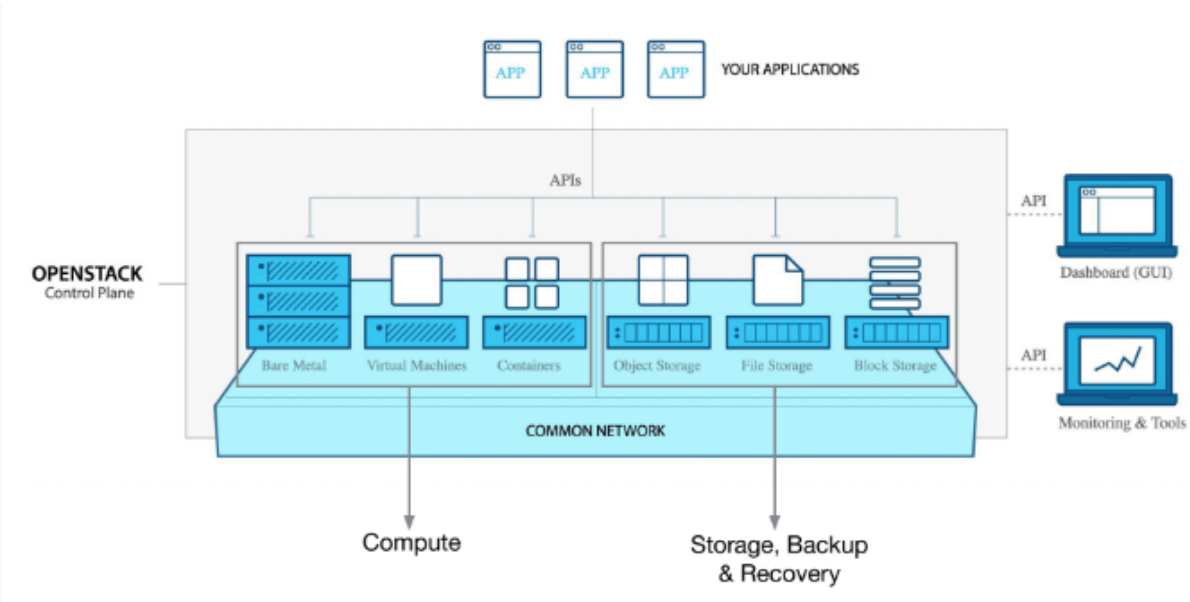
- oferece um encapsulamento de uma abstração de um ambiente de desenvolvimento que pode ser utilizado para desenvolvimento, deployment e executar aplicações
- Ex: Google App Engine

Software-as-a-Service (SaaS):

- Pode ser aplicações completas ou software genérico como bases de dados
- oferecido como um serviço e acessível a partir de um serviço web ou web browser
- Ex: Gmail

OpenStack

- Software open source para criar redes privadas e públicas
- Controla grandes pools de recursos computacionais, armazenamento e redes dentro do datacenter
- Gestão a partir de uma dashboard ou OpenStack API



OpenStack - Cinder

Serve para Block storage. Virtualiza a gestão de dispositivos de block storage. Fornece uma API para interagir com esses recursos.

OpenStack - Swift

Armazenamento de objectos/blobs com grande availability,distribuido e consistente. Ideal para armazenar dados não estruturados (imagens VM, videos) que podem crescer sem limites

Ephemeral Storage vs Cinder vs Swift

	Ephemeral storage	Block storage	Object storage
Used to...	Run operating system and scratch space	Add additional persistent storage to a virtual machine (VM)	Store data, including VM images
Accessed through...	A file system	A block device that can be partitioned, formatted and mounted (such as, /dev/vdc)	REST API
Accessible from...	Within a VM	Within a VM	Anywhere
Managed by...	OpenStack Compute (Nova)	OpenStack Block Storage (Cinder)	OpenStack Object Storage (Swift)
Persists until...	VM is terminated	Deleted by user	Deleted by user
Sizing determined by...	Administrator configures size settings, known as *flavors*	Specified by user in initial request	Amount of available physical storage
Example of typical usage...	10 GB first disk, 30GB second disk	1 TB disk	10s of TBs of dataset storage

OpenStack - Neutron

Software-Defined-Network (SDN) - focada em entregar networking-as-a-service (NaaS) em ambientes de computação virtuais. Abstrai a topologia da rede e endereços

OpenStack - Telemetry

Serviço de monitorização: coleciona metricas de monitorização para recursos físicos e virtuais. Oferece persistência das métricas para retrieval e análise. Oferece a configuração de ações quando certas regras se verificam.

De IaaS para PaaS

- De alocação gerida e provisionamento de recursos para gestão de infraestrutura
- Os recursos tornam-se transparentes
- Foco na aplicação, que é o item de deployment
- A interface é um ambiente de programação, com APIs para serviços IaaS/SaaS
- O utilizador pode focar na funcionalidade a dar deploy em vez dos recursos necessários para a suportar

Em suma, em PaaS, o utilizador escreve uma aplicação, dá deploy e a plataforma assegura-se de que a aplicação corre e tem acesso aos recursos necessários. Ex: Google App Engine

De PaaS para Saas

- São oferecidos serviços específicos
- Componentes de software geridos e mantidos que exportam as suas APIs
- Ex: sistemas de gestão de bases de dados

- Não há item de deployment

Em suma, o utilizador escreve uma aplicação que precisa de uma DB, então utiliza a API da plataforma que oferece um serviço de bases de dados.

Conveniência

- de IaaS: evitar custos upfront em gestão de infraestrutura e hardware. Deploy mais fácil de aplicações legacy
- para PaaS: foco no desenvolvimento da aplicação e os seus requisitos. Ferramentas poderosas de development, deployment, debugging e benchmarking já disponíveis
- para SaaS: oferece componentes já existentes (bases de dados, web/ servidores de aplicações

Velocidade

- de IaaS: infraestrutura já está instalada e configurada
- para PaaS: framework de development já instalada e configurada
- para SaaS: rápida integração de diferentes soluções de software na cloud

Elasticidade

- de IaaS: ilusão de recursos virtualmente infinitos. Aumenta e diminui o poder computacional, armazenamento e outros recursos de acordo com a procura (manualmente ou com ferramentas de terceiros).
- para PaaS e SaaS: não há necessidade de gerir manualmente a elasticidade

Desvantagens

Perda de controlo:

- De IaaS: não há controlo sobre o hardware específico e software de virtualização. Não há possibilidade de 'fine tuning' e optimização da infraestrutura
- para PaaS: não há controlo do hardware específico e da plataforma PaaS. Gestão, 'fine tuning' e monitorização estão limitados às ferramentas oferecidas pela plataforma
- para SaaS: aplicações web de terceiros

Segurança

- IaaS, PaaS e SaaS: é tão seguro quanto o provider, qualquer vulnerabilidade do providere é uma vulnerabilidade da aplicação. Correções a vulnerabilidades têm de ser feitas pelo provider. Se o provider falhar, a aplicação falha e está fora do controlo do dono. Privacidade de dados em infraestruturas de terceiros

DataCenters

ALS à porta dos servidores do SWAP - Circa, Setembro, 2021 (Colorized)

Virtualização

Técnica que permite a criação de software-based recursos ou dispositivos virtuais que, na prática, é uma abstração no topo de hardware e software existente

Vantagens:

Heterogeneidade

- Os recursos virtuais podem ser fornecidos no topo de recursos físicos diferentes
- Um recurso virtual pode suportar diferentes aplicações/sistemas operativos, utilizando o mesmo hardware

Transparência

- Interação do utilizador com recursos virtuais é similar à interação com recursos físicos

Isolation

- Recursos virtuais estão isolados do hardware/software em baixo, em termos de: segurança, performance e falhas

Optimização de recursos

- Recursos virtuais podem ser configurados para suportar mais clientes/aplicações: consolidação de servidores, custos mais baixos

Gestão simplificada

- Gerir recursos virtuais é mais simples que gerir recursos físicos

Desvantagens

Performance

- A virtualização de recursos inclui uma penalização de performance

Overprovisioning

- Deploy de mais recursos virtuais do que os disponíveis fisicamente pode resultar na degradação de performance

Segurança

- Se a isolação não for bem feita, ou um utilizador malicioso tem acesso aos recursos físicos, a segurança pode ser comprometida

Dependência

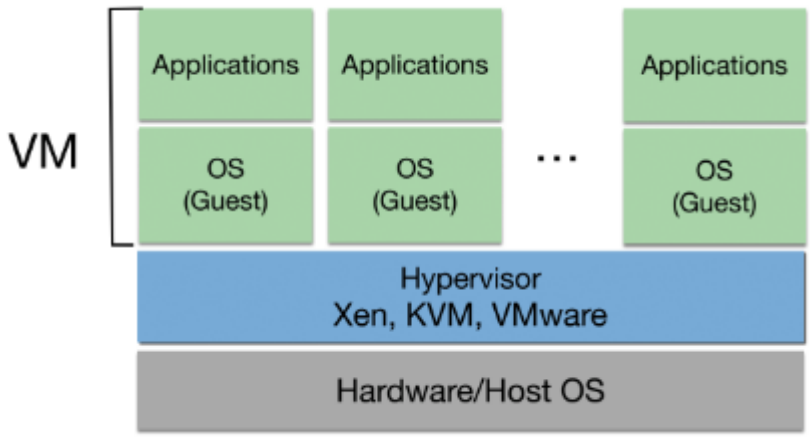
- A falha de recursos físicos pode resultar na falha da múltiplos recursos virtuais

Máquinas virtuais

- Mudar uma aplicação para funcionar em deferentes sistemas operativos é complexo e custoso
- VMs permitem correr diferentes sistemas operativos no topo do mesmo servidor físico
- VMs permitem a consolidação de recursos do servidor

Arquitetura:

As operações são interceptadas pelo hypervisor e só depois é que são executadas no hardware físico.



Hypervisor

Controla as interações low-level entre as VMs e o OS/hardware do host

Modos de virtualização

Full virtualization

- VM OS não é modificado uma vez que o hypervisor emula todos os dispositivos de hardware no ambiente virtual
- Vantagens: não haver modificações no OS guest significa um maior número de OS suportados e migração mais fácil / maior portabilidade de VMs
- Desvantagens: Todas as operações são processadas por duas camadas (VM e hypervisor), resultando em pior performance de I/O e CPU

Paravirtualization

- Precisa de modificações no OS guest
- As operações mais custosas são feitas diretamente no host nativo em vez do ambiente virtual
- Vantagens: Melhor performance
- Desvantagens: OS guest tem de ser modificado, o que piora a sua manutenção e portabilidade

Tipos de virtualização

Tipo 1 - hypervisor bare metal

Hypervisor é o OS - Melhor performance

Tipo 2 - hosted hypervisor

Hypervisor é um componente de software que corre no topo de um OS genérico - melhor portabilidade

Containers

Ambiente virtual leve que agrupa e isola um conjunto de processos e recursos do hosts e quaisquer outros containers

Porquê containers?

- Correr diferentes versões de software num OS/kernel partilhado
- Portabilidade/migração entre servidores
- Packaging de software e dependências mais fácil

Linux Containers

- mais leves que VMs, não requerem virtualização completa dos recursos
- O engine isola e configura recursos
- são criados compartimentos no sistema Host para cada container (em termos de CPU e I/O)
- Containers são isolados uns dos outros

Linux Containers - Building Blocks

- Namespaces (isolation) - componente do kernel Linux que faz com que um recurso do host global apareça como um recurso dedicado para um grupo de processos a correr no mesmo namespace
- Permite partilhar os recursos do host entre diferentes containers sem quaisquer conflitos
- Grupos de controlo (resource management) : permite alocar recursos entre grupos de processos
- Em outras palavras, este mecanismo limita a quantidade de recursos utilizados por cada container, prioritizando um container sobre os outros
- SELinux (segurança) : fornece maior segurança que namespaces para que um container não seja capaz de comprometer o sistema host e outros containers
- Enforça controlo de acesso e políticas de segurança

Linux Containers - Tipos

- OS Containers - correm múltiplos processos e simulam um sistema operativo leve
- Application Containers (ex: Docker) - focado em deployment de aplicações. Cada aplicação é visto como um processo independente

Containers vs VMs

- VMs são mais úteis quando virtualização completa é necessária
- Containers são úteis para gerir diferentes libraries/aplicações nos hosts

Vantagens dos Containers sobre as VMs

- testes/provisioning/migration mais rápidos
- melhor utilização de recursos (mais leve) e performance
- Pode ser deployed em ambos servidores físicos e virtuais

Desvantagens dos Containers sobre as VMs

- isolação/segurança mais fraca (OS/Kernel são partilhados)
- Menor flexibilidade a correr em diferentes OSs

Armazenamento

- É muitas vezes desvalorizado em sistemas e infraestruturas
- Performance é a chave!
- Persistência e disponibilidade de dados

Tipos de Armazenamento - Archival

- Write-once data
- Capacidade de escrita/leitura favorecida em vez de latência
- Workloads sequenciais
- Grandes quantidades de dados devem ser lidos/escritos com eficiência

Tipos de Armazenamento - Backup

- Data é atualizada esporadicamente
- Capacidade de escrita/leitura favorecida em vez de latência
- Sequential workloads
- Atualizações aos dados têm de ser consideradas

Tipos de Armazenamento - Armazenamento Primário (não só RAM)

- armazenamento de suporte para bases de dados, análises...
- Grande capacidade de leitura/escrita e baixa latência são desejados
- workloads sequenciais e random
- Os dados são atualizados frequentemente

Meios de armazenamento

- Tape - archival storage
- HDD - archival, backup e storage primário
- SSD (NVMe) - Primary storage
- Persistent Memory - primary storage
- RAM - primary storage

Storage Interfaces

- Block device (dados são geridos como blocos)
- Sistema de ficheiros (hierarquia de ficheiros)
- object storage (dados são geridos como objetos)

Armazenamento Distribuído - Datacenters

- grande escala
- stable churn
- sem ponto único de falha
- optimizado para grande performance

Armazenamento altamente distribuído - P2P

- muito grande-escala - até milhões de nodos
- Grande churn
- não há ponto único de falha

Storage features - disponibilidade de dados

- RAID
- Replication
- Erasure-Codes

Storage features - optimizações de performance

- Localidade de dados - ter servidores à beira do local de computação
- Caching - evitar escritas/leituras no disco e nodo de armazenamento

Storage features - eficiência de espaço

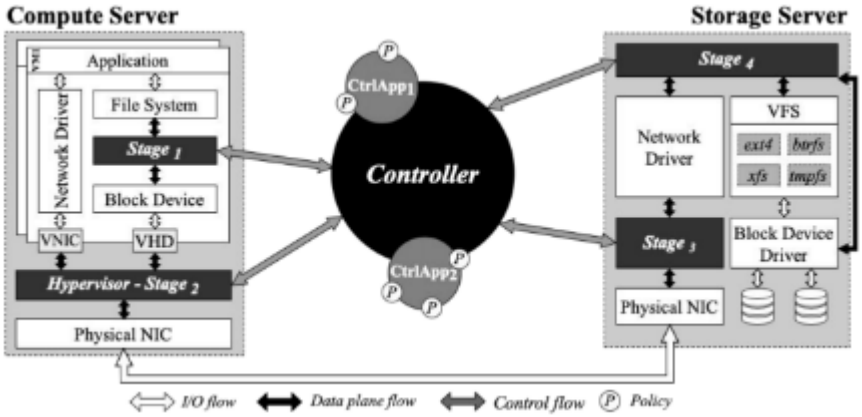
- Compressão - reduz redundância no conteúdo de ficheiros ou blocos
- Deduplication - elimina cópias redundantes do mesmo ficheiro / bloco

Storage features - segurança

- Encriptação de dados
- Controlo de acesso

Software-defined Storage

- segue os príncipios de software-defined networks
- fluxo de I/O (data plane) é separado do fluxo de controlo (control plane)
- O controlo é logicamento centralizado



- Data plane: programável e extensível, segue uma arquitetura de camadas (organizado por stages). Cada stage é responsável por pedidos nesse caminho de I/O. Stage inclui funcionalidades como caching, compressão, encriptação
- Control plane: Distribuído e dependente; Assegura a correta configuração dos stages do data plane. Assegura políticas ao longo do fluxo de I/O (qualitfy of service, ex:fairness, prioritization, load balancing; e transformações, ex: encriptação, deduplication, compression)

Monitoring

Um monitor observa a atividade do sistema

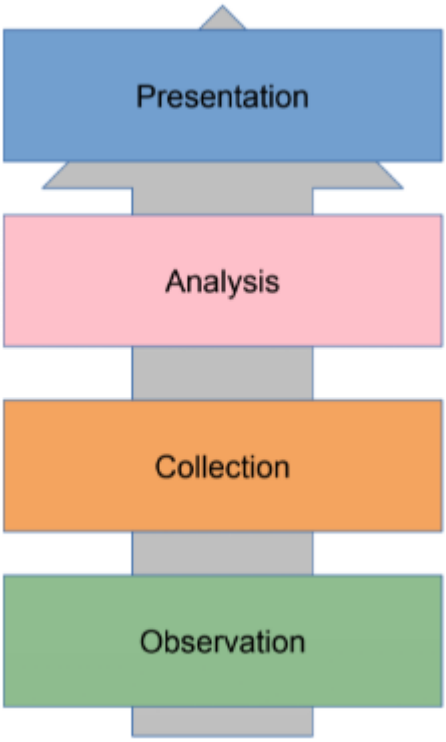
- um sistema contém recursos físicos e lógicos com estados
- estados mudam com eventos
- Trace é um registo de eventos: time stamp, variable detail....
- Domain é o conjunto de atividades observadas
- Um monitor impõe um overhead, mudando a atividade observada

Classificação de monitores

- Event-driven vs sampling: o que desencadeia a observação
- On-line vs batch: quando a observação está disponível
- Hardware vs software

- Centralizado vs distribuído

Arquitetura de um monitor



- Presentation - produz relatórios, alarmes...
- Análysis - camadas de filtros, relaciona e resume dados
- Collection - coleciona e normaliza os dados
- Observation - observação de eventos nos sistemas

Observation

- Observação passiva ou espia: network sniffer
- Instrumentação: contadores construídos no sistema (hardware e OS); geração de relatórios
- Probing (sondar) com pedidos adicionais: ping

Collection

- Push data vs pull data
- Reliability e persistência
- Sincronização do tempo em sistemas distribuídos

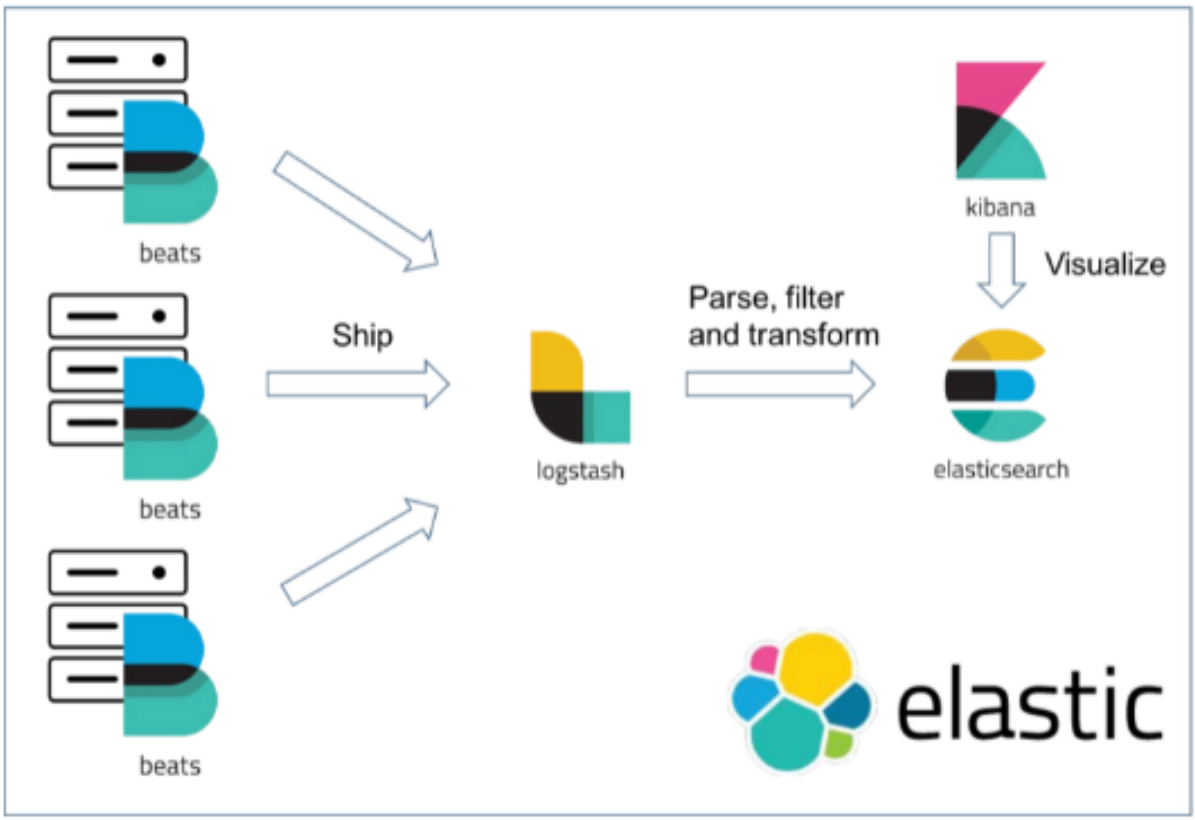
Analysis

- tarefa de processar dados
- Big data em grandes infraestruturas
- ex: elasticsearch

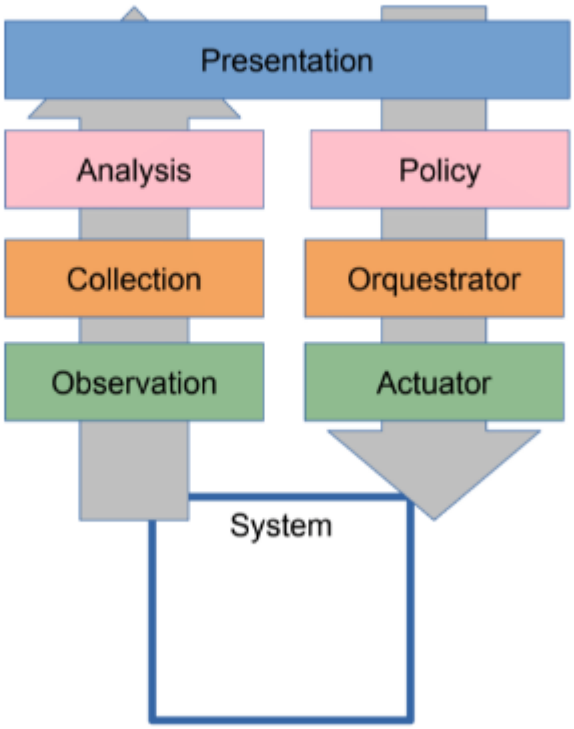
Presentation

- Objetivos - metricas de performance, deteção de erros e tracking de configuração
- Resultados: geração de alertas e apresentação gráfica
- Ex: kibana

Caso de estudo : ELK



Monitoring and Management



Benchmarking

Metrics

- Response time - intervalo entre o pedido do utilizador e a resposta do sistema
- Throughput - taxa à qual os pedidos são servidos pelo sistema