

Cloud and big data technology

Cloud Architecture

INF-2220

Dilip K. Prasad



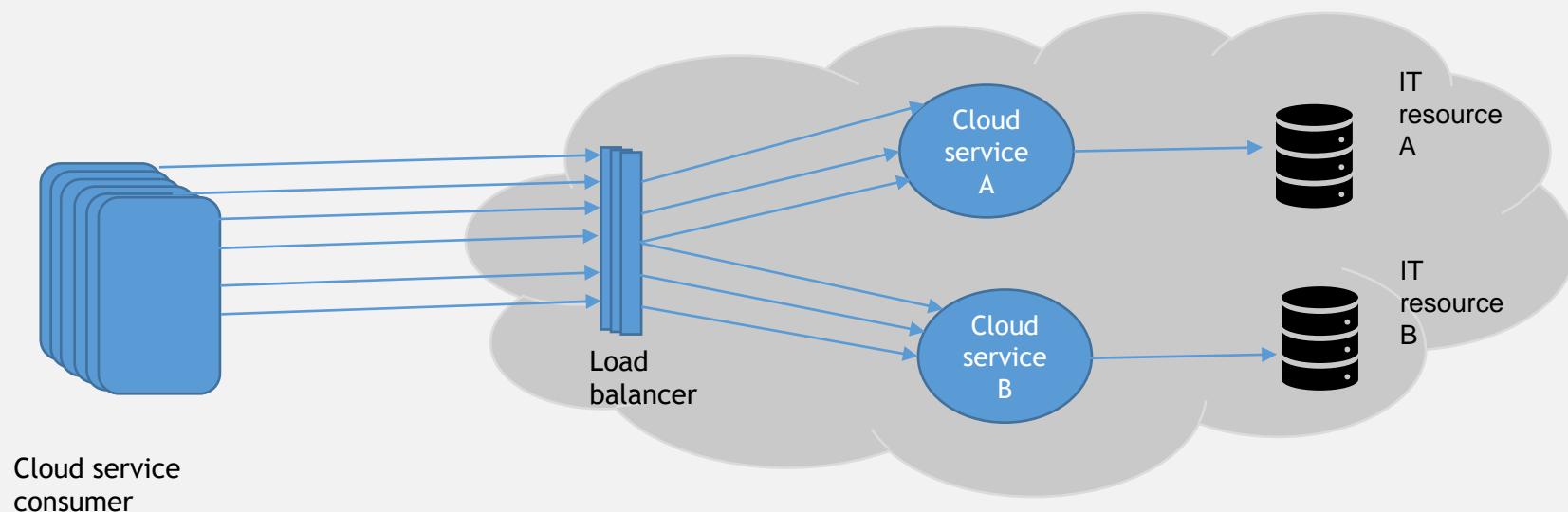
Outline

- Overview
- Workload distribution architecture
- Resource pooling architecture
- Dynamic scalability architecture
- Redundant Storage architecture
- Cloud bursting architecture
- Hypervisor Clustering architecture
- Bare-metal provisioning architecture

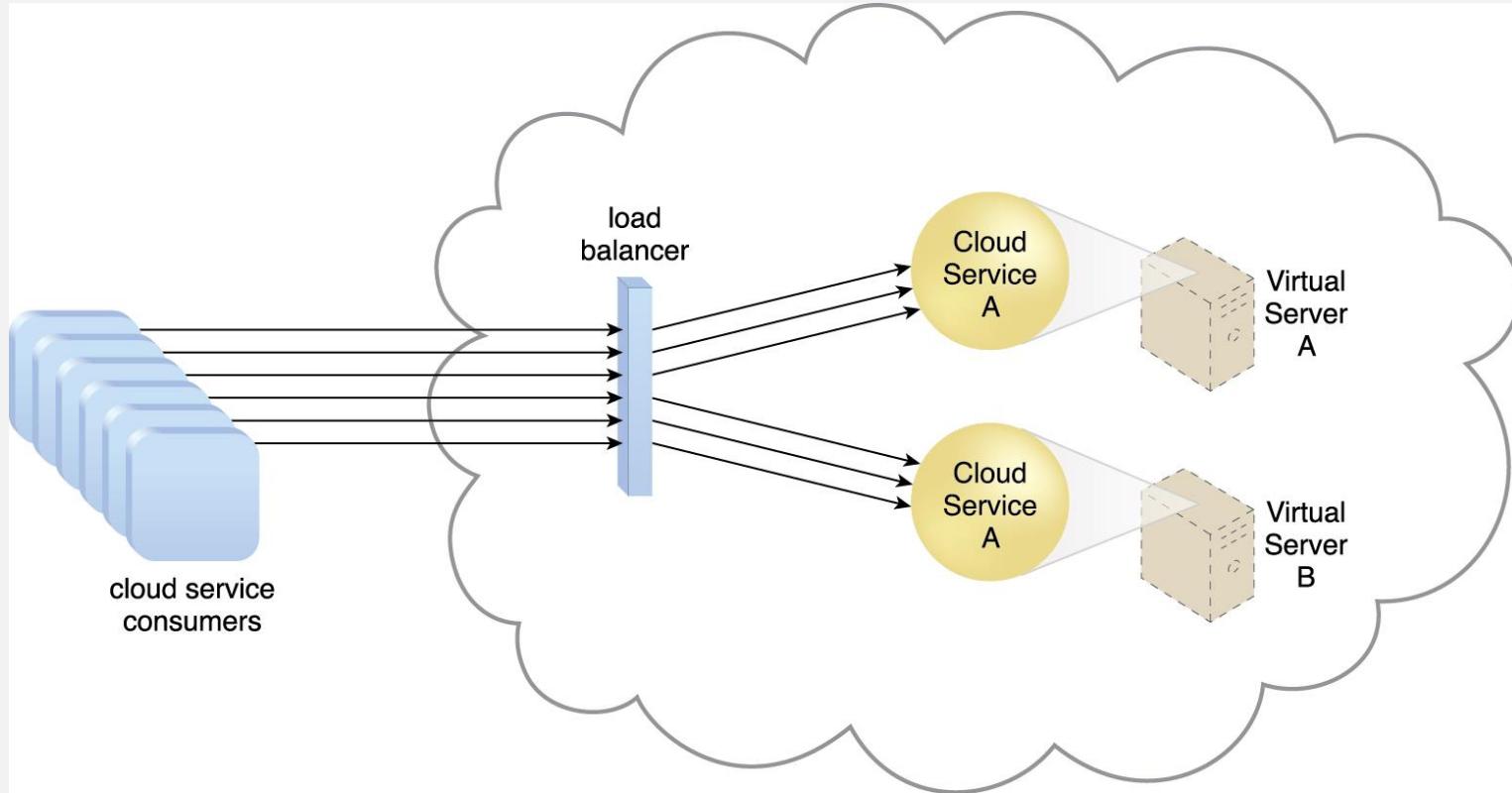


Workload distribution architecture

- Whenever there is a shortage of IT resources, easiest way to overcome this additional resource need is by adding extra resources in horizontal scaling fashion.
- A load balancer will try to distribute the load evenly among the available IT resources.



Workload distribution architecture



Audit Monitor, cloud usage monitor, hypervisor, etc can also be a part of this architecture

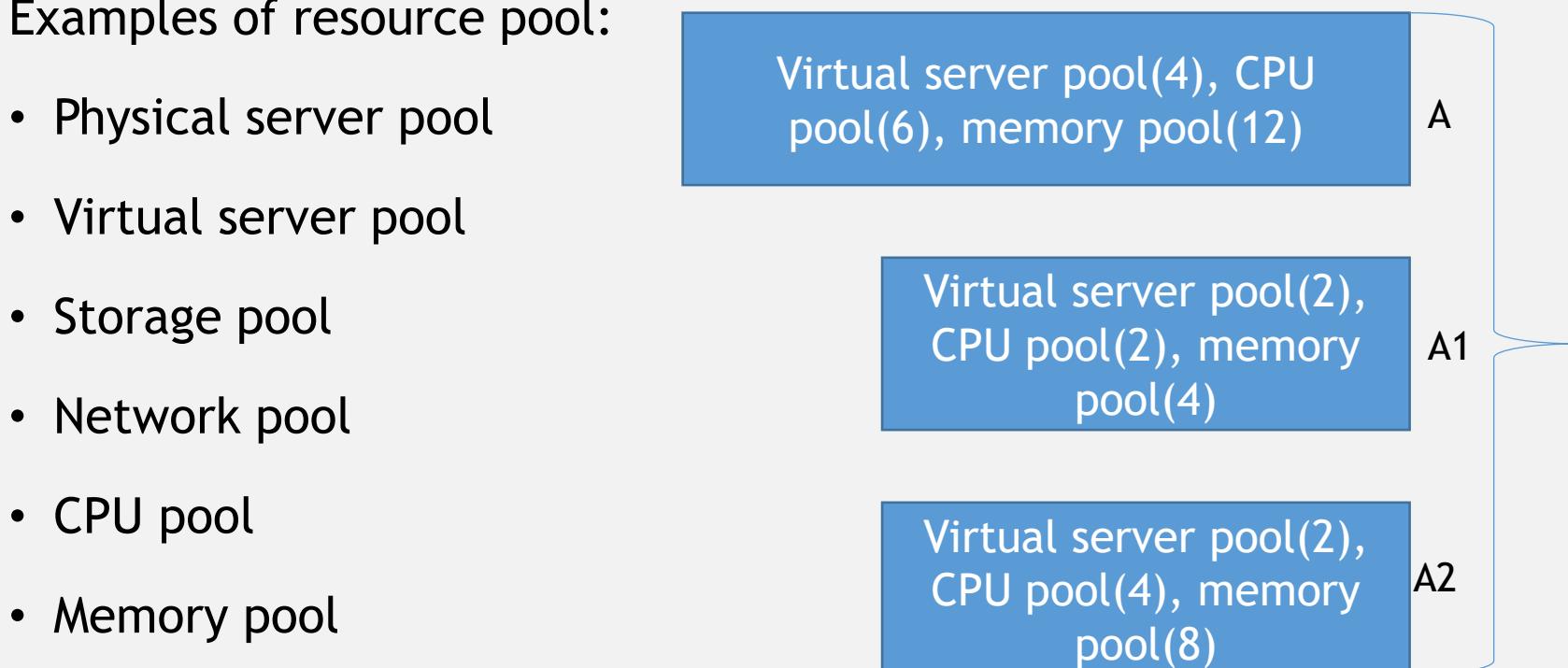
Image Source: www.informit.com/title/9780133387520

Resource pooling architecture

- It's based on the use of one or more resource pools, in which identical IT resources are grouped and maintained by a system that automatically ensures that they remain synchronized.

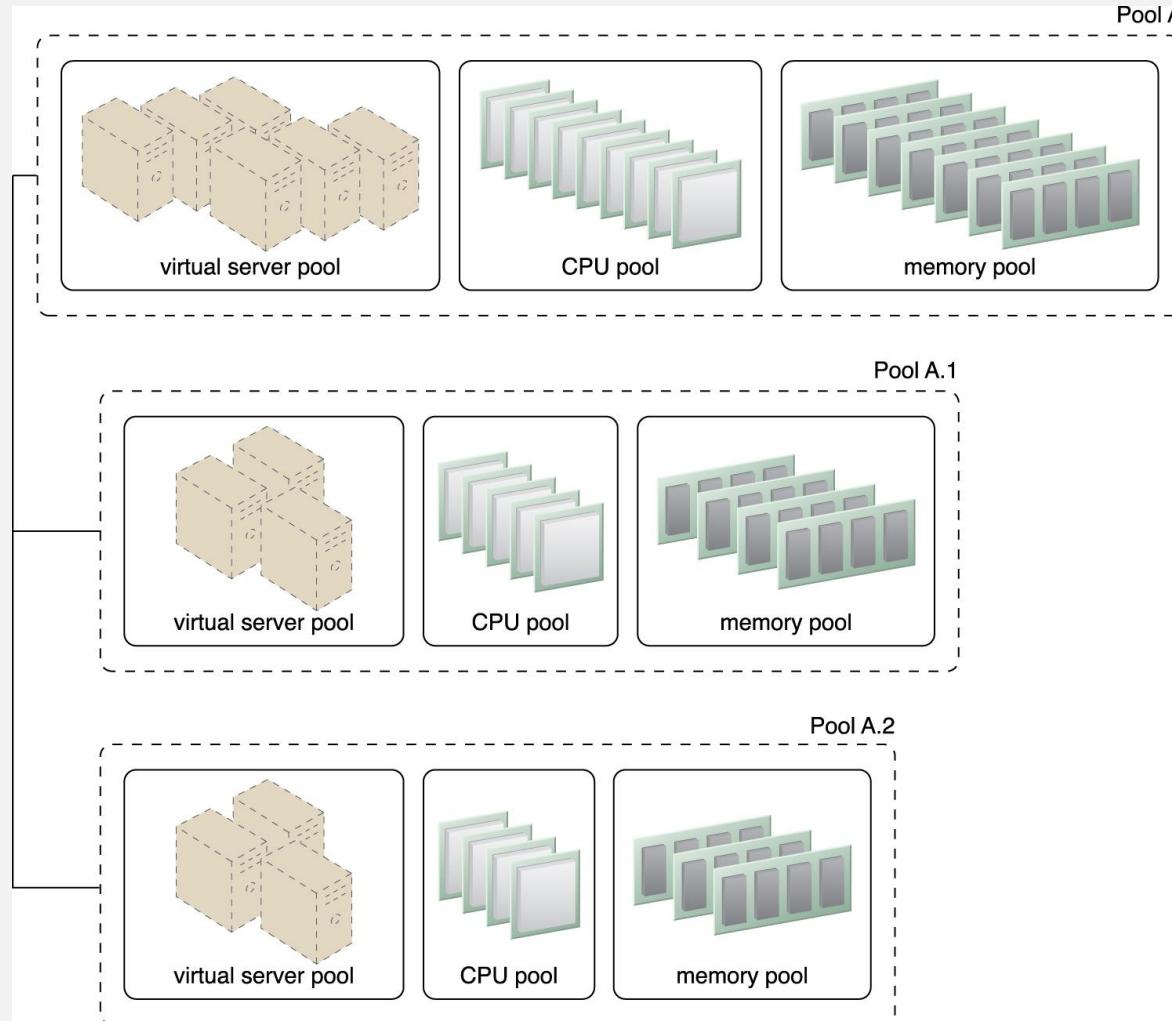
Examples of resource pool:

- Physical server pool
- Virtual server pool
- Storage pool
- Network pool
- CPU pool
- Memory pool



Nested pool A1 and A2 have same resource pool as pool A, but in diff. quantities

Resource pooling architecture



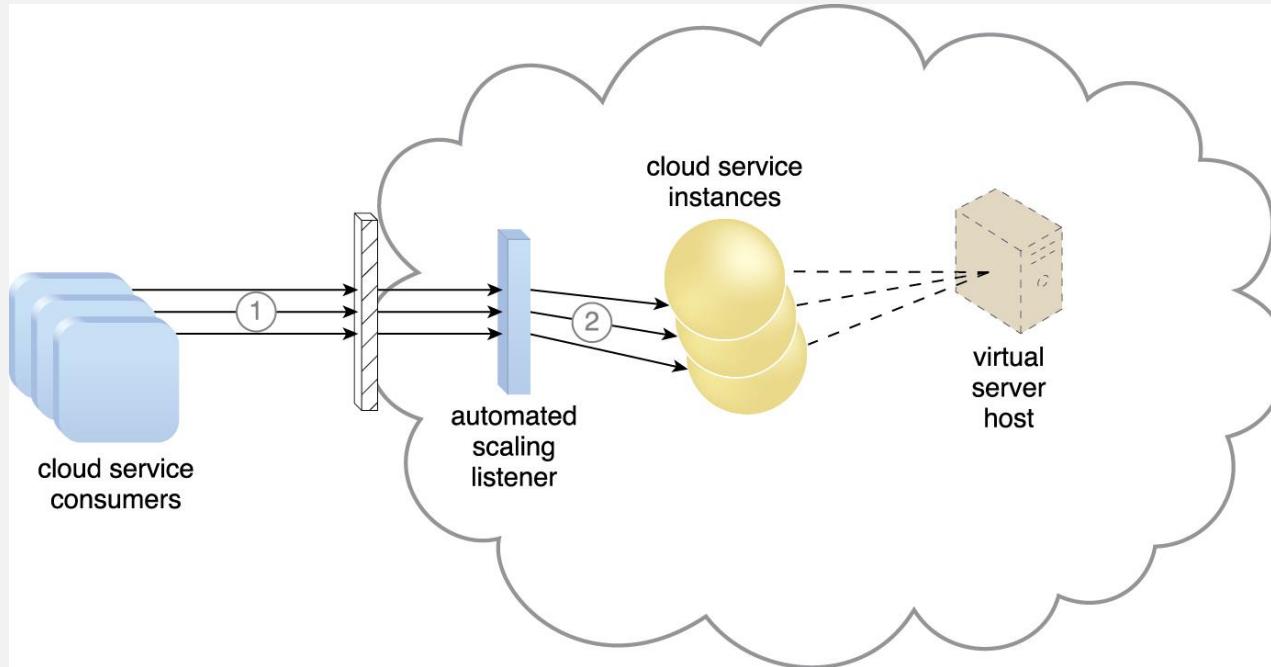
Audit Monitor, cloud usage monitor, hypervisor, etc can also be a part of this architecture

Image Source: www.informit.com/title/9780133387520

Dynamic scalability architecture

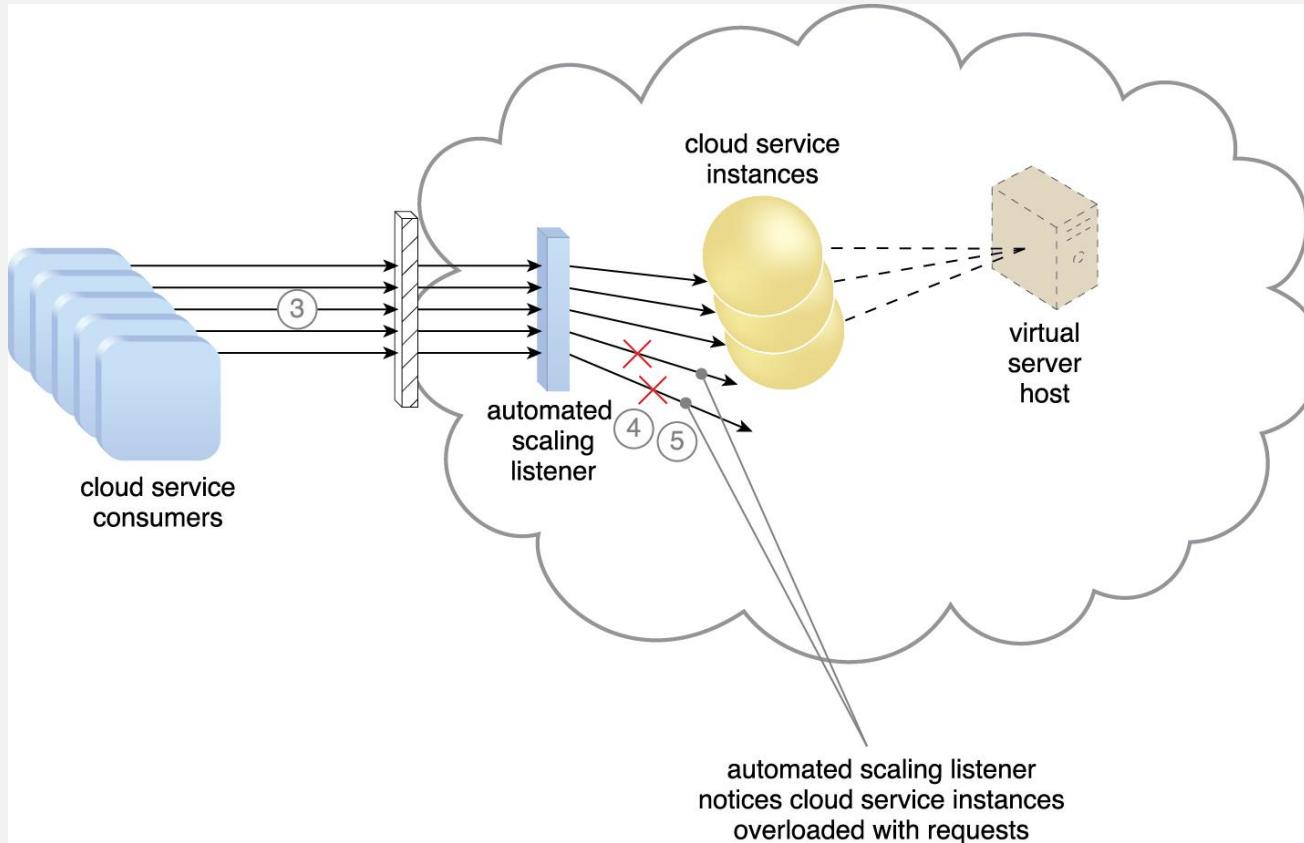
- It is based on pre-defined trigger conditions that trigger the dynamic allocation of IT resources from resource pools.
- Dynamic horizontal scaling - IT resource instances are scaled **out-and-in** to handle fluctuating workloads.
- Dynamic vertical scaling - IT resource instance are scaled up-and-down when there is a need to adjust the processing capacity of single IT resource.
- Dynamic relocation - IT resource is relocated to a host with more capacity.

Dynamic scalability architecture



1. Cloud service consumers sending requests to a cloud service
2. Automated scaling listener monitors the cloud service to determine if predefined capacity are being exceeded

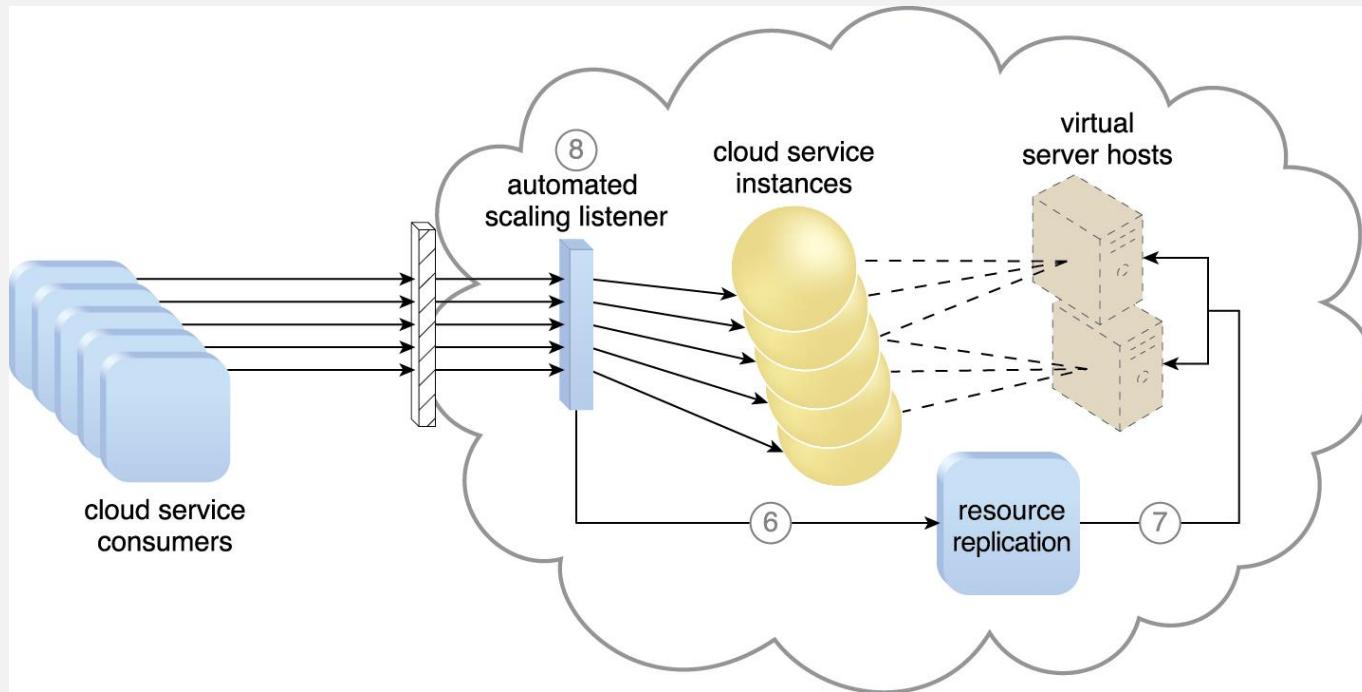
Dynamic scalability architecture



Cloud usage monitor, hypervisor, Pay-per-use monitor can also be a part of this architecture

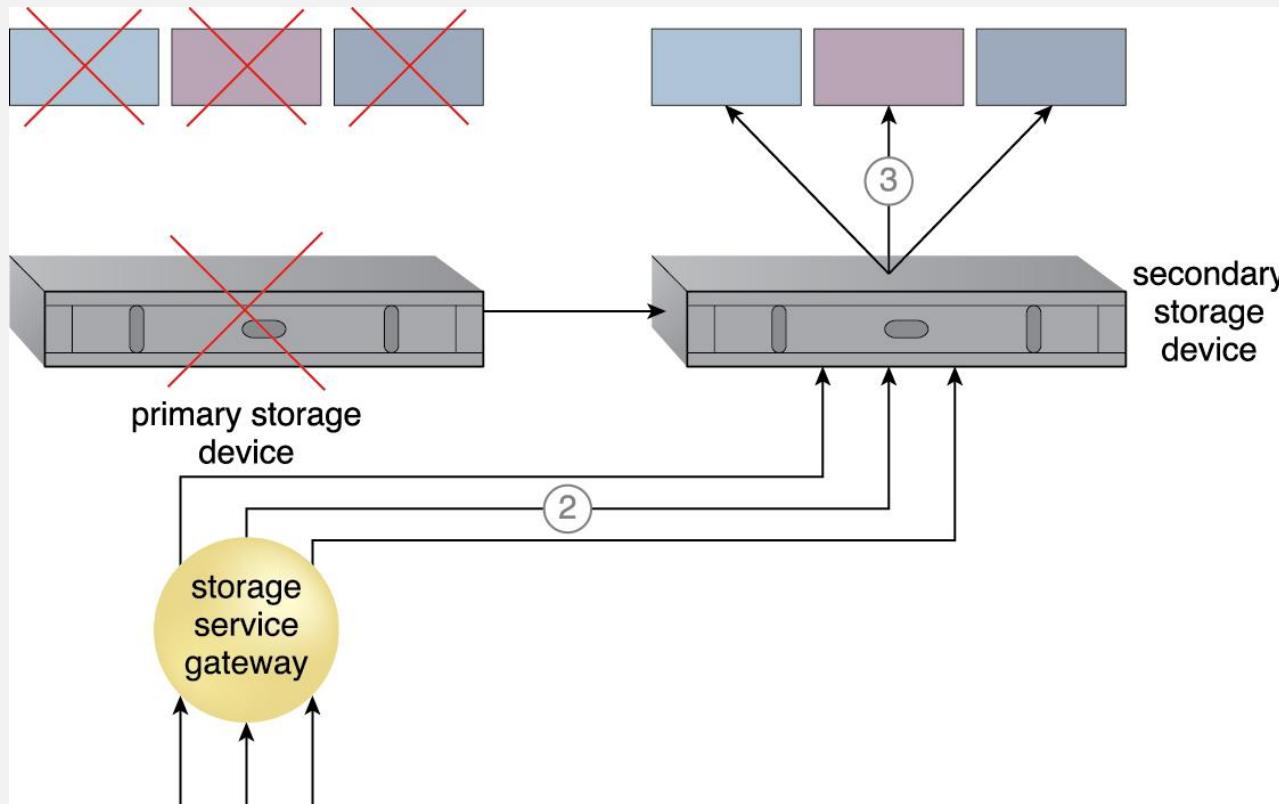
Image Source: www.informit.com/title/9780133387520

Dynamic scalability architecture



Redundant Storage architecture

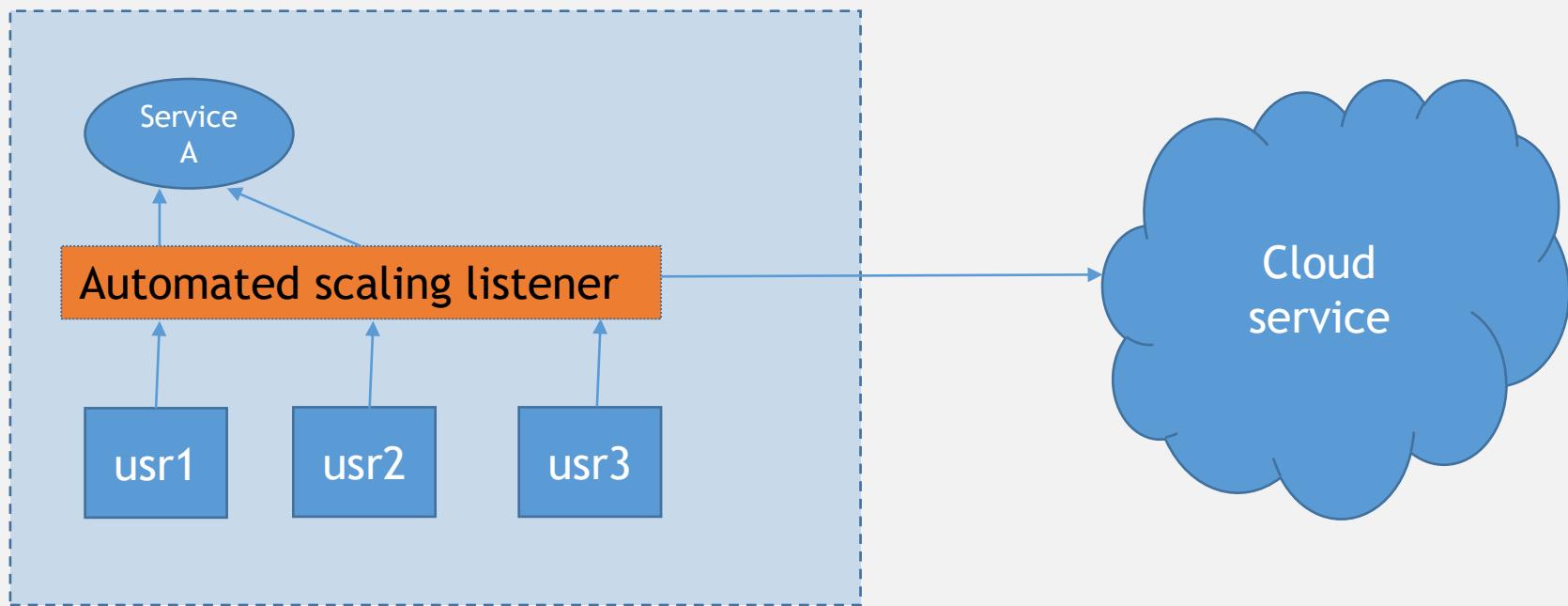
Cloud storage are occasionally subject to failure and disruptions that are caused by network connectivity failure, or security breaches, or hardware failure.



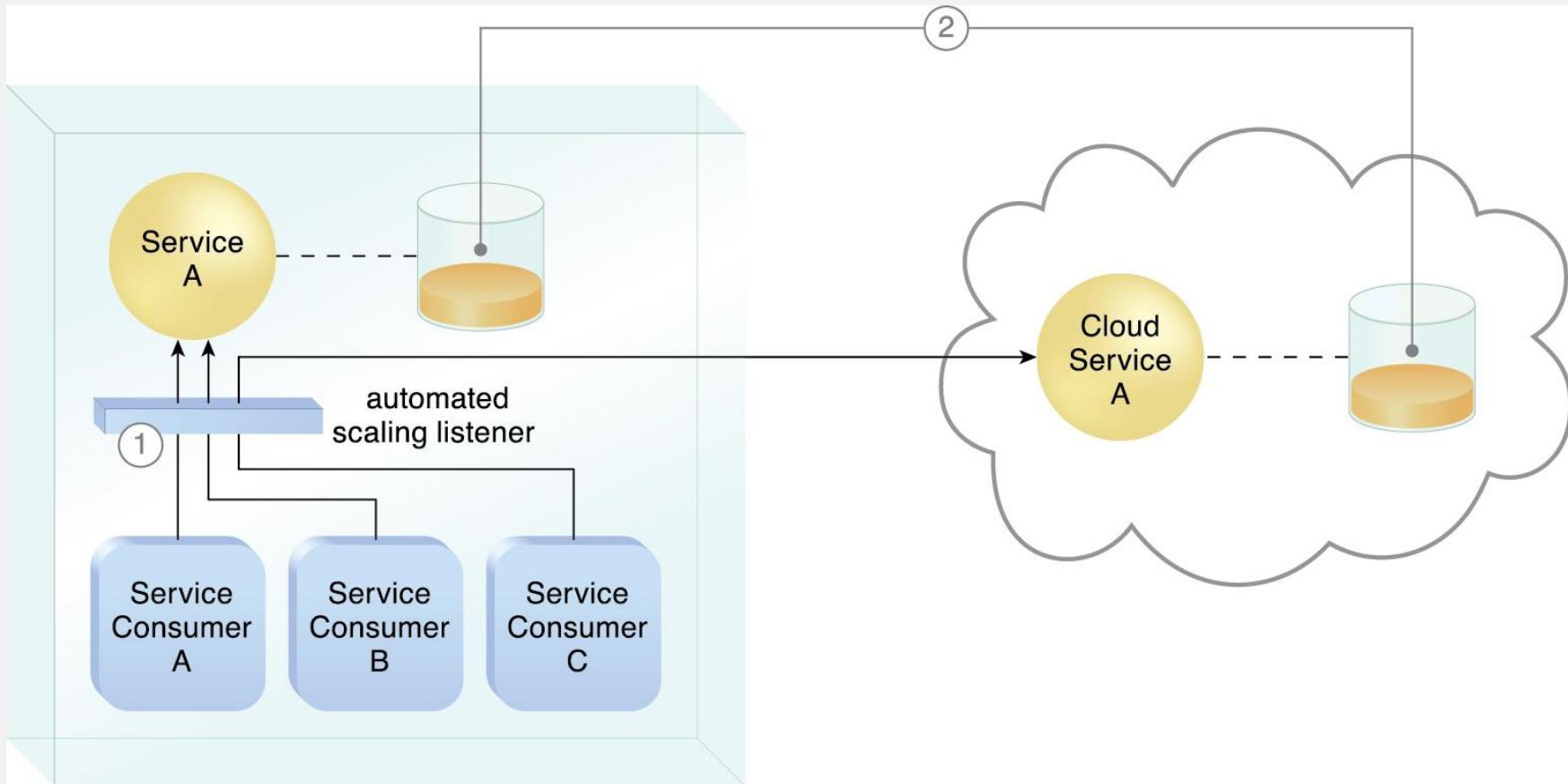
Storage service gateway is capable of automatically redirecting cloud consumer requests whenever location of requested data has changed

Cloud bursting architecture

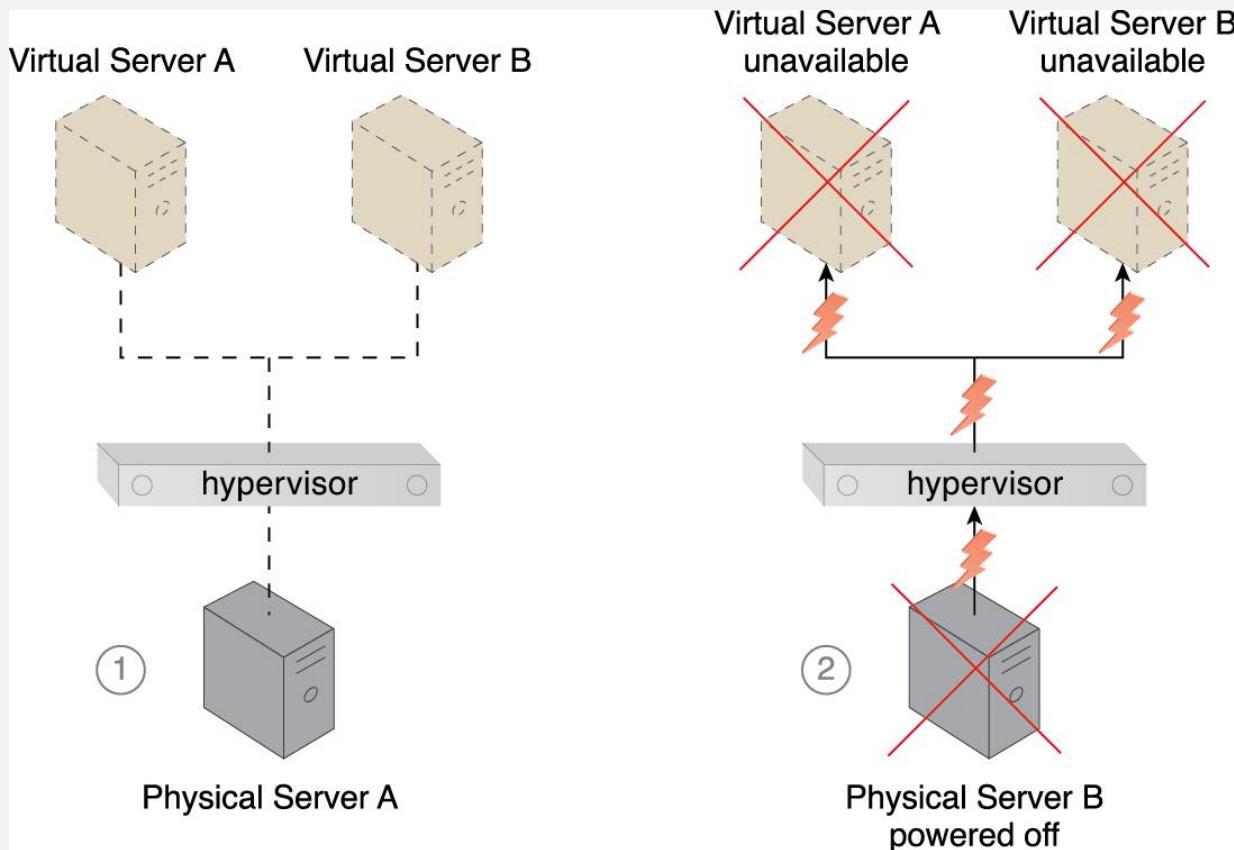
- It is based on pre-defined trigger conditions that trigger the dynamic scaling that scales or “burst out” on-premise IT resource into a cloud whenever the trigger condition like capacity threshold reached.



Cloud bursting architecture

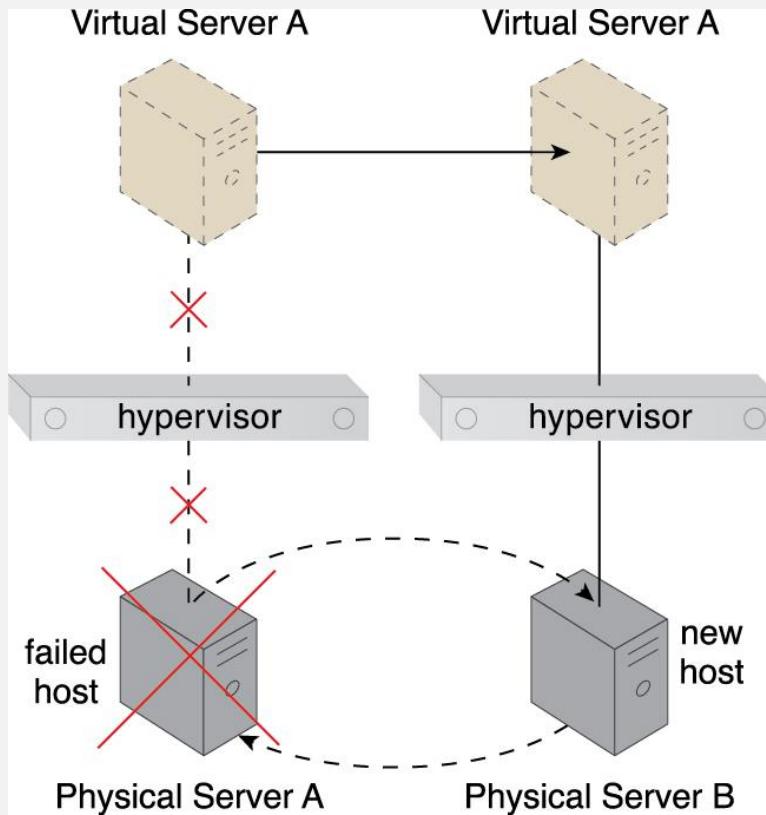


Hypervisor Clustering architecture



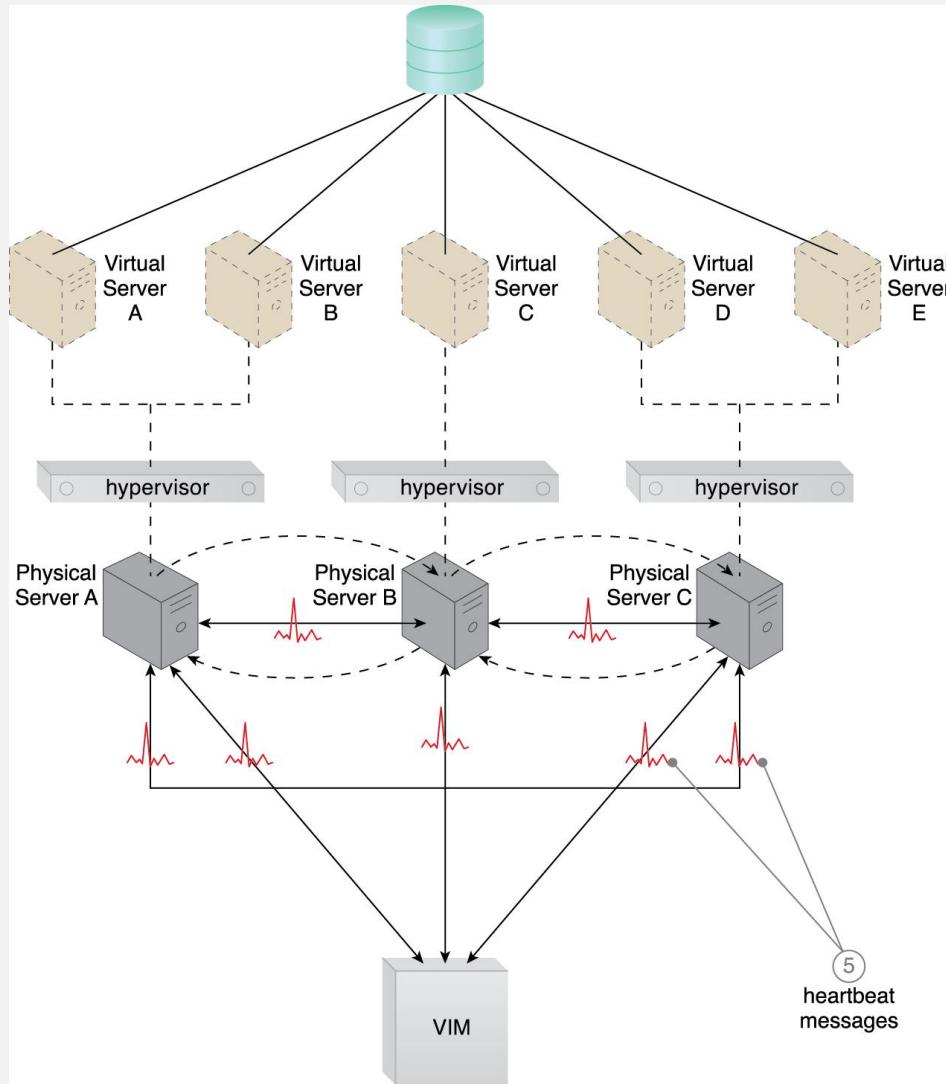
Hypervisor is a key component for VM creation so what if Hypervisor fails.

Hypervisor Clustering architecture



Hypervisor is a key component for VM creation so what if Hypervisor fails. This architecture can be used to handle such situations

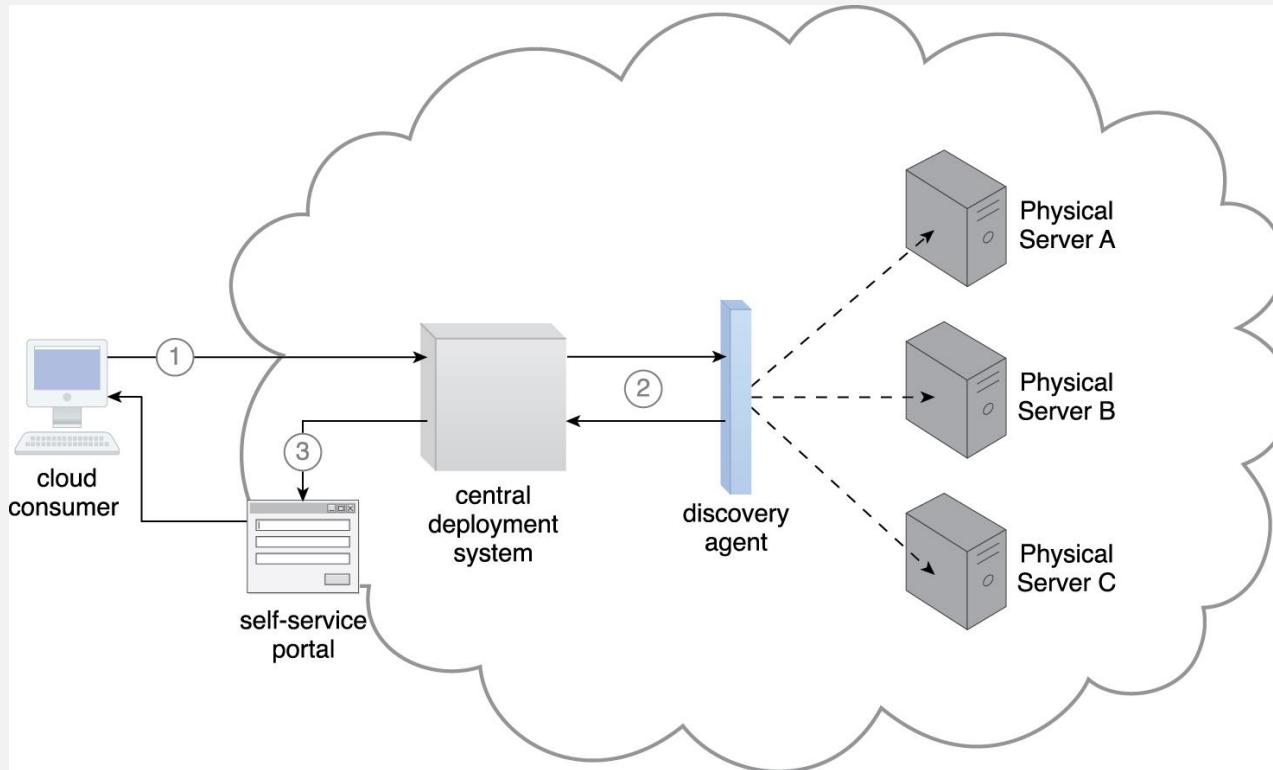
Hypervisor Clustering architecture



Heartbeats are system level messages exchanged between hypervisors and virtual servers, and hypervisors and VIMs



Bare-metal Provisioning architecture



Bare-metal hypervisor(Type1)- where it is directly installed on the server hardware

Hypervisor(Type 2 or Hosted) is installed on top of the parent OS

Image Source: www.informit.com/title/9780133387520

Thanks!



Cloud and big data technology

Cloud Computing Security

INF-2220

Dilip K. Prasad



Outline

- Overview
- Some concepts
- Cloud security threats
- Cloud security mechanisms

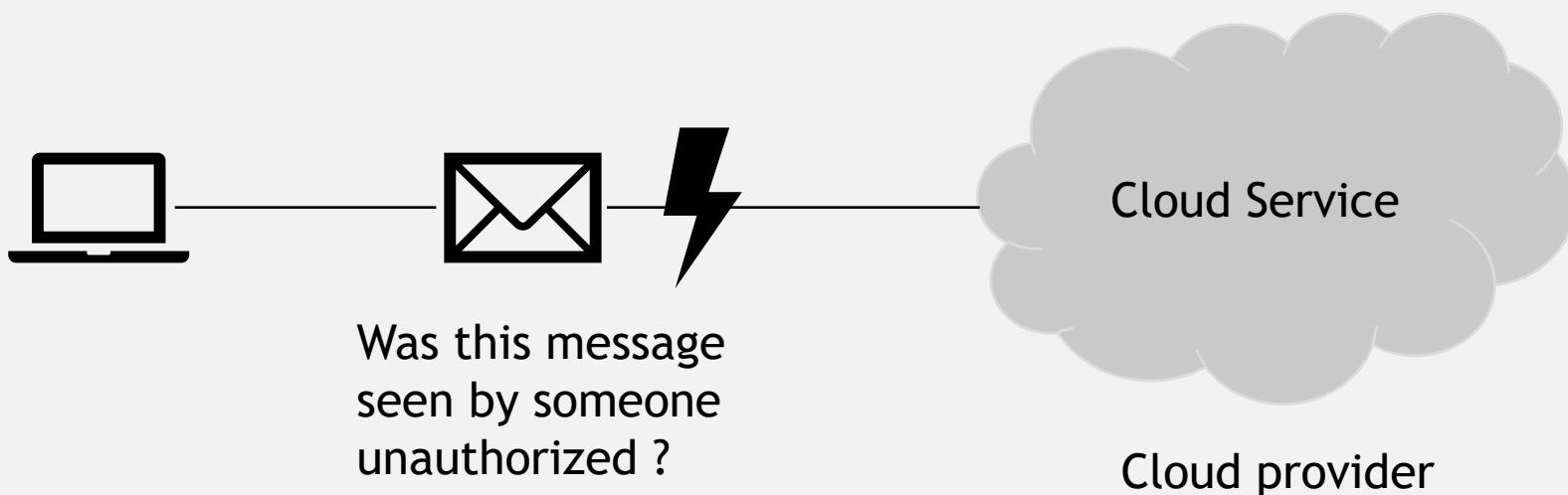


Overview

- Confidentiality
- Integrity
- Authenticity
- Availability
- Threat
- Vulnerability
- Risk

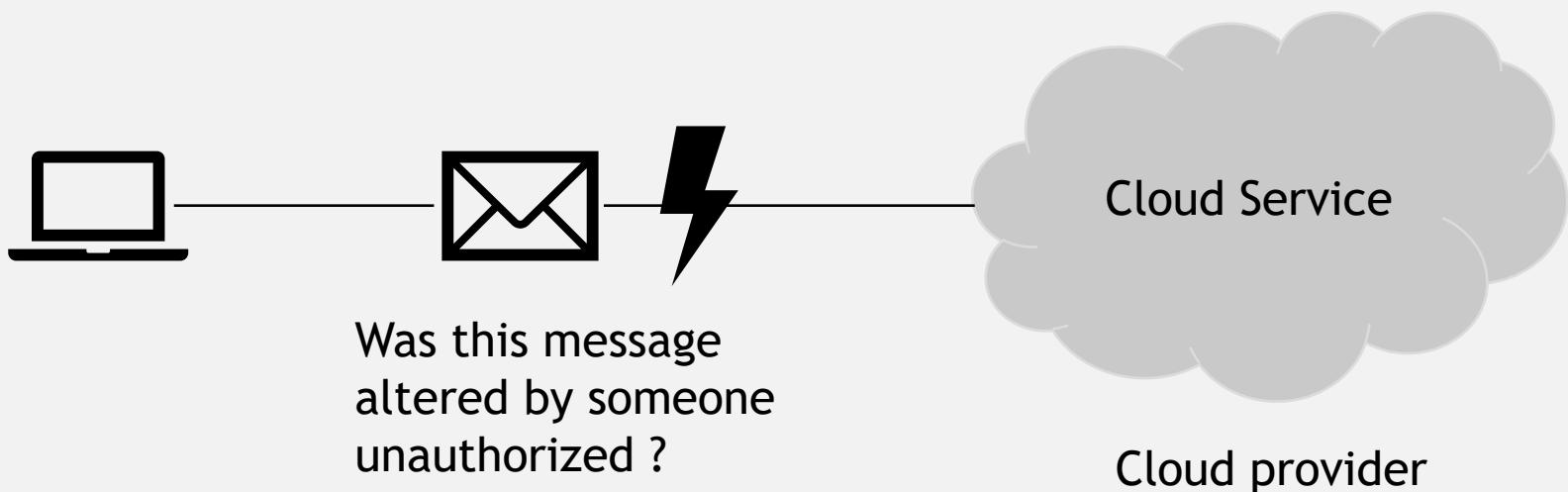
Confidentiality

- Confidentiality - is the characteristic of something being made accessible only to the authorized parties.
- With respect to cloud computing for confidentiality data access should be restricted during data transfer and storage.



Integrity

- Integrity - is the characteristic of not having been altered by an unauthorized party.
- With respect to cloud computing for data integrity, whether the data transferred is data received by the consumer can be guaranteed by the cloud provider.
- Data lifecycle - create, store, process, archive, destroyed.



More security concepts

- Authenticity - something having been provided by an authorized source.
- Availability - being accessible and usable during the specified time period.
- Threat - potential security violation that can challenge privacy or harmful to the organization.
- Vulnerability - weaknesses of the cloud computing system that can be exploited to cause attack to the system.
- Risk - probabilities of loss or harm from performing an activity. Risk is measured in terms of threat levels or number of vulnerabilities.
- Security controls, mechanism, policies are associated with establishing counter measures and safeguards in support of improving security.

Threat agents

- Anonymous attacker
- Malicious service agent
- Trusted attacker
- Malicious insider



Anonymous attacker

- A non trusted cloud service consumer without permissions in the cloud. It typically exists as an external software program that launches network-level attacks through public networks.

Malicious service agent

- An agent who is able to intercept and forward the network traffic that flows within a cloud. Usually a service agent(or a program pretending to be a service agent) with compromised or malicious logic.

Trusted attacker

- A trusted attacker exists as an authorized cloud service consumer with legitimate credentials that it uses to exploit access to cloud-based IT resources.

Malicious insider

- A malicious insider is a human that attempts to abuse access privileges to cloud premises.

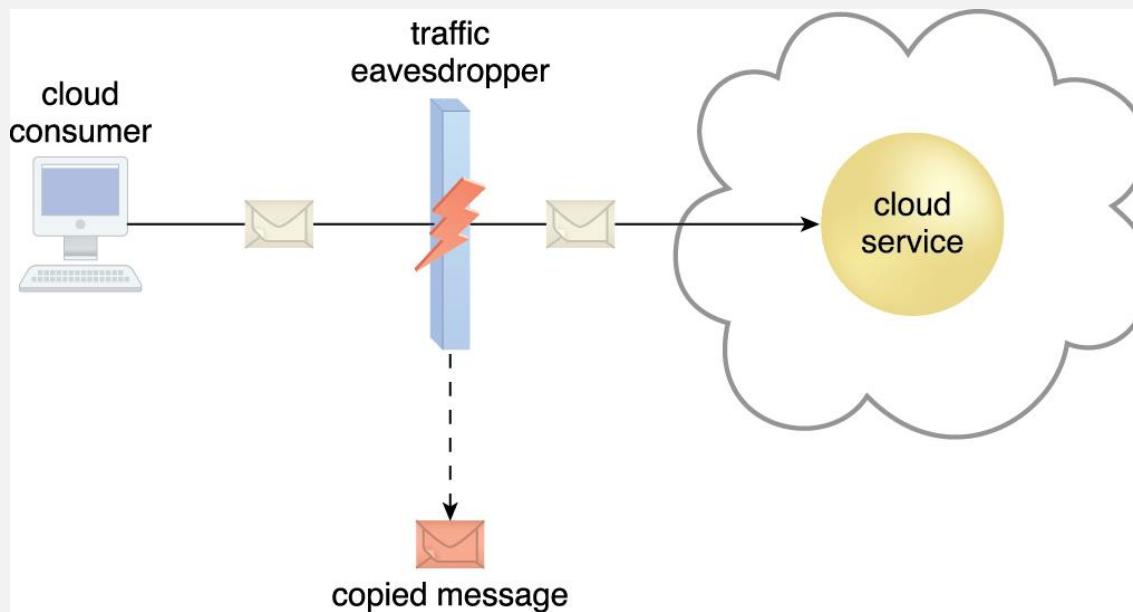
Cloud security threats

- Traffic eavesdropping
 - Malicious intermediary
 - Denial of Service(DoS)
 - Insufficient authorization
 - Virtualization attack



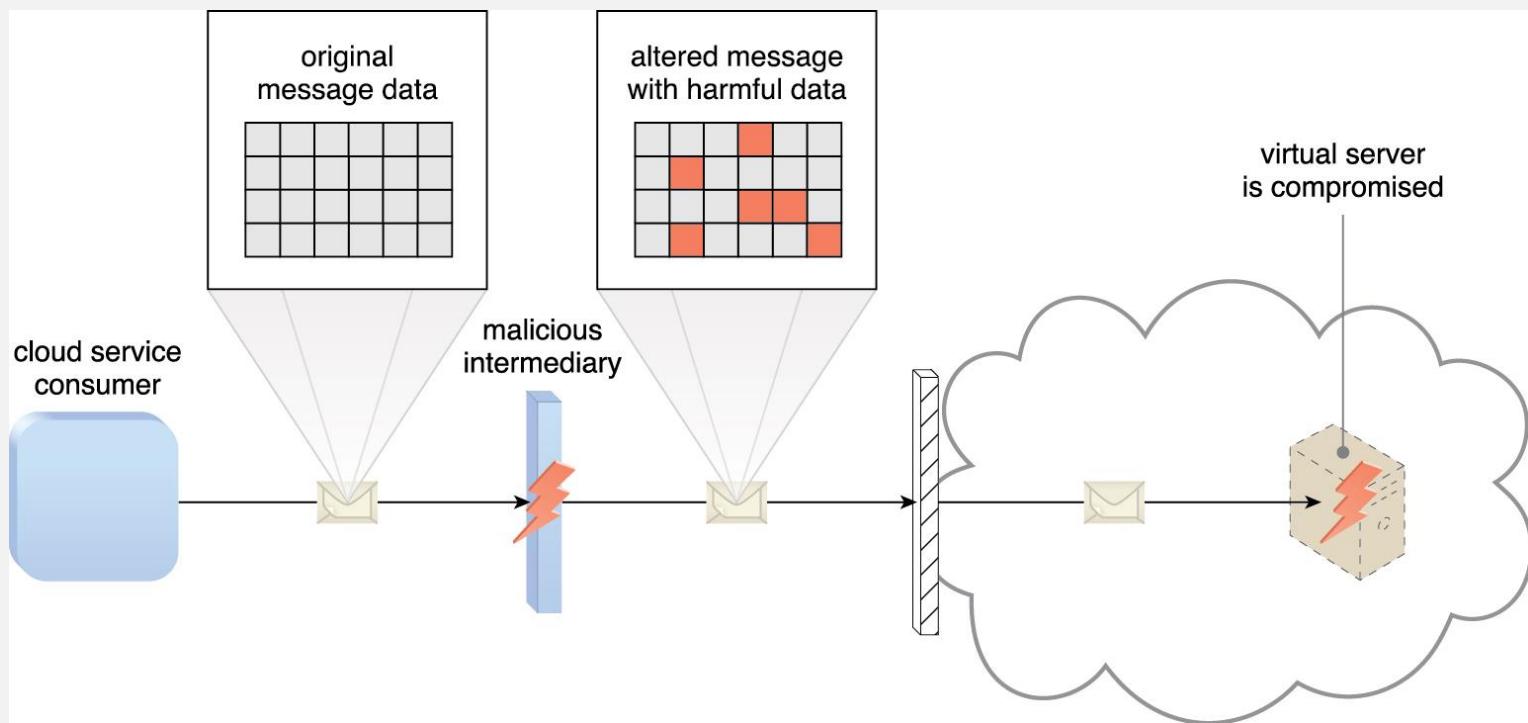
Traffic Eavesdropping

- It occurs when the data being transferred to or within a cloud(from the cloud consumer to the cloud provider) is passively intercepted by a malicious service agent for illegitimate information gathering purposes.



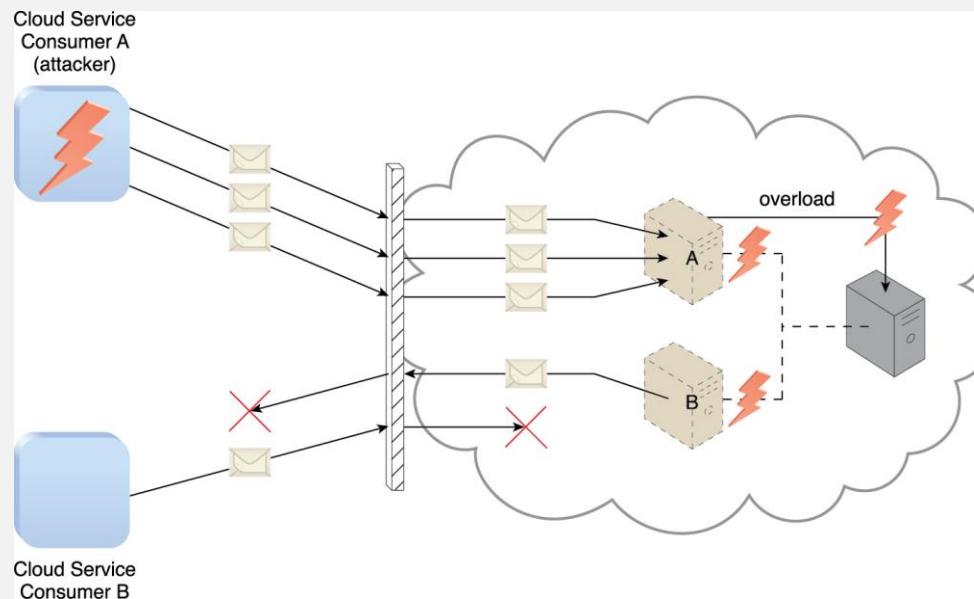
Malicious Intermediary

- This threat arises when messages are intercepted and altered by a malicious service agent.
- Potentially compromise of message confidentiality and/or integrity.



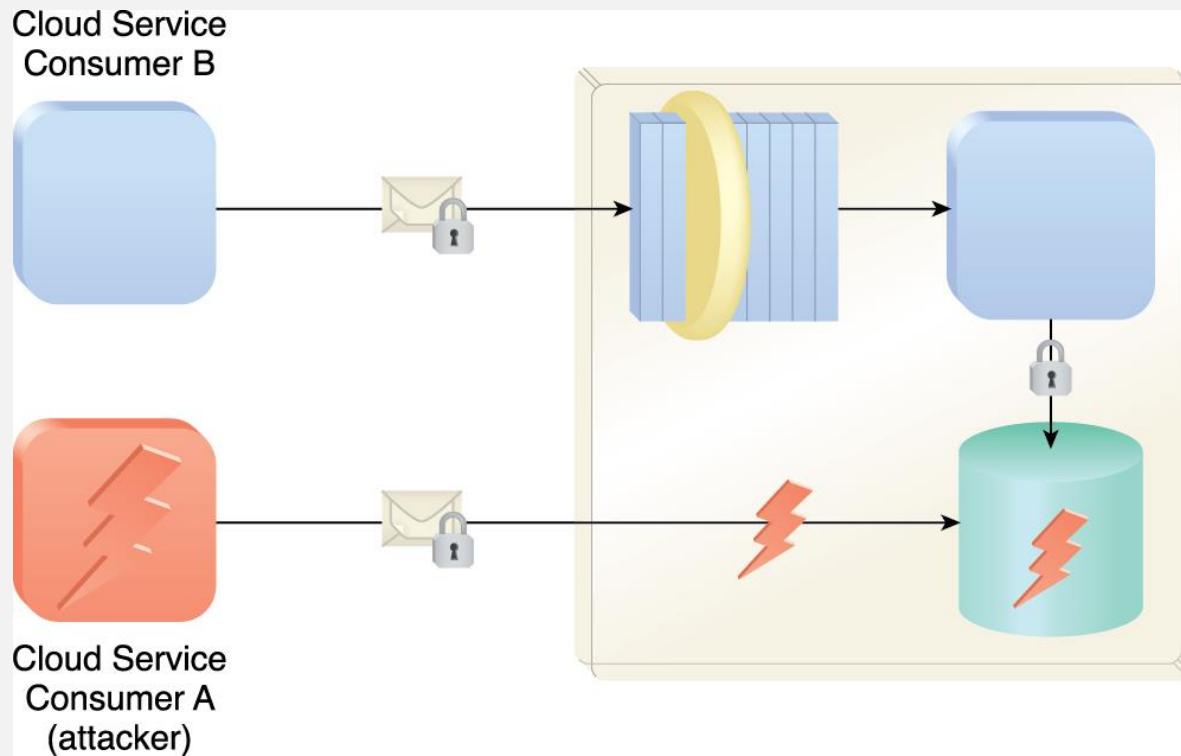
Denial of Service

- The objective of DoS attack is to overload IT resources to the point where they can not function properly.
 - Workload on cloud services are artificially increased
 - Network is overloaded with traffic to reduce its responsiveness and cripple its performance
 - Multiple cloud service requests are sent to consume excessive memory and resources.

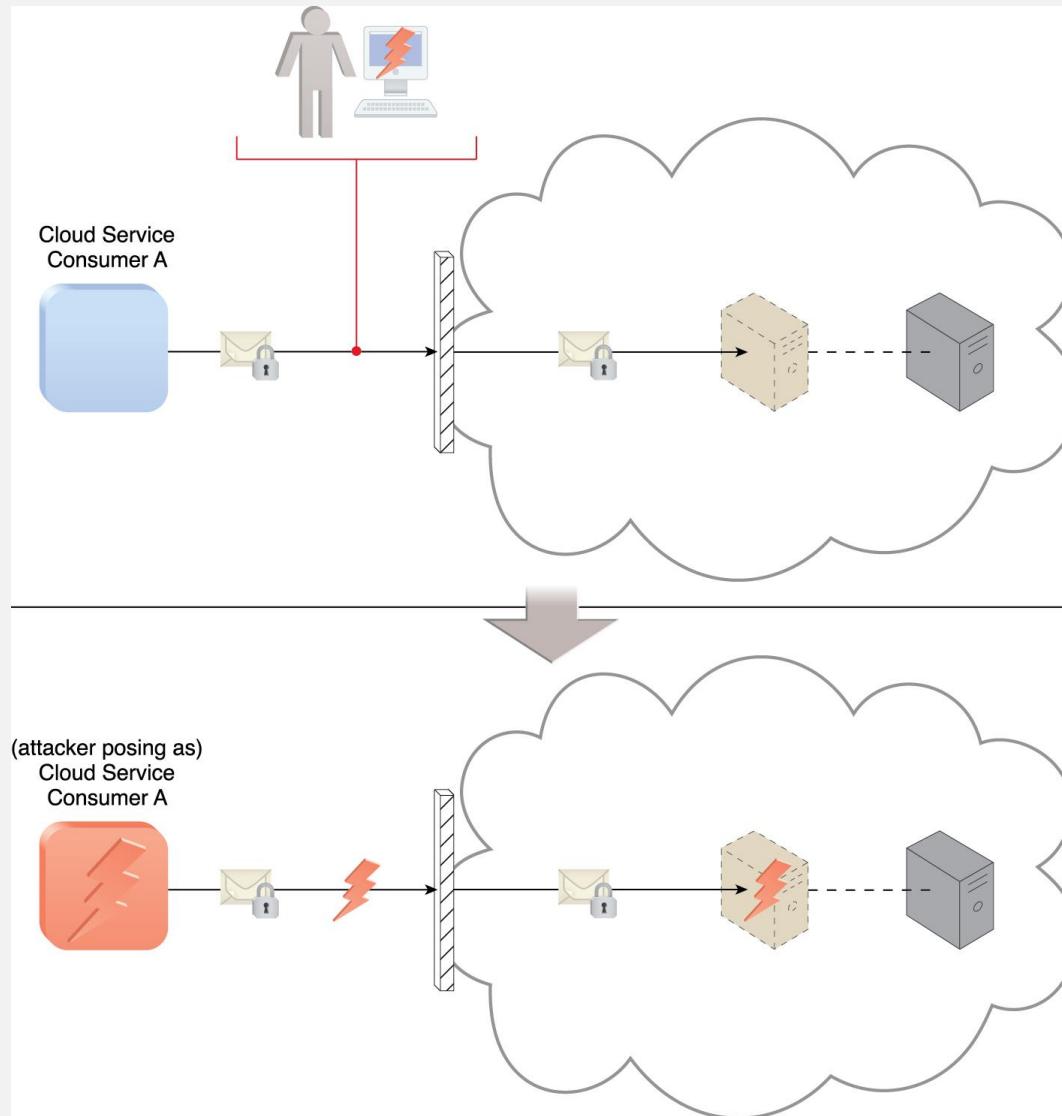


Insufficient authorization

- This attack occurs when access is granted to an attacker erroneously or too broadly, resulting in the attacker getting access to IT resources that are normally protected.

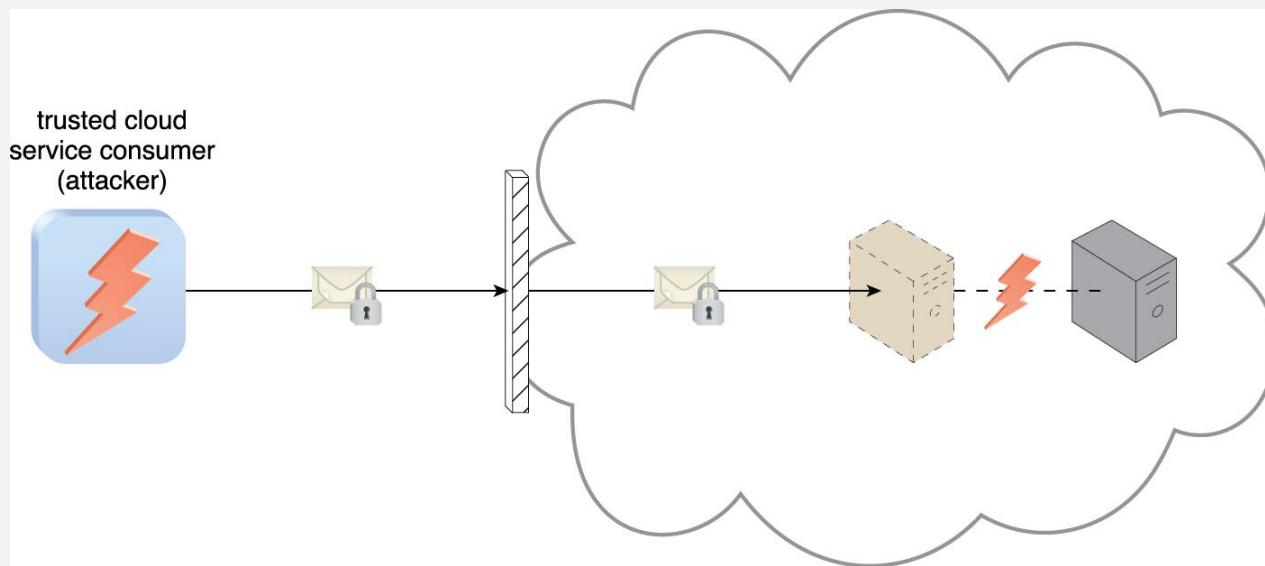


Insufficient authorization



Virtualization Attack

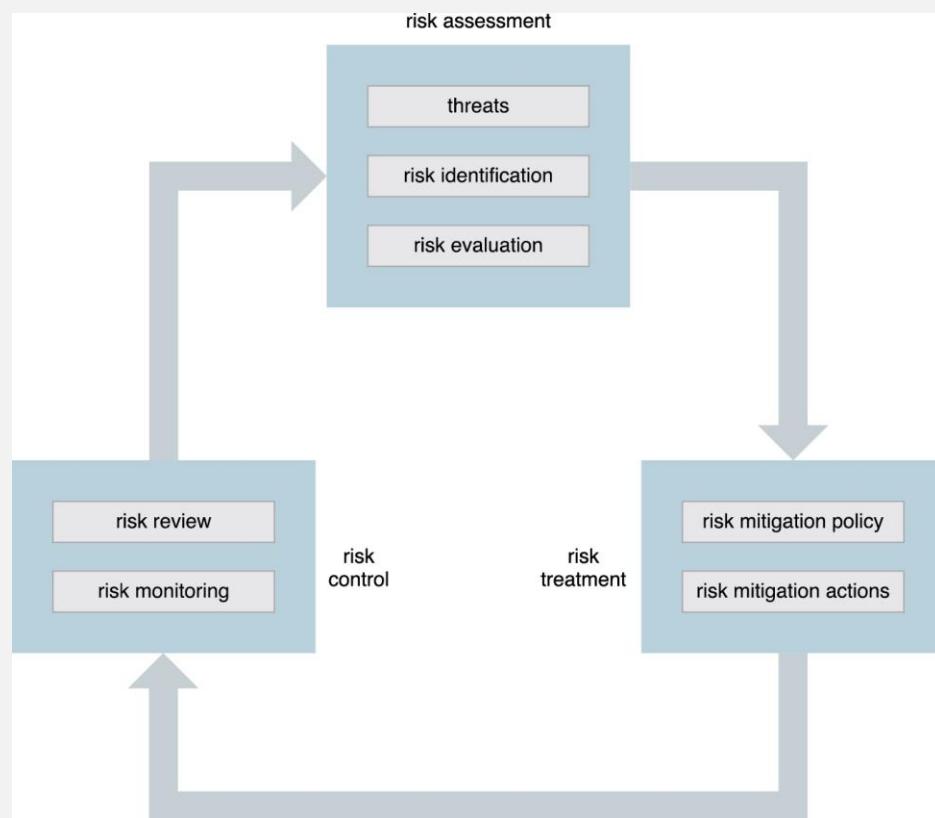
- Virtualization provides multiple cloud consumers with access to IT that share underlying hardware but are logically isolated from each other.
- There is an inherent risk that underlying physical IT resources could be attacked by a tenant in a multi-tenancy cloud system.



Additional Consideration

- Flawed implementations
- Security Policy disparities
- Contracts(SLA)
- Risk Management
 - Assessment
 - Treatment
 - Control

Cloud Control Matrix



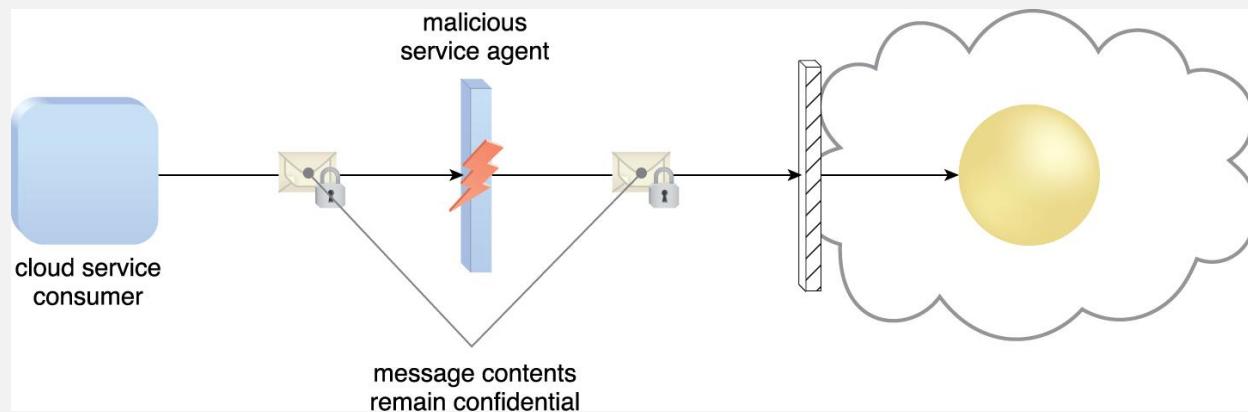
Cloud Security Mechanism

- Encryption
- Hashing
- Digital signature
- Public key Infrastructure(PKI)
- Identity Access management(IAM)
- Single sign-on(SSO)
- Hardened virtual server images



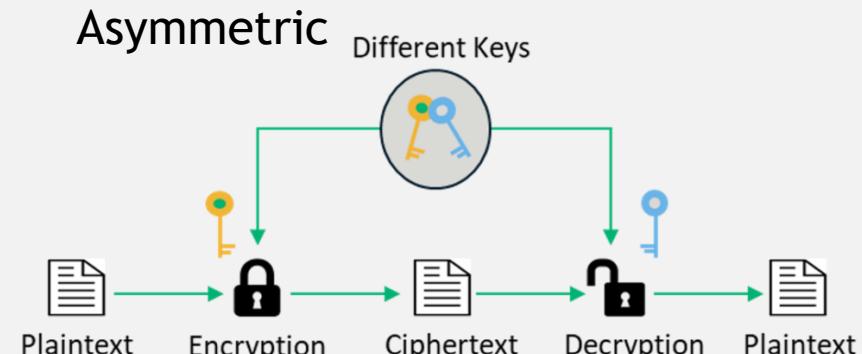
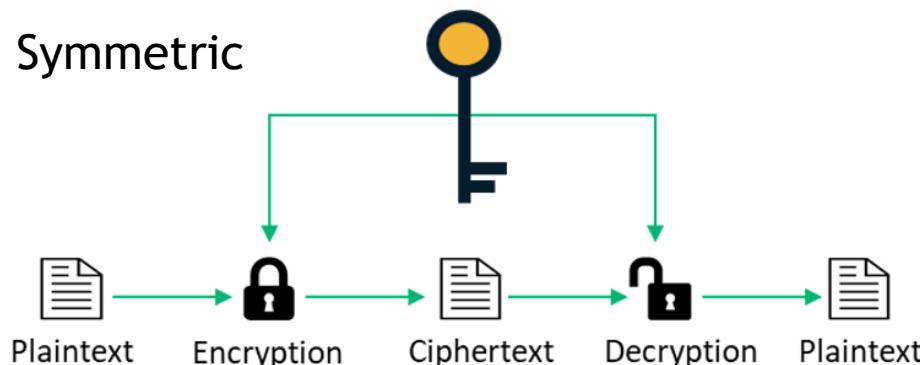
Encryption

- Encryption mechanism is a digital coding system dedicated to preserving the confidentiality and integrity of the data.
- Data is cipherized using encryption key and transferred while authorized user will use the



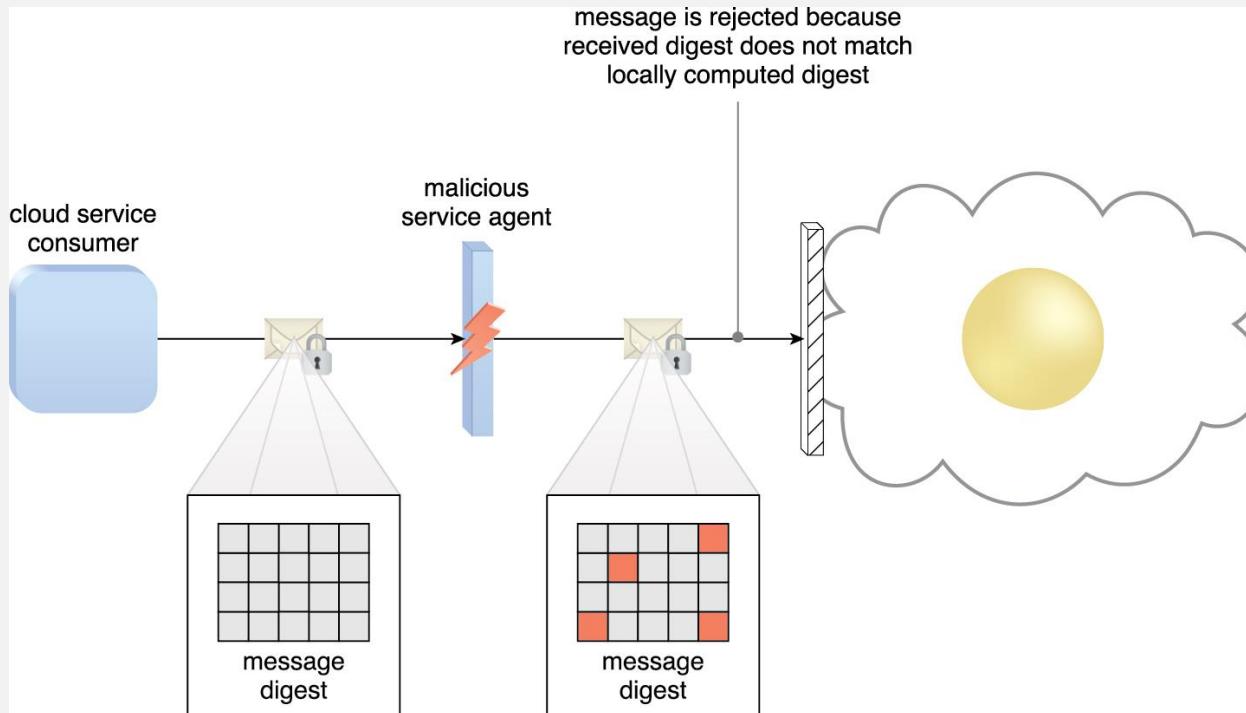
Symmetric/Asymmetric Encryption

- If the same key is used for both encryption and decryption then it's known as Symmetric encryption
- If the different keys(a public key and a private key) are used for encryption and decryption then it is known as Asymmetric encryption.
- Asymmetric encryption is slower than symmetric encryption



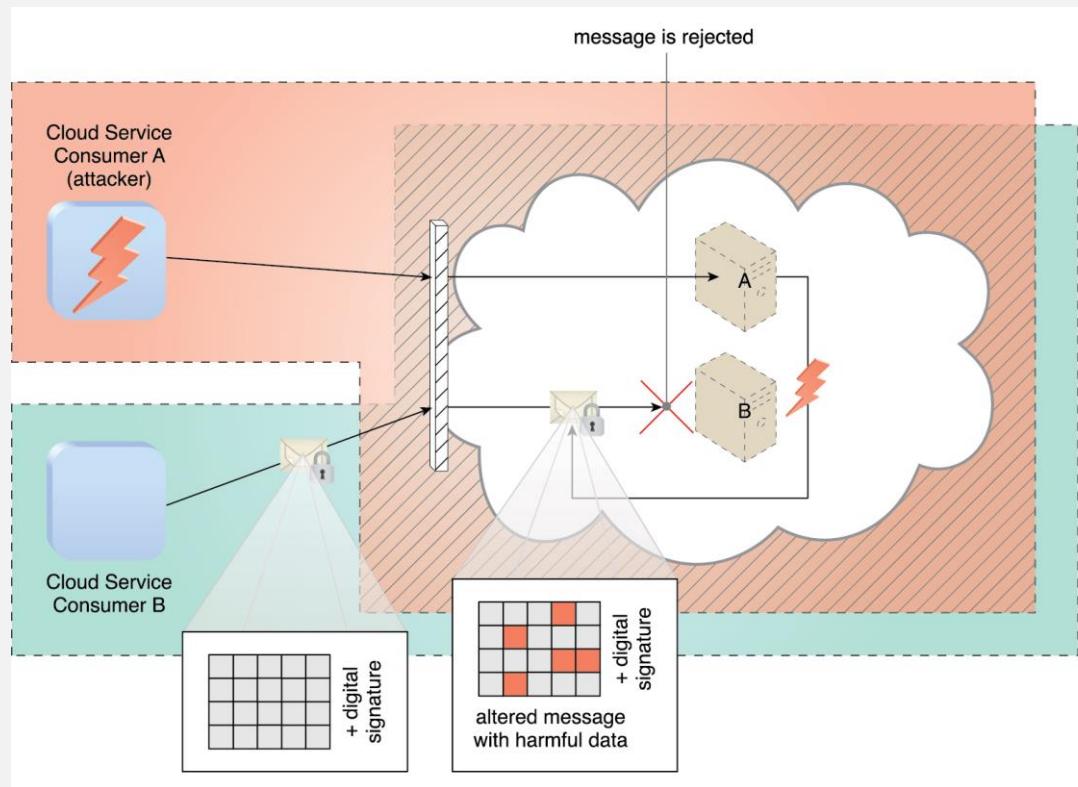
Hashing

- Hashing is used when one-way, irreversible form of data encryption is required. E.g MD5, SHA-2, CRC-32
- Common use is in storage of passwords



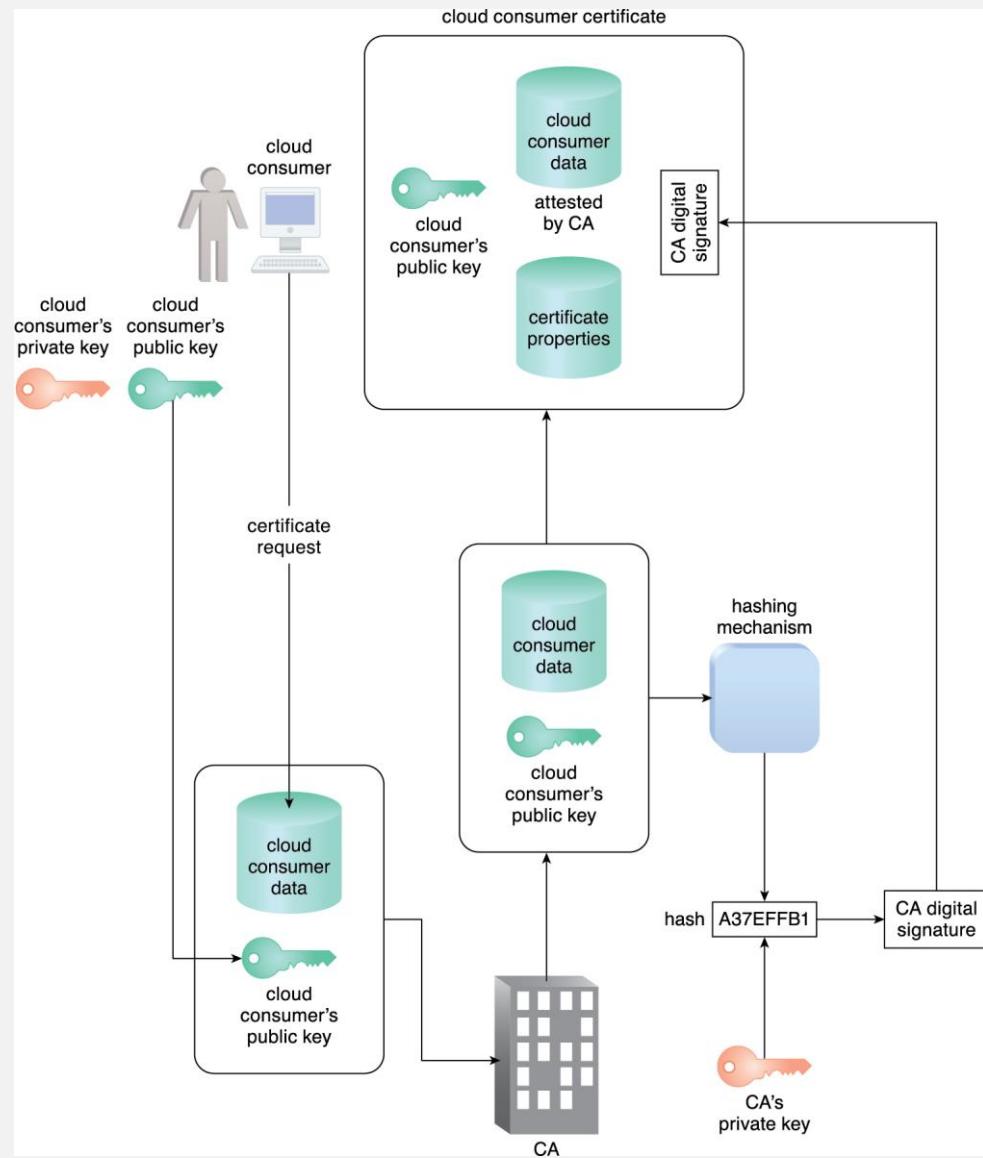
Digital Signature

- It is used to ensure data authenticity and integrity through authentication and non-repudiation.
- Both hashing and asymmetric encryption are used in the creation of digital signature.



Public Key Infrastructure (PKI)

- It's a mechanism to manage issuance of asymmetric keys.
- PKI is a system of protocols, data formats, rules, and practices that enable large scale systems to securely use public key cryptography.

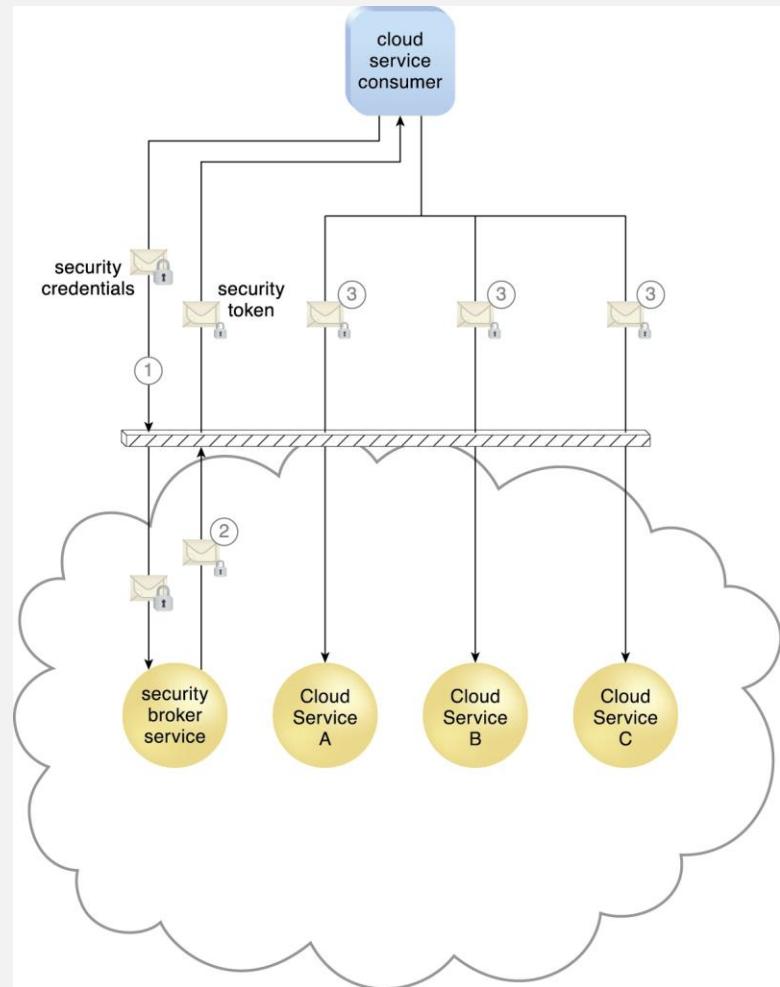


Identity and Access Management (IAM)

- Its process comprises components and policies necessary to control and track user identities and access privileges for IT resources, environment and systems.
- Authentication
- Authorization
- User management
- Credential management

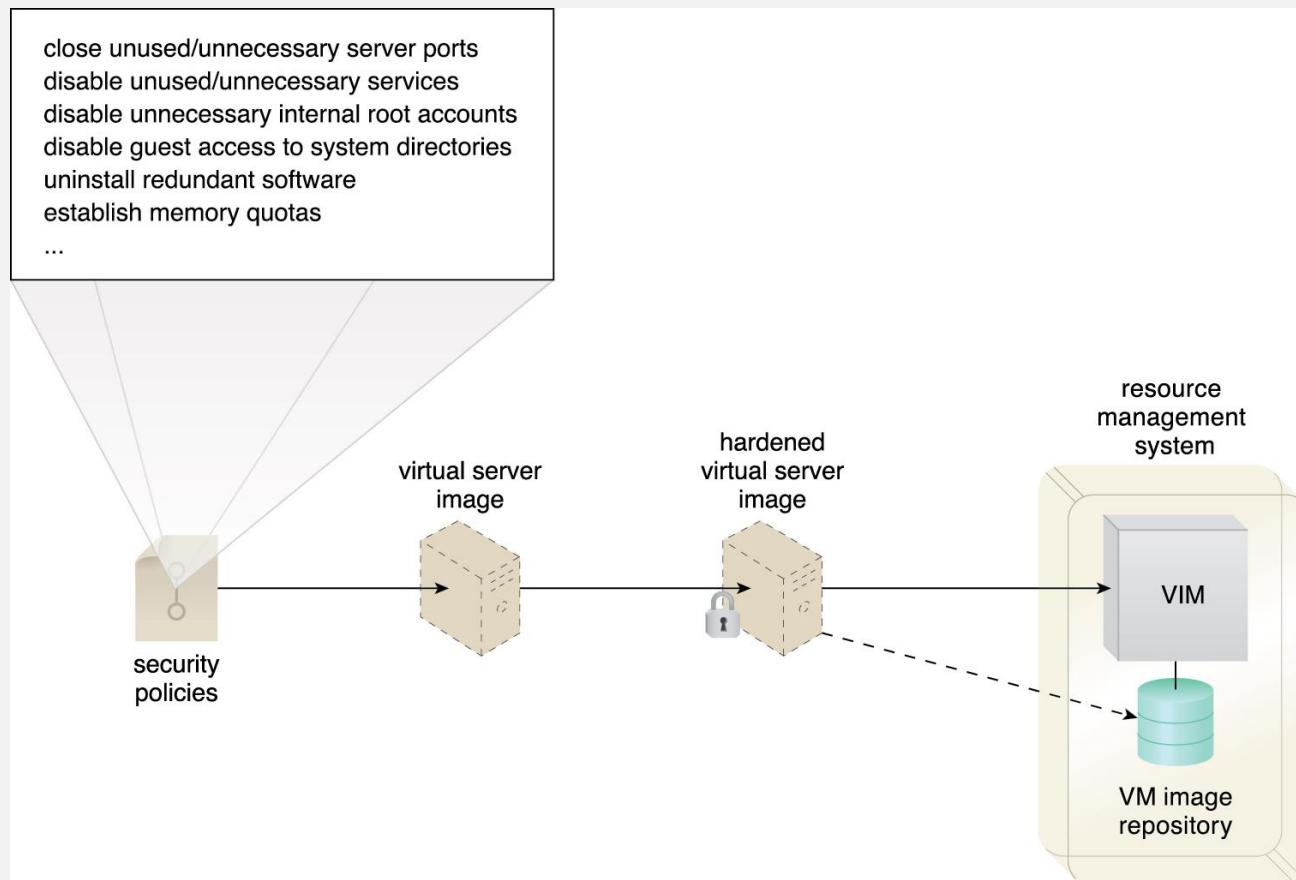
Single sign-on(SSO)

- Single sign-on is an authentication scheme that allows a user to log in with a single ID and password to any of several related, yet independent, software systems.
True single sign-on allows the user to log in once and access services without re-entering authentication factors.
- SSO is useful when cloud service consumer needs to access cloud services residing on different clouds.



Hardened virtual server images

- Usually, a virtual server is created from a template configuration called VM or VS image.
- Hardening is the process of stripping unnecessary software from a system to limit potential vulnerabilities that can be exploited by attackers.
- E.g. redundant programs, ports, unused services, internal root a/c, guest access



Thanks!



Cloud and big data technology

Selecting cloud providers

INF-2220

Dilip K. Prasad



Outline

- Overview
- Cloud service providers
- Considerations for selecting a cloud solution
- Business considerations
- Data safety and security
- Interoperability, portability and integration
- Service level considerations
- Pricing structure and commercial considerations
- Hosting considerations
- Geographical considerations



Outline

- Contingency and recovery management
- Ethical and legal considerations
- Scalability and flexibility
- Standard and best practices
- Cloud computing standards organizations
- Understanding the best practices
- Practical issues to be considered



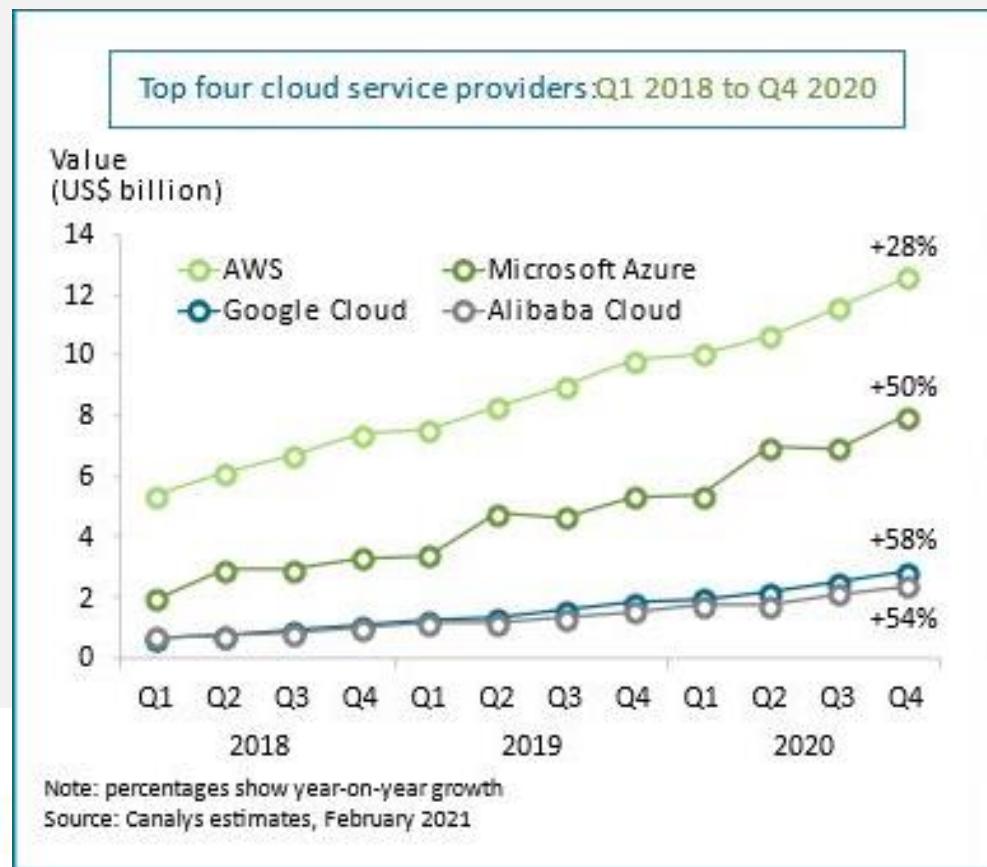
Overview

- Cloud adoption offers major IT and business benefits to any organization. On the other hand, cloud adoption poses some unique and serious security threats and complications.
- Moving the secure data of an organization into the hands of cloud service providers increases the risk of data security information theft, and hence the organization may lose its market position to its competitors

- The most fascinating fact about cloud computing for CIOs(Chief Information officers) is the unprecedented freedom that it brings in. Once the decision to embrace the technology has been made, organizations begin to enjoy the “no-commitment” working model of the cloud
- While measure pertaining to security, compliance and the like can be handles through the journey, the first step towards adopting the cloud starts with evaluating the market options for the right cloud provider.
- The cloud industry has a tantalizing array of cloud vendors offering SaaS, PaaS, IaaS solutions. Before we begin exploring the various criteria to be satisfied by cloud service providers, let us have a look at the topmost players in the industry.

Top cloud providers

- AWS
- Microsoft
- Google
- Alibaba
- IBM



Considerations for selecting cloud solution

- Business considerations
- Data safety and security
- Interoperability, portability and integration
- Service level considerations
- Pricing and commercials
- Hosting and geographical considerations
- Contingency and recovery management
- Ethical and legal considerations
- Scalability and flexibility considerations

Business considerations

- The cloud vendor must be technically advanced and be able to understand the specific business needs of the client.
- If you are an organization in a specific market vertical such as healthcare, banking, or retail, it is recommended to choose a cloud service provider with vast experience and expertise in the same industry.

Data safety and security

- Working with a trusted cloud service provider is the key to managing the various security and regulatory issue that arise within the cloud.
- The cloud service provider must address each of these criteria with solutions that are in adherence with the requirements of the business as well as the industry.

While choosing a cloud service vendor, organization must consider the following criteria:

- Regulatory compliance
- Segregation of data in multitenant environments
- Data recovery
- Access privileges (logical and physical)
- Portability of data for business continuity
- Data provenance
- Monitoring and reporting
- Network security

Service level considerations

- When determining the service level of a cloud vendor, there are essentially three factors to be considered, availability, performance and reliability
- Availability of the service is determined by the number of “nines” mentioned in the SLA. Cloud service providers generally promise their availability by a guaranteed uptime of 99.9% or 99.999% uptime for an entire year.
- 1 year = 525600 minutes ; 99.9% uptime means 525074.4 minutes uptime which means 525 minutes and 36 seconds downtime in a year.
- 99.999% uptime means 525594.744 minutes uptime which means 5 minutes and 15 seconds downtime in a year.
- While availability of the service is crucial, the speed at which it handles the business-critical operations has a direct impact on the business outcome.
- The reliability of a cloud vendor is determined through its transparency in operations.

Pricing structure and commercials

- The pricing offered by a vendor may depend upon several factors like security level, storage space and so on. In any case, the pricing must be flexible and must not carry any hidden costs.
- The pricing offered by a vendor may depend upon several factors like security level, storage space and so on. In any case, the pricing must be flexible and must not carry any hidden costs.
- A historical review of the prices offered by cloud service providers may offer some standard insights about their cost structure, and organizations must always ensure that the comparison made between different vendors is strictly apples-to-apples.

Geographical considerations

- Compliance and regulatory demands for data changes with geographical locations. Your cloud service provider must adhere to these changes and protect data in any given location.
- Your cloud service provider may carry your data across multiple locations in various geographic regions to mitigate risks such as localized outages, service latency and increased costs.
- Enterprises must learn about the different locations of their data as the laws governing the storage and use of data vary with different jurisdictions.
- Additionally, certain locations may also violate certain regulatory requirements and cause serious threats to data security.
- Organizations that wish to limit the geographic location on any of the above basis must practice care while choosing their cloud service provider.

Business process examples

- As important as it is to choose cloud service provider with the right Disaster Recovery (DR) solution, it is very critical to keep track if the DR functions as functioning as promised.
- Organizations are recommended to conduct annual audits to ensure that the DR process is in line with the evolving business.
- The cloud is very vulnerable, and disaster recovery for applications hosted in the cloud must not be assumed as an inherent feature in the architecture of the cloud host.
- Due diligence for DR must be implemented by the client organization just like it would be performed for their own in-house infrastructure.
- The cloud vendor must be prepared to share the documents that explain the various data protection solutions in detail. The client must look for the DR features enclosed within the base price and have a thorough understanding of the vendor's backup capabilities.

Cloud computing standards Organizations

1. National Institute of Standards and Technology (NIST)
2. [Cloud Security Alliance](#)
3. Open Grid Forum (OGF)
4. Object management Group (OMG)
5. Cloud Computing Interoperability Forum (CCIF)
6. Distributed Management Task Force (DMTF)
7. Storage Networking Industry Association (SNIA)
8. Open Cloud Consortium (OCC)
9. ENISA

Practical issues

Once you have chosen the right cloud vendor and equipped yourself with the right expertise for the cloud, here is a list of practical issue to be remembered to make the cloud journey as smooth as possible.

- Proper negotiation of SLA
- Vendor Lock-in
- Change in organizational culture
- Compliance and security of data
- Commercial implication of shipping large volumes of data
- Integration of the cloud into the existing system

Thanks!



Cloud and big data technology

Cloud Technologies

INF-2220

Dilip K. Prasad



Outline

- Overview
- High Performance Computing (HPC)
- Message Passing Interface (MPI)
- Dryad
- Eucalyptus
- OpenNebula
- OpenStack
- Nimbus



High performance computing

- For solving data intensive problem cloud computing is a first choice due to its ability to handle large datasets and better quality of services.
- Parallel Virtual Machine(PVM) and Message Parsing Interface(MPI) communication protocols are used by the high-performance computing(HPC).

Parallel Virtual Machine(PVM)

- Parallel Virtual Machine (PVM) is a software tool for parallel networking of computers.
- It is designed to allow a network of heterogeneous Unix and/or Windows machines to be used as a single distributed parallel processor.
- Large computational problems can be solved more cost effectively by using the aggregate power and memory of many computers.

Message Passing Interface(MPI)

- MPI is a HPC library used to establish information flow between various nodes and clusters.
- It allows programmers to focus on algorithms rather than communications overheads between nodes and clusters.
- OpenMPI is widely used open source MPI library.
- pyMPI is opensource software which integrate MPI into python interpreter

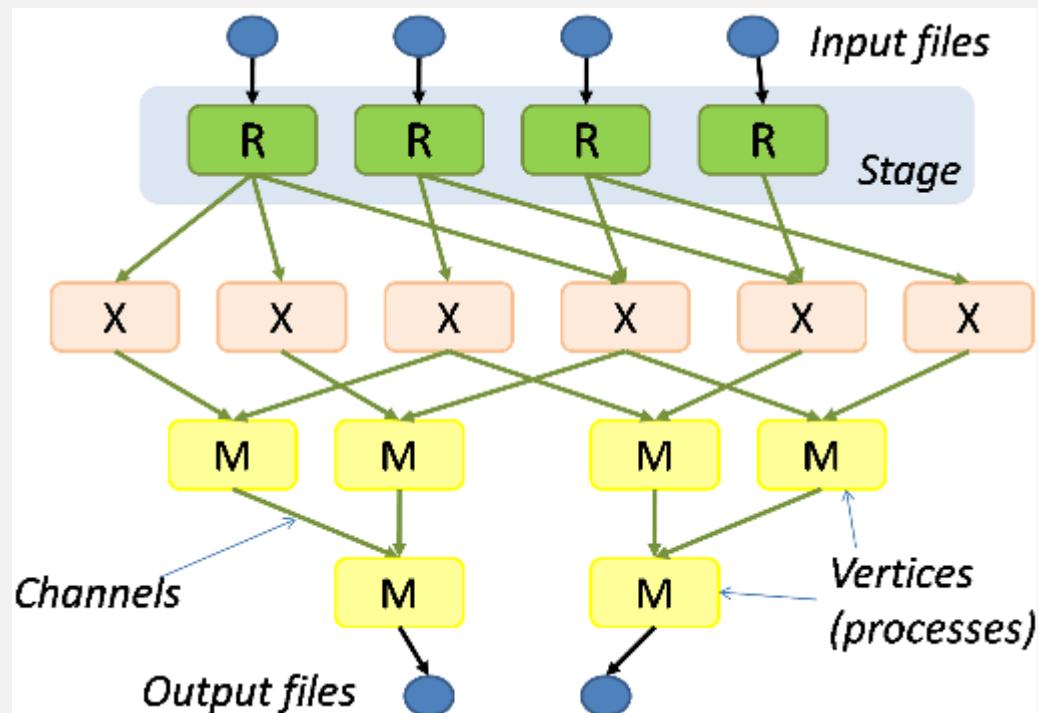
```
$ mpirun -np 3 pyMPI
> import mpi
> print "Hi, I'm process #%" % mpi.rank
```

```
Hi, I'm process #0
Hi, I'm process #1
Hi, I'm process #2
```

Dryad & DryadLINQ

- Dryad is an infrastructure which allows a programmer to use the resources of a computer cluster or a data center for running data-parallel programs.
- A Dryad programmer can use thousands of machines, each of them with multiple processors or cores, without knowing anything about concurrent programming.
- Dryad consider computation task as directed acyclic graph(DAG), where the vertices represent computation tasks and the edges act as communication channels over which the data flows from one vertex to another
- It is optimized for data locality rather than CPU utilization to support disk I/O jobs

<https://www.microsoft.com/en-us/research/project/dryad/>



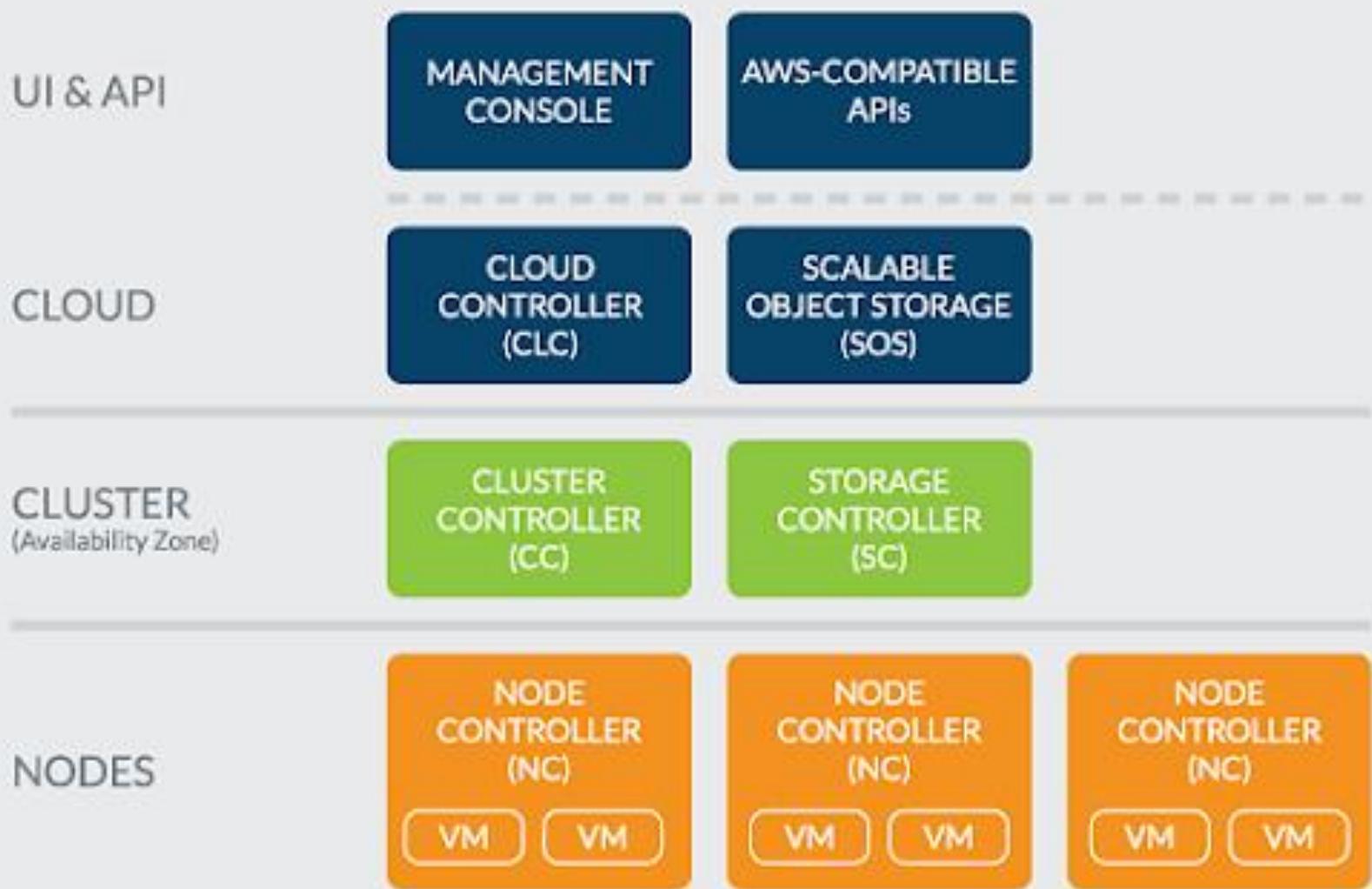
Eucalyptus cloud platform

- Its open-source, AWS compatible cloud computing environment for both private and hybrid cloud. It allows sites with existing clusters and server infrastructure to host a cloud.
- **Hybrid Cloud Management** - Launch instances, create snapshots and manage autoscaling groups in either your private or public clouds from a single environment.
- **AWS Compatibility** - Eucalyptus provides industry-leading compatibility with popular Amazon Web Services (AWS) APIs including EC2, S3, Elastic Block Store (EBS), Identity and Access Management (IAM), Auto Scaling, Elastic Load Balancing (ELB), and CloudWatch.
- **Compute** - Eucalyptus allows you to use industry-standard servers, storage, networking, and virtualization technologies to deliver cost-effective, AWS-compatible cloud services in your datacenter. Eucalyptus is compatible with AWS's EC2 and allows you to easily deploy compute resources and efficiently increase or decrease compute capacity based on application demands.

Component of Eucalyptus

- **Cloud controller**- It is the main controller(CLC), which manages the entire cloud platform. It also provide web and Amazon EC2 compatible interface. It performs scheduling resource allocation and accounting.
- **Cluster controller(CC)**- It manages VM(instance) execution and SLA. It communicates with storage and network controller.
- **Storage Controller(SC)**- It is similar to AWS Elastic block Storage.
- **Node controller (NC)**- It hosts all the instances and manages their end points.

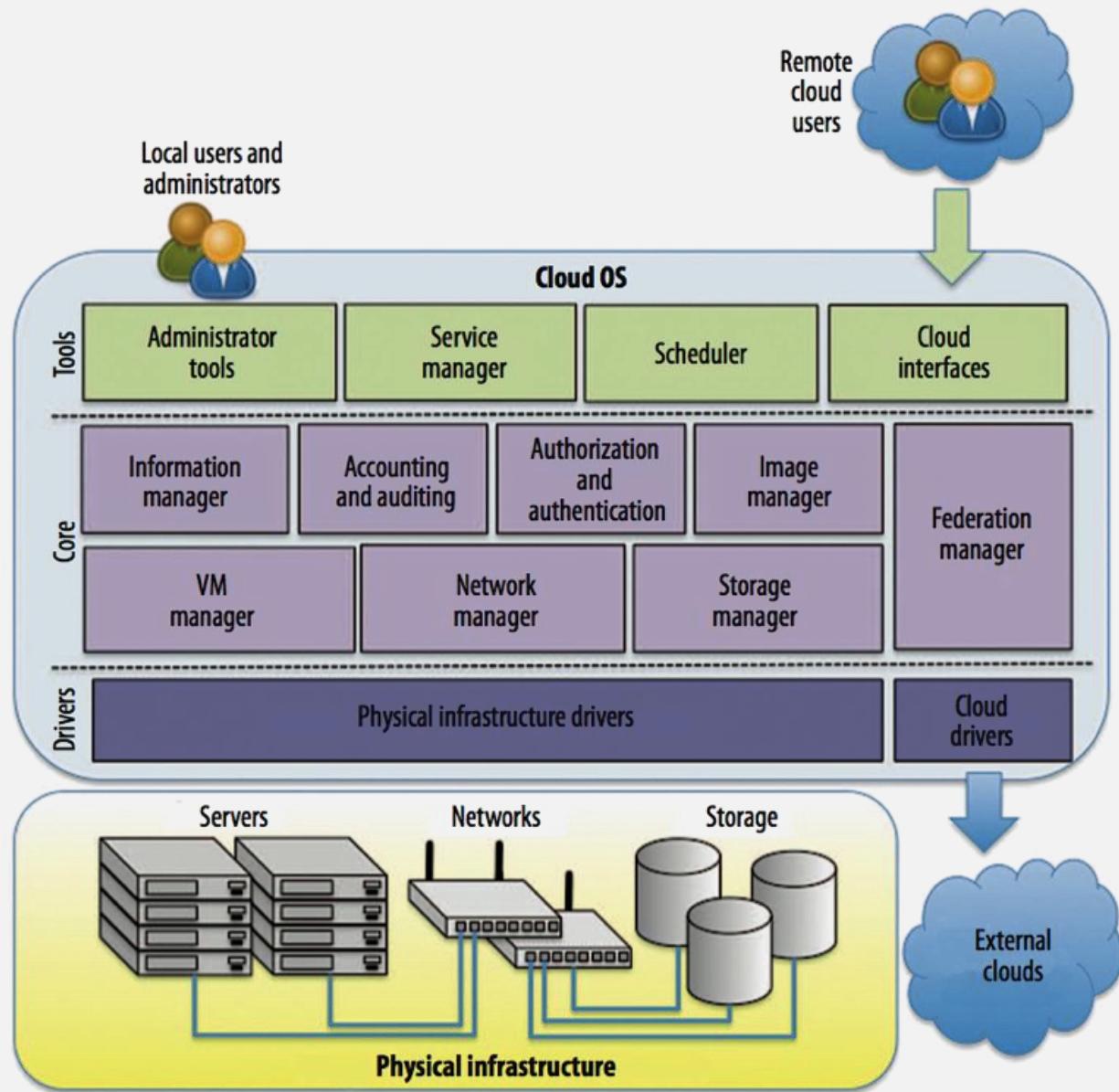
Component model of Eucalyptus cloud



OpenNebula Cloud Platform

- It's an open-source cloud service framework used to manage heterogeneous distributed data center infrastructures.
- It is full fledged solution to build private clouds and manage data center virtualization.

OpenNebula Architecture



Layers of OpenNebula

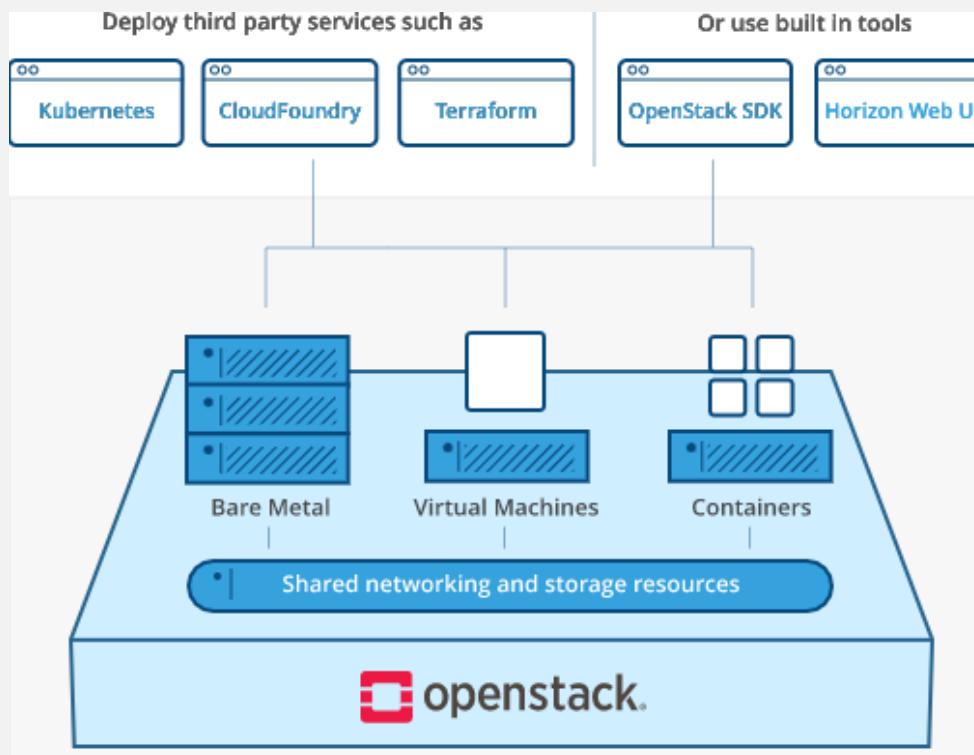
- Data center virtualization management - This layer is used to manage data center virtualization, consolidate servers, and integrate existing IT assets for computing, storing and networking. It integrates directly with Hypervisor.
- Cloud management - This layer is used to provide a multi-tenant, cloud-like provisioning layer on top of an existing infrastructure management solution.

Features of OpenNebula

- **Cloud Bursting** - Extension of local private infrastructure with resources from remote clouds
- **On demand provision of virtual data centers** - A virtual data center is a fully-isolated virtual infrastructure environment where a group of users can create and manage computing, storing and networking capacity
- **Multiple zones** - Centralized management of multiple instances of OpenNebula(zones) for scalability, isolation or multiple-site support.
- **Multi-VM application management** - Automatic execution of multi-tiered applications with auto-scaling.

OpenStack Cloud Platform

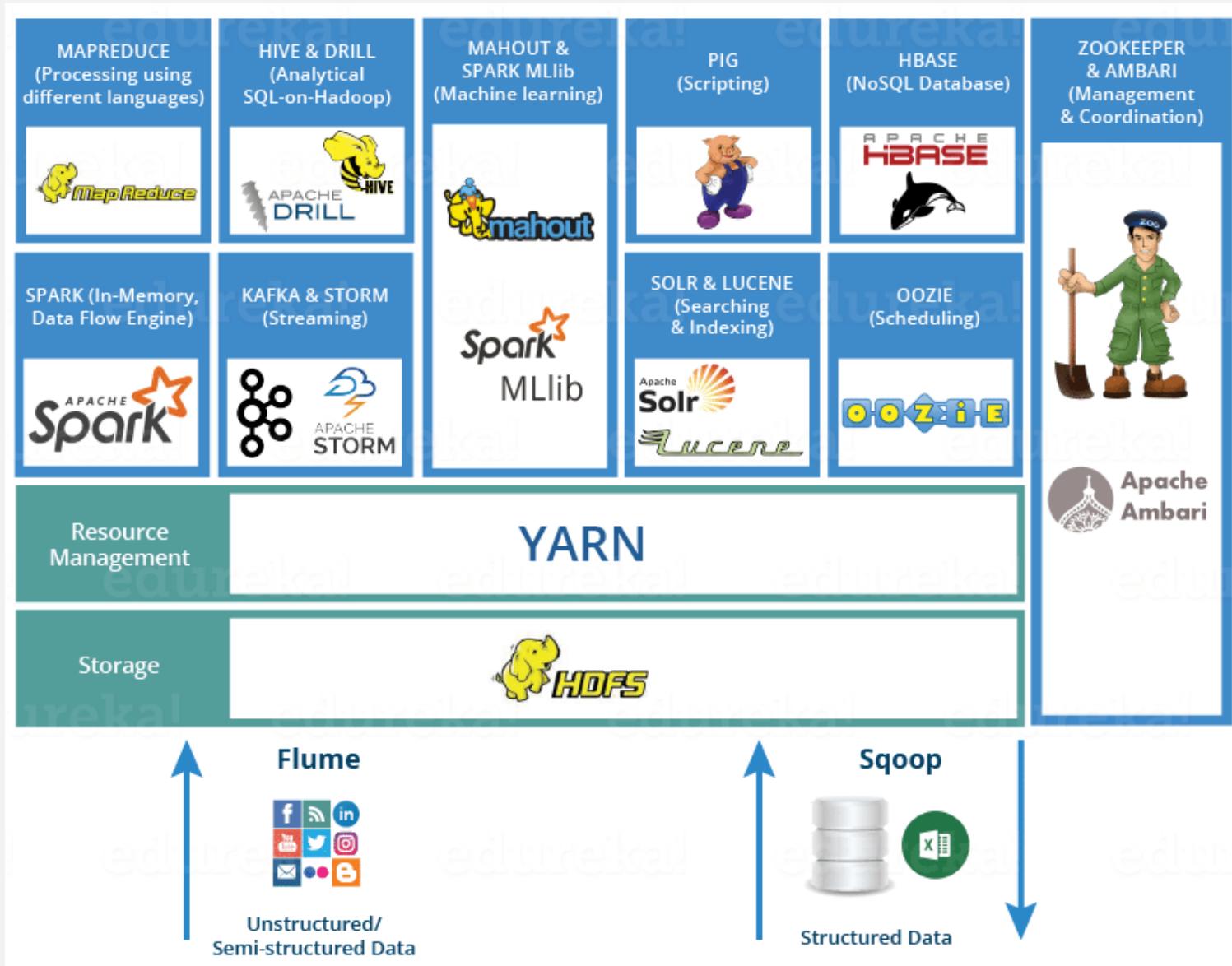
- It is a free and open source software platform for IaaS cloud.
- It provides a modular cloud infrastructure that allows you to deploy the tools of your choice when you need them, using a single dashboard based on the OpenStack API



Nimbus Cloud Platform

- Opensource IaaS solution
- Remote deployment and lifecycle management of VMs
- Compatibility with Amazon N/w protocols
- Multiple credential protocol support
- Local resource management, XEN & KVM plugins
- Nimbus platform- It is a versatile integrated set of tools that provides infrastructure services like deployment, scaling cloud resource management, etc. Its utility is in ability to combine itself with other cloud platforms like OpenStack, AWS, Azure etc.
- Nimbus Infrastructure -

Apache Hadoop Ecosystem



Thanks!



Cloud and big data technology

Virtualization, Hypervisor, VMware

INF-2220

Dilip K. Prasad



Outline

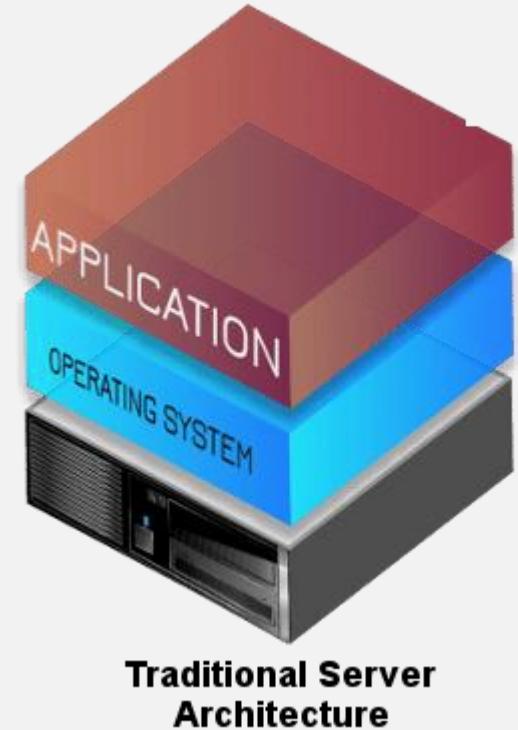
- Traditional Architecture
- Virtualization
- Virtual Architecture
- Hypervisor
- Types of Virtualization
- ESXi
- vCenter client
- vSphere server



Traditional Architecture

Traditional architecture has inherent challenges:

- Poor use of physical resources
- High management and maintenance costs
- High Physical infrastructure costs
- Provisioning challenges
- Insufficient failover and poor disaster protection



What is virtualization?

- It is the technique of splitting a physical resources into as many logical resources as we want. E.g CPU, memory

Or

- Virtualization is a technology that transform hardware into software.

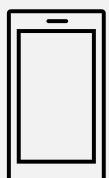
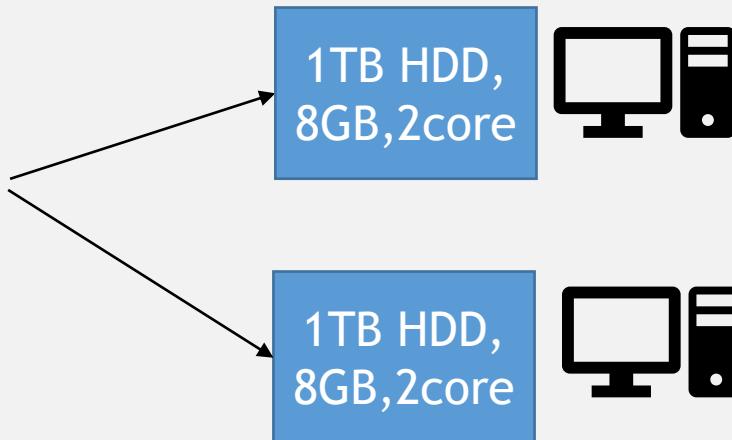
What is virtualization?



40k
NOK

i7 8700K processor
32 GB RAM
512 GB SSD
2TB HDD
6core

Computer



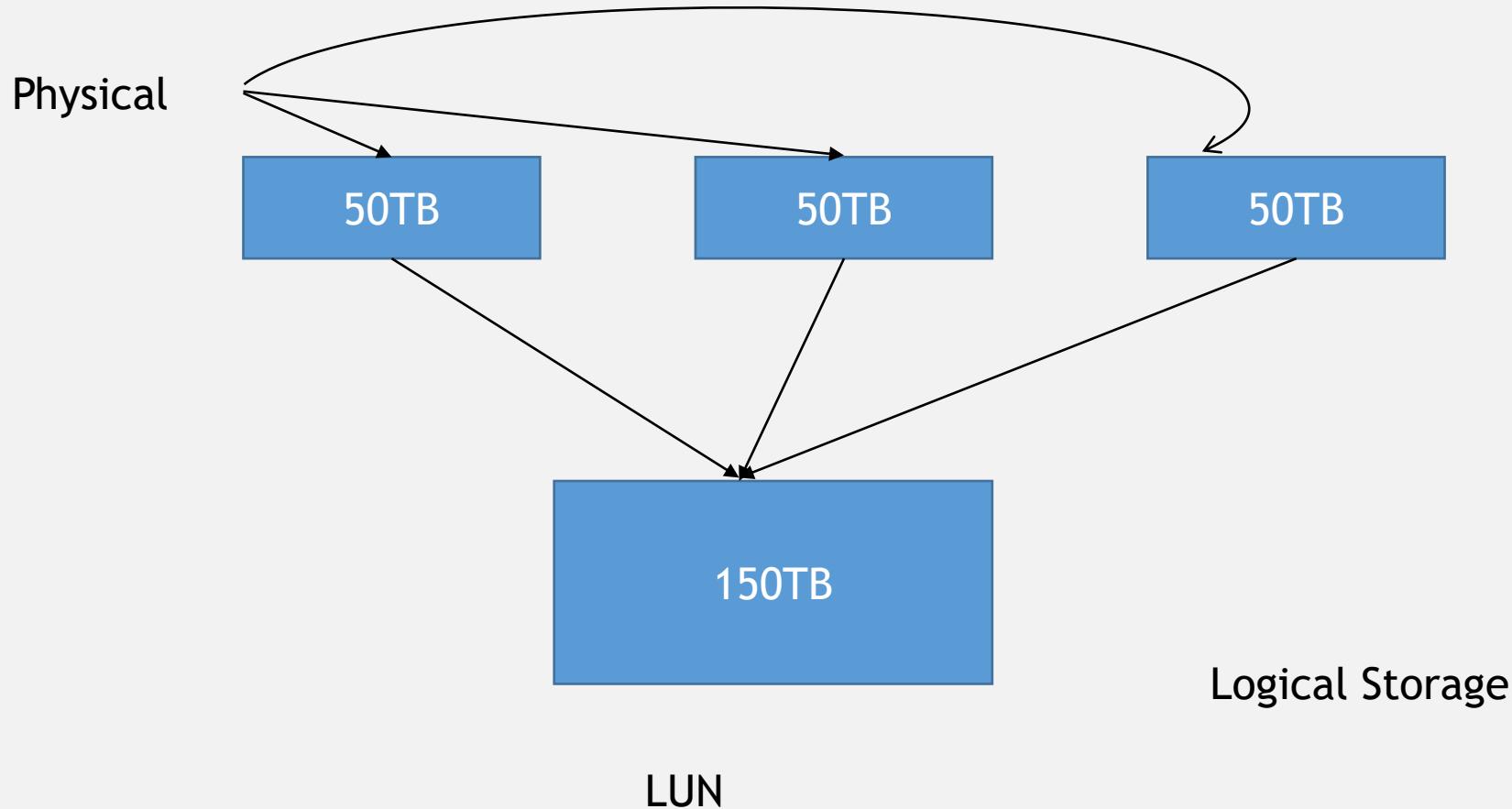
16 GB memory card
32 GB internal memory

Phone

40 GB
movie file

Can we transfer this 40
GB file to phone?

Virtual storage concept

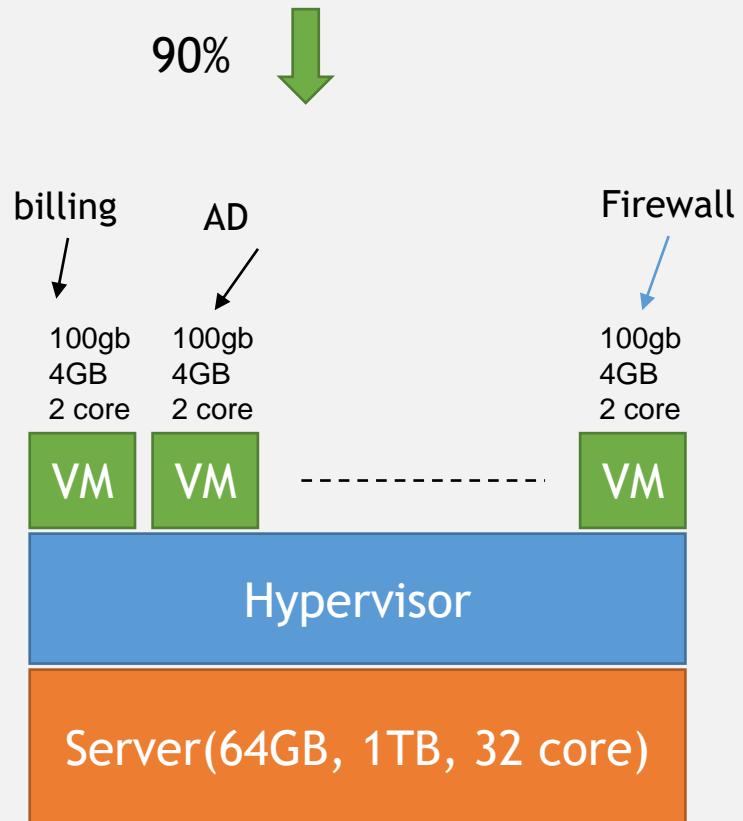


Case study

- Data centre setup (cost 2 million)
- 2000 sq. ft. datacentre
- 100 physical server (30 for Infra , 70 for opp)
 - (4GB, 2 CPU, 60GB HDD)
- Cooling system for 100 servers
- 100 Physical disk for server
- Cabling for 100 servers & power backups
- DC staff (operation)
- Electricity & power backup (operational)

Case study

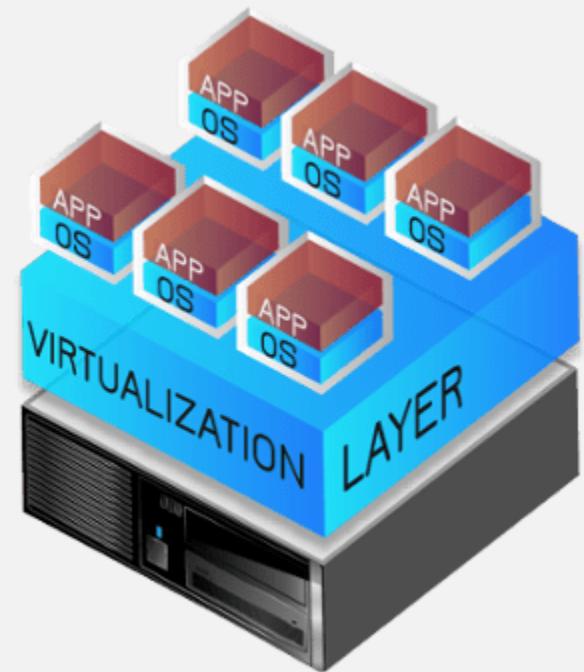
- Data centre setup (cost 2 million)
- 2000 sq. ft. datacentre
- 100 physical server 35 for infra 70% for app
 - (4GB, 2 CPU, 60GB HDD)
- Cooling system for 100 servers
- 100 Physical disk for server
- Cabling for 100 servers & power backups
- DC staff (operation)
- Electricity & power backup (operational)



Virtual Architecture

Virtual architecture has its inherent benefits:

- Expanded use of physical resources
- Reduced management and maintenance costs
- Improved desktop manageability and security
- Increased availability of applications
- Increased operational flexibility



Virtualized Server Architecture

Virtual Machine

- A virtual machine is a software representation of a physical computer and its components.
- The virtualization software converts the physical machine and its component into files.
- Components of Virtual machine
 - Operating system
 - Virtualization s/w(VMWare, HyperV, etc.)
 - Virtual resources like
 - CPU and memory,
 - Network adapters, disks and controllers, Parallel and serial ports



Comparative benefits of Virtual Machine

Physical Machine

- Difficult to move or copy
- Bound to a specific set of hardware components
- Often has short lifecycle
- Requires personal contact to upgrade hardware

Virtual machine

- Easy to move or copy
- Encapsulated into files
- Independent of physical h/w
- Easy to manage
- Isolated from other VM running on same physical h/w
- Insulated from physical h/w changes

Difference between physical server and virtual server

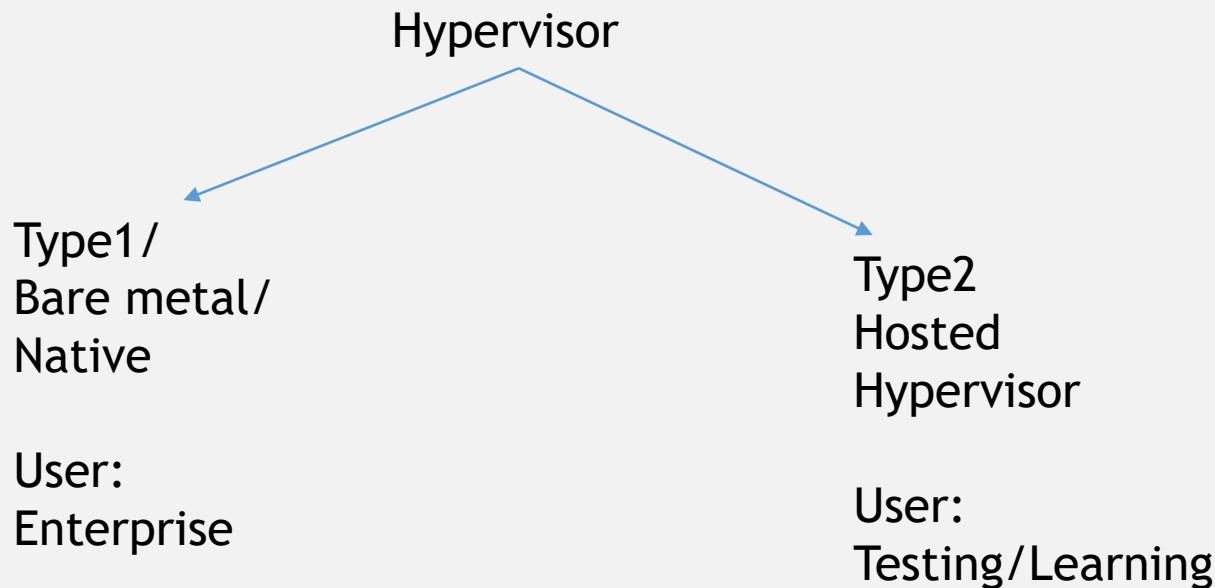
- High Availability
- Fault tolerance
- Low cost
- Capex+Opex reduction (capital expenditure and operational expenditure)

Hypervisor



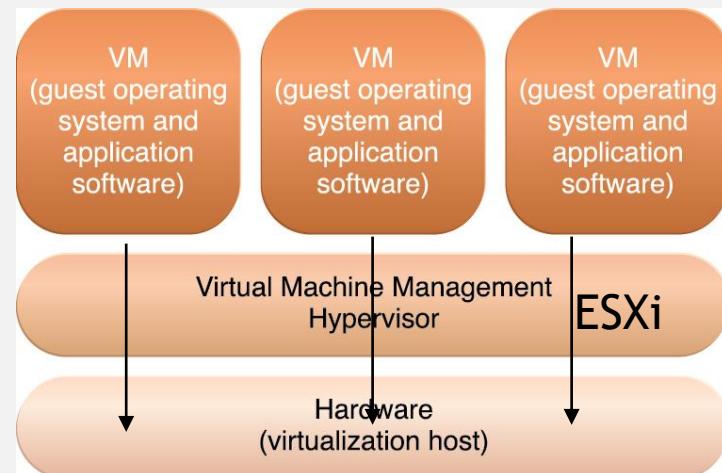
Hypervisor

- Hypervisor is a software or a firmware that creates and run virtual machine. A hypervisor is sometime also called as a Virtual machine Manager (VMM).



Type 1 - Hypervisor

- Also called Bare Metal Hypervisor (without OS). Type-1 hypervisor run directly on the system hardware. A guest OS runs on another level above the Hypervisor.
- VMware ESXi is a Type-1 hypervisor that runs on the host server hardware without an underlying operating system.
- Type-1 hypervisor is a firmware.
- It can host a guest operating system like windows/linux/Mac, etc.



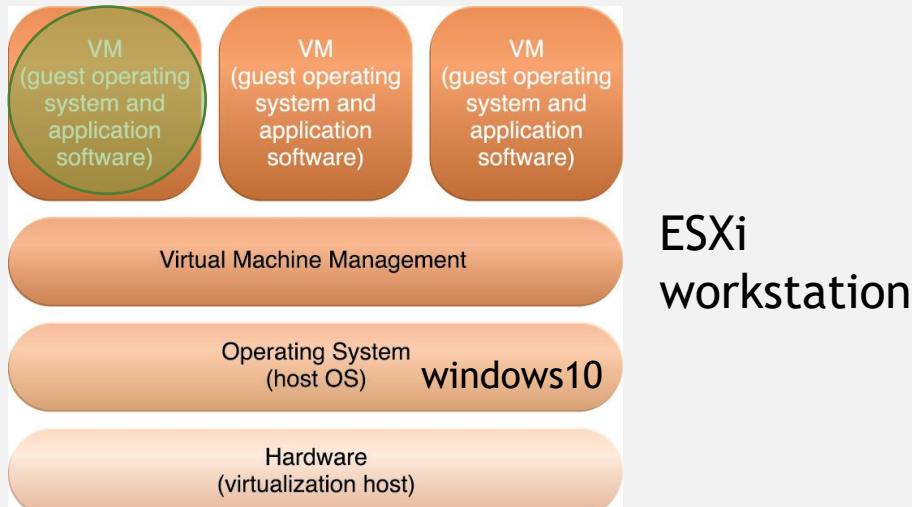
Type-2 Hypervisor

Type-2 hypervisor runs within a conventional operating system environment, and the host OS provides

Example of Type-2 hypervisor are VMWare workstations, Oracle VirtualBox, Microsoft Virtual PC, etc.

It does not have direct access to the host hardware and resources.

Type-2 hypervisor is a software. It's a hosted type of hypervisor.



Comparison of Type-1 and Type-2

Criteria	Type-1 Hypervisor	Type-2 Hypervisor
Also knows as	Bare Metal or Native	Hosted
Virtualization	Hardware virtualization	O S virtualization
Operation	Guest OS and application run on the hypervisor	Run as an application on the host OS
Scalability	Better scalability	Not so much because of its resilience on the underlying OS
System dependence	Has direct access to hardware along with virtual machine it host	Are not allowed to directly access the host hardware and its resources
Performance	Higher performance as there is no middle layer	Comparatively has reduced performance rate as it runs with additional overhead
Security	More secure	Less secure, as any problem in the base OS affect the entire system
Examples	VMWare ESXi, Hyper-V, Xenserver	VMWare workstation, Oracle Virtual Box

Types of Virtualization

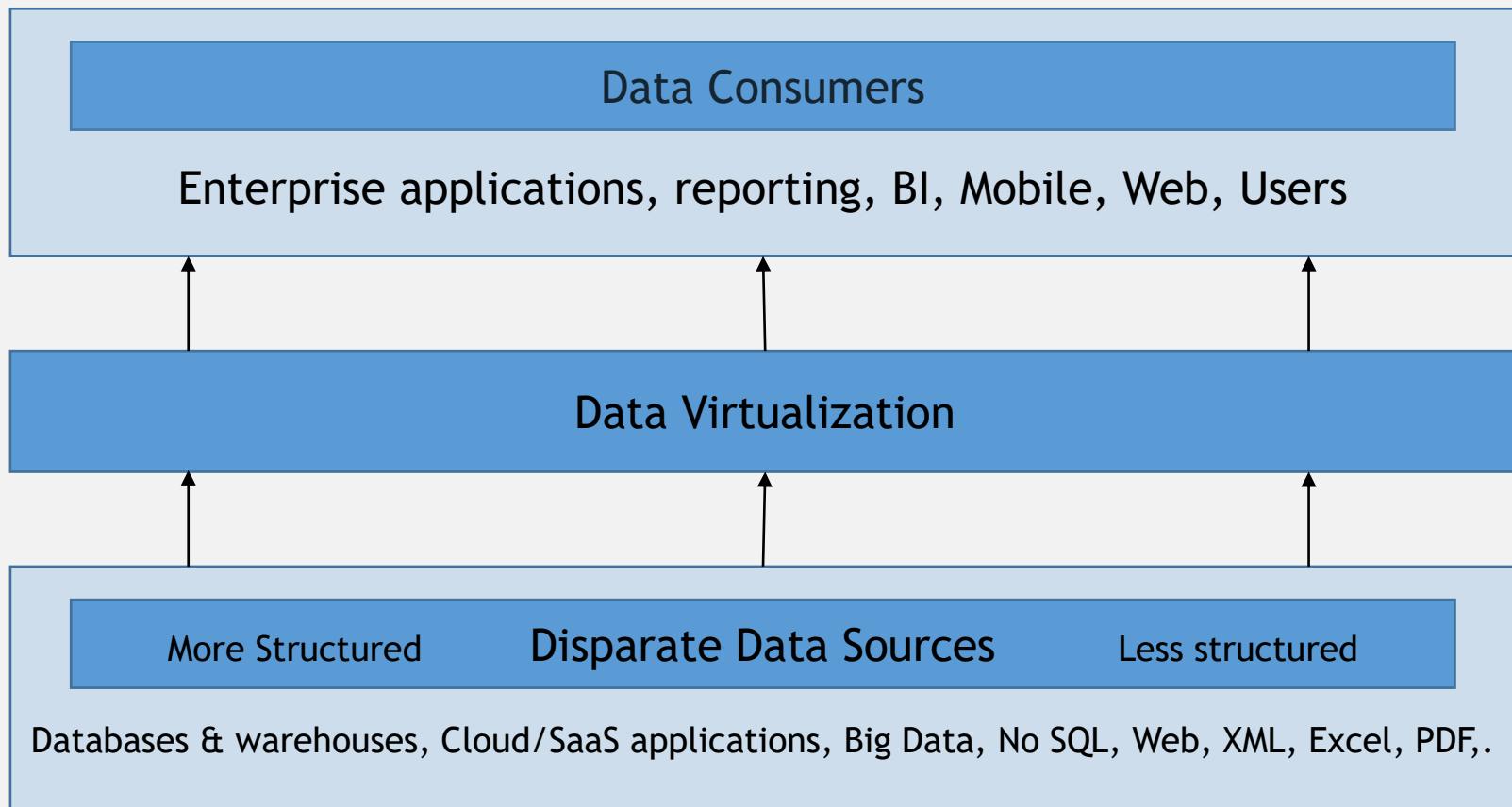


Types of Virtualization

- Data Virtualization
- Desktop Virtualization
- CPU Virtualization
- Network Virtualization
- Storage Virtualization
- Server Virtualization

Data Virtualization

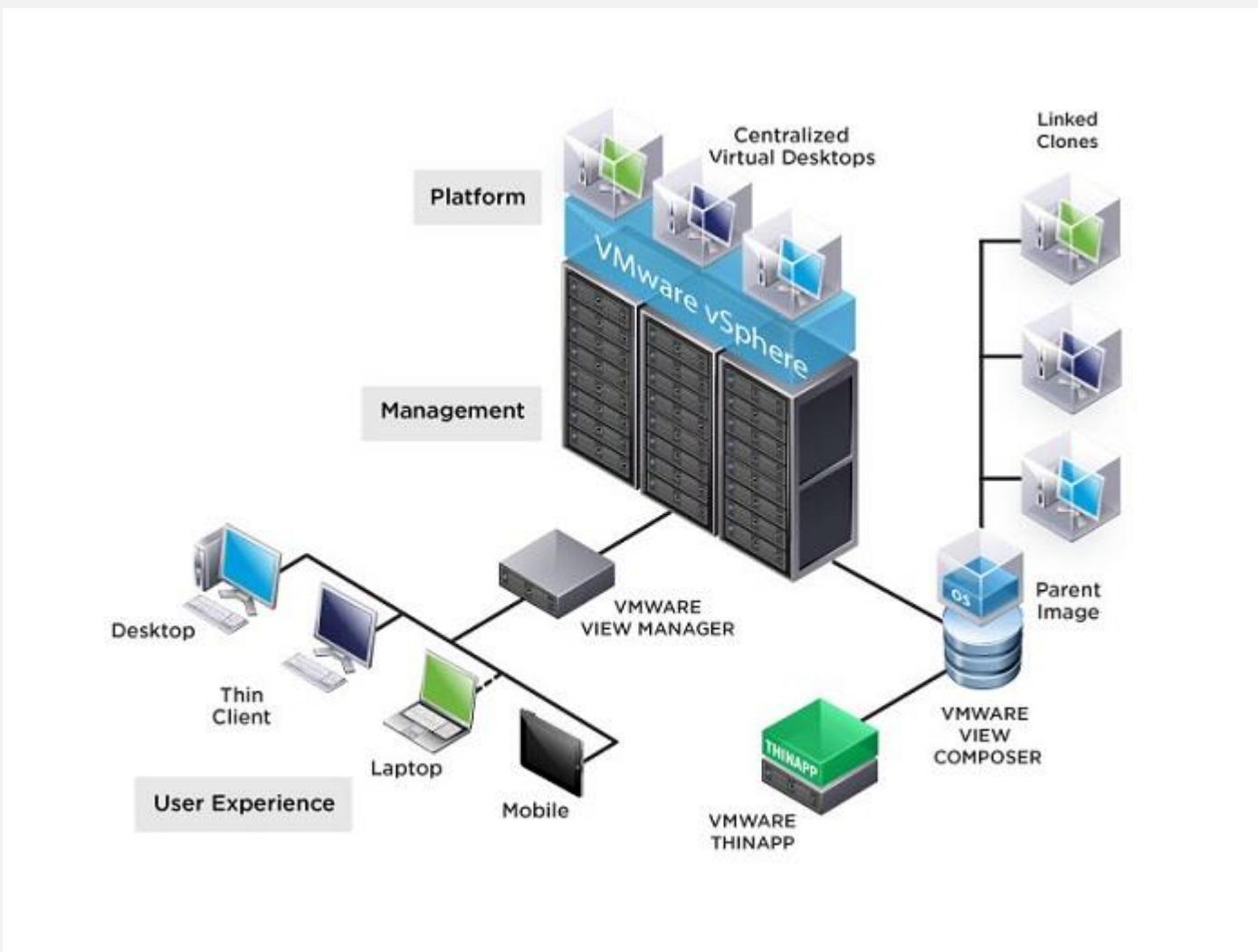
- Data Virtualization is analogous to data agility in which an application is allowed to access data irrespective of its technical details, formatting style and physical location.
- Example tools are Red Hat Jboss, Denodo, etc



Desktop Virtualization

Desktop virtualization is a technique in which virtualization layer runs on top of a hypervisor and provides desktop provisioning.

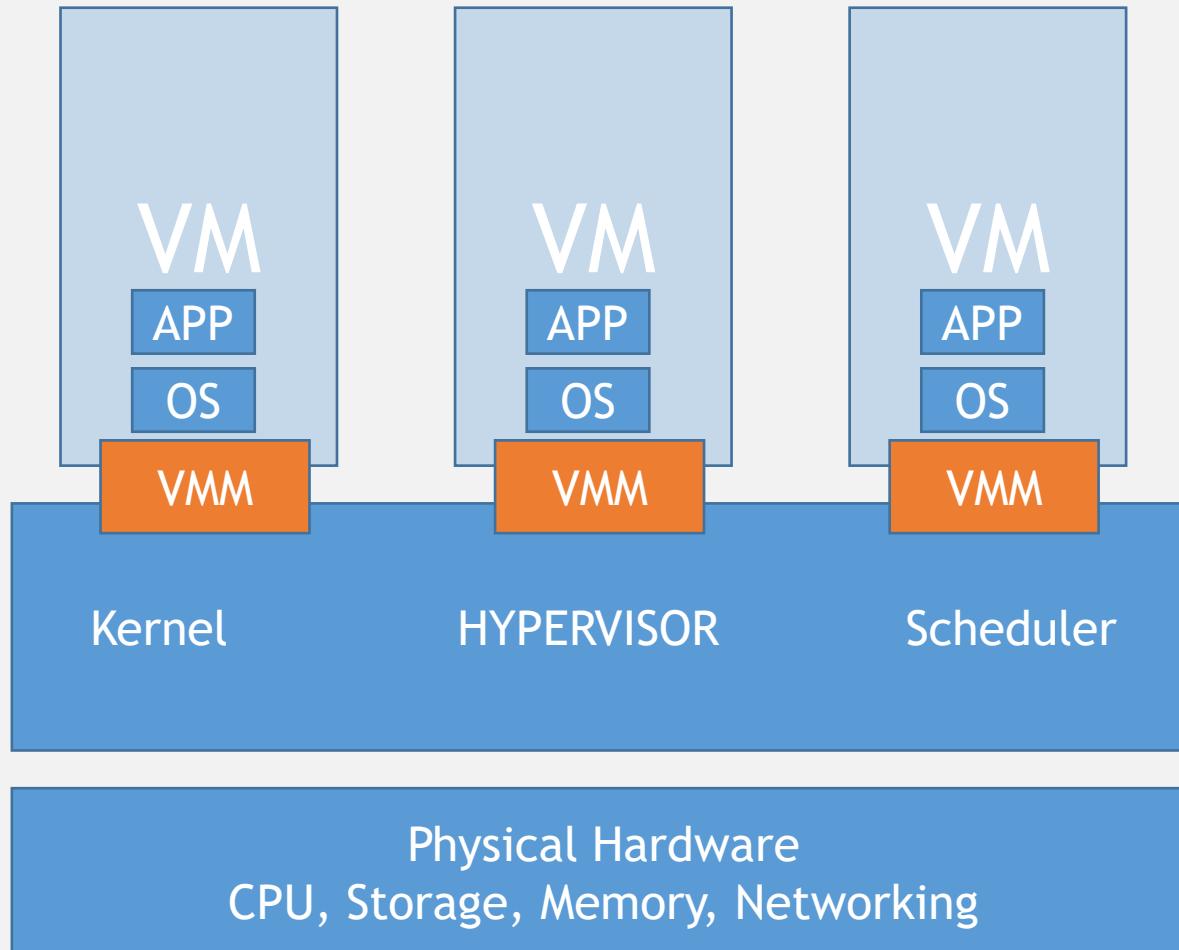
Desktop Virtualization



CPU Virtualization

CPU virtualization allows a single processor to act as a multiple individual CPUs, that is , two separate systems running on a single machine. The objective of CPU virtualization is to allow the user to run different operating system simultaneously.

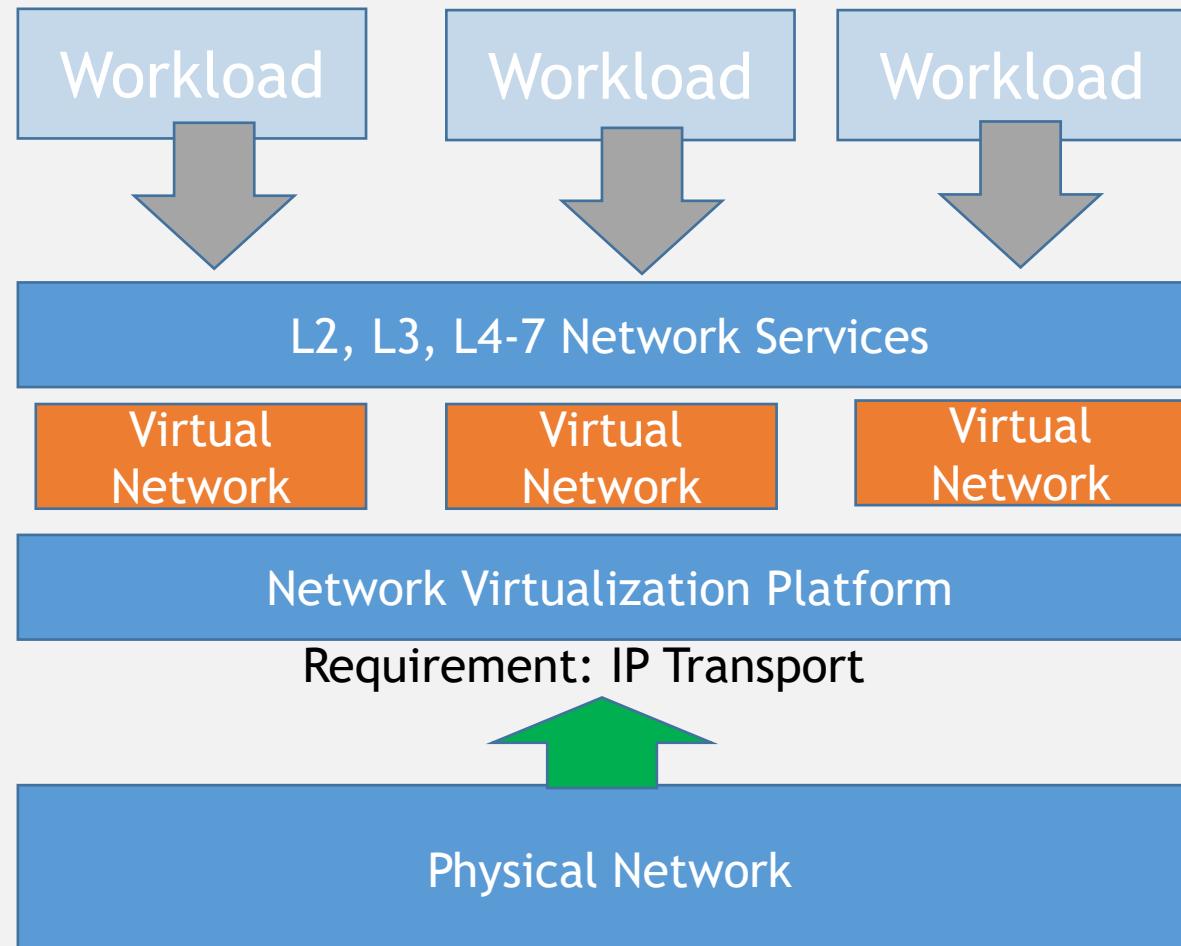
CPU Virtualization



Network Virtualization

- Creates virtualized combination of available resources by splitting up the bandwidth into channels so that each device or user can have shared access to all the resources on the network.
- In Network virtualization, hardware and software network resources and their functionalities are encapsulated into a software based administrative entity.

Network Virtualization



Storage Virtualization

- Assembly of physical storage from heterogenous storage devices to form a large pool of memory resources, which is managed centrally.
- Storage area network (SAN).
- SAN is different from Network area storage(NAS)

Server Virtualization

- Hypervisor or VMM
- Paravirtualization -
 - para-virtualization is a virtualization technique that presents a software interface to the virtual machines which is similar, yet not identical to the underlying hardware-software interface.
 - paravirtualized platform may allow the virtual machine monitor (VMM) to be simpler (by relocating execution of critical tasks from the virtual domain to the host domain)

VMWare

- ESXi
- vCenter Server
- vSphere Client

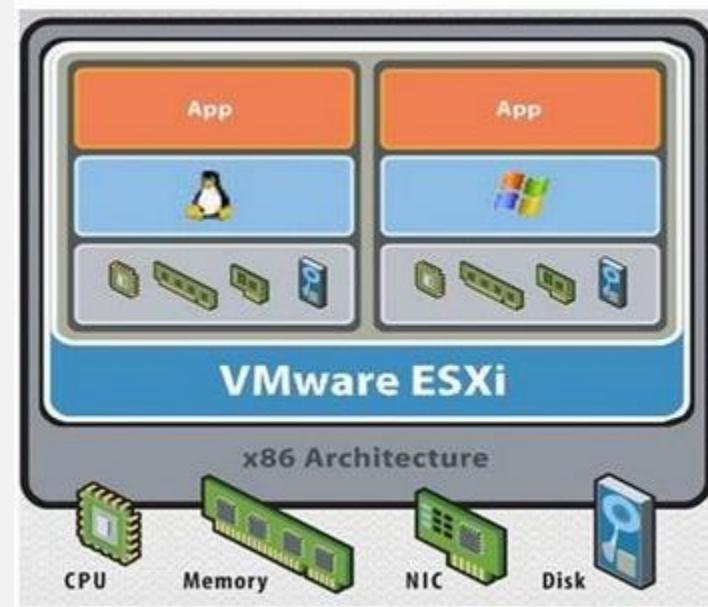


ESXi

ESXi provides a virtualization layer that abstracts the CPU, storage, memory and networking resources of the physical host into multiple virtual machine.

ESXi- Elastic Sky X integrated

The architecture of ESXi is ultra-thin that makes it highly reliable and due to its small code base it is highly secure as there are fewer codes to patch. In place of service control, the ESXi uses DCUI which is direct console user interface to manage the ESXi server.



ESXi

It has its own kernel vmkernel. vmkernel is a microkernel with three interface; hardware, guest systems, and the service console (console OS).

Vmkernel handles its CPU and memory directly using scan-before-execution (SBE) to handle special or privileged CPU instructions and system resource allocation table (SRAT) to track allocated memory.

ESXi limitations

Infrastructure limitations

Some maximums in ESXi Server 7.0 may influence the design of data centers:

Guest system maximum RAM: 24 TB

Host system maximum RAM: 24 TB

Number of hosts in a high availability or Distributed Resource Scheduler cluster: 96

Maximum number of processors per virtual machine: 768

Maximum number of processors per host: 768

ESXi limitations

Maximum number of virtual CPUs per physical CPU core: 32

Maximum number of virtual machines per host: 1024

Maximum number of virtual CPUs per fault tolerant virtual machine: 8

Maximum guest system RAM per fault tolerant virtual machine: 128 GB

VMFS5 maximum volume size: 64 TB, but maximum file size is 62 TB -512 bytes

Maximum Video memory per virtual machine: 4 GB

Performance limitations: Overhead (5%) used for virtualization

Network limitations

VMWare Vsphere

1. VMWare is the market leader in server virtualization. It has 77% of the market share.
2. It is better than Citrix Xenserver and Microsoft hyper-V due to stability and flexible usage.
3. vSphere 7.0 - VMWare vSphere is the brand name for VMWare suite of virtualization products
4. vSphere include following components
5. VMWare ESXi
 1. vCenter Server
 2. vSphere Client
 3. vSphere web client
6. Other features includes - vmotion, high availability, fault tolerance, distributed resource scheduler

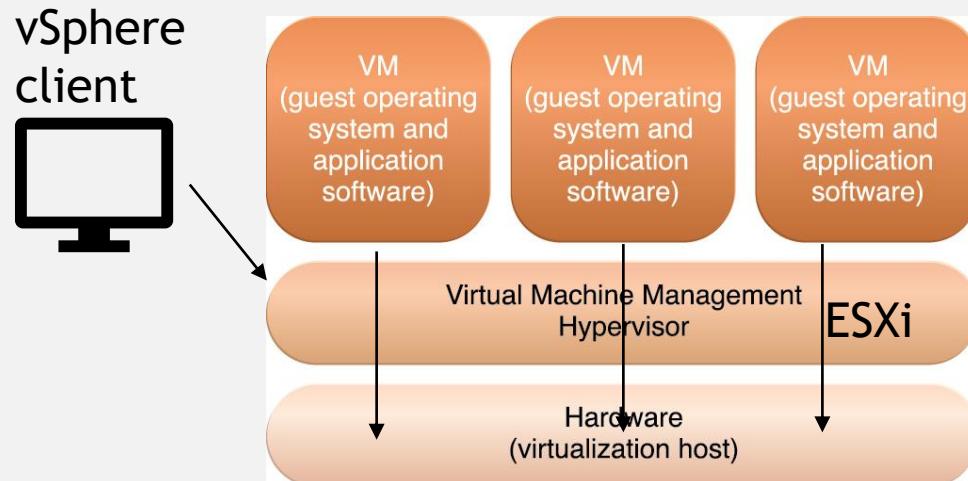
Difference between vcenter Server and vSphere Client

Vsphere client is an interface (GUI) used to connect remotely to an ESX/ESXi host from window PC.

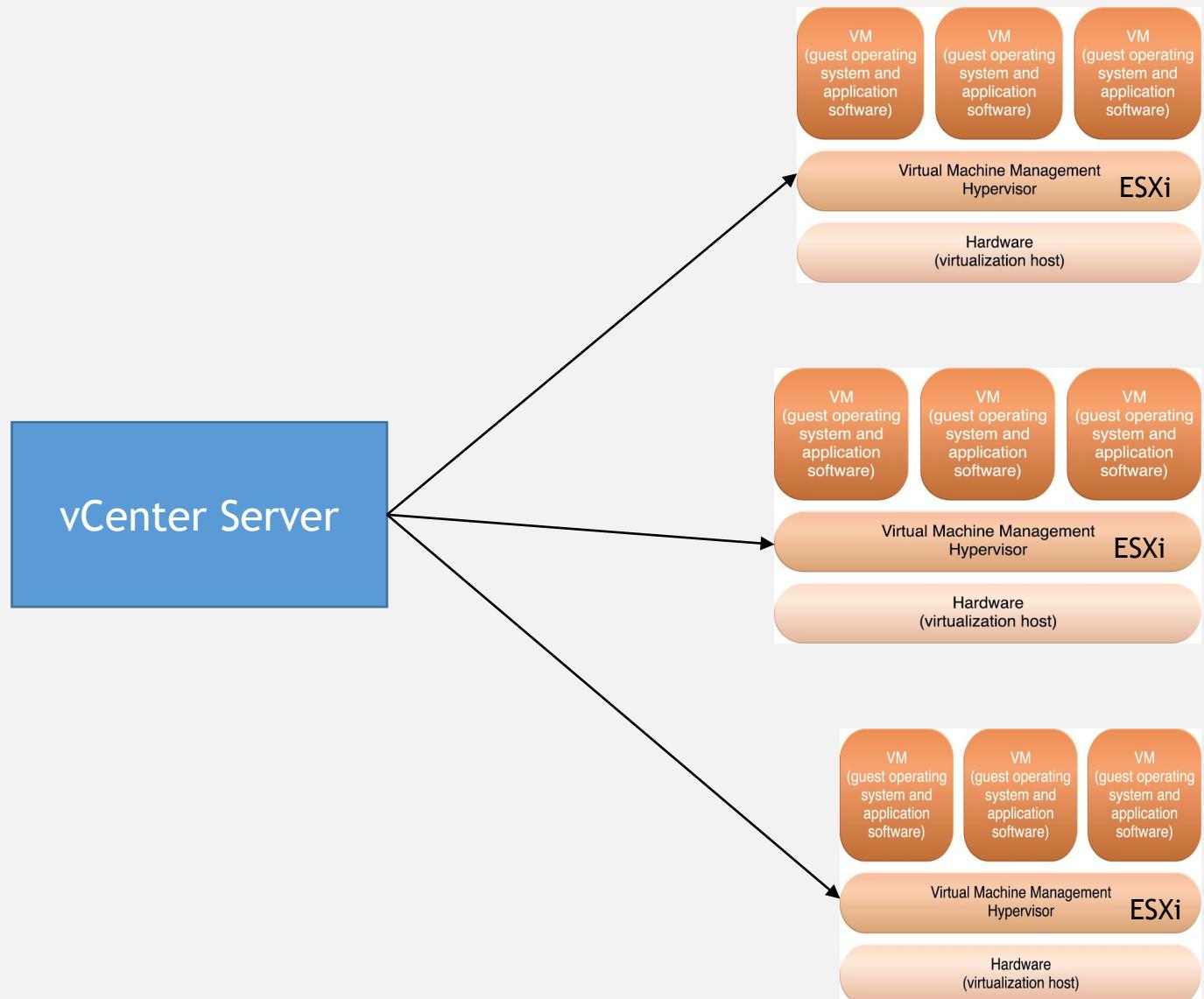
This client can be used to access and manage virtual machine on the ESXi host and also perform other management and configuration task.

If we want to have all the ESXi host in a single console then we need vCenter Server.

Difference between vcenter Server and vSphere Client



Difference between vCenter Server and vSphere Client



vSphere

- vCenter Server is similar to vSphere Client but it comes with rich features and more powers. It is the centralized management tool.
- Multiple ESXi host and VM can be managed from a single console, whereas using vSphere Client we were accessing only a single host.
- For using feature like DRS, High availability, vmotion and fault tolerance we need vcenter Server
- vcenter 7.0 includes flash and html5 based interface

VMWare availability

Availability or high availability in VMWare deals with ESXi host failure and what happens to the VMs running on those host?

Availability is to make sure that downtime is very less in the event of failure and machines are always up.

Level of agreement	Downtime per year
99%	87 hours
99.9%	8.76 hours
99.99%	52 minutes
99.999%	5 minutes

Thanks!



Cloud and big data technology

Machine Migration, High Availability, Fault Tolerance

INF-2220

Dilip K. Prasad



Outline

- Machine Migration
- High Availability
- Fault Tolerance



Machine migration to the cloud

- Cold migration
 - Suspended migration
 - Vmotion
 - P2V migration
 - V2V migration



Migration

- Migration is a technique of moving a Virtual machine from one host to another host or from one Datastore to another Datastore.
- Datastore stores virtual machine files, log files, virtual disk and ISO image.
- Two types of datastores are there - Virtual machine file system(VMFS) and Network file system (NFS).

Types of migration

- Cold migration
- Suspended Migration
- Vmotion
- Physical to virtual(P2V) - Physical machine to virtual machine
- Virtual to Virtual(V2V) - virtual machine to virtual machine on a different virtualization platform like Microsoft hyperV to Vmware etc

Cold migration

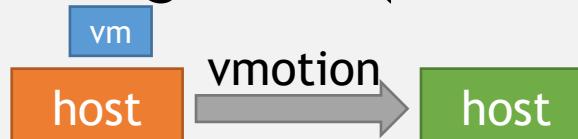
- Movement of virtual machine to another host in power-off state
- VM must be powered-off during migration
- Cold migration are flexible than vmotion
- Cold migration can be used to move a virtual machine between data centers, as long as both data centers are on the same vCenter server instance
- Chances of failure is less in cold migration, in comparison to hot migration

Suspended migration

- Migrating a Virtual Machine that is in suspended state.
- Suspended state is like paused state in which you resume from same point on later stage.
- Suspended and vmotion migration are considered hot because in both cases the virtual machine is running.
- The primary reason to suspend a virtual machine on a ESXi host is for troubleshooting.

vMotion

- Migrating a Virtual machine that is in “Powered ON” state. This is very useful as this does not cause any downtime for the VM.
- In VMWare vMotion machine is migrated from one ESXi host to another in powered ON state, whereas in storage vMotion machine is migrated from one Datastore to another datastore in powered ON state.
- vMotion moves a running virtual machine to a different ESXi host in the same cluster.
- It is also known as Live Migration.(2msec downtime)



P2V migration

- P2V migration converts a physical computer to a virtual one.
- E.g - You have a webserver running on physical hardware. You can run Vmware vCenter converter, target the webserver and have a copy of the physical server created on an ESXi host.

V2V migration

- V2V migrations are exactly like P2V migrations except that the source machine is already a Virtual Machine.
- Example- Migrating from hyper-V and Vmware workstation to ESXi would be considered a V2V migration.

High Availability



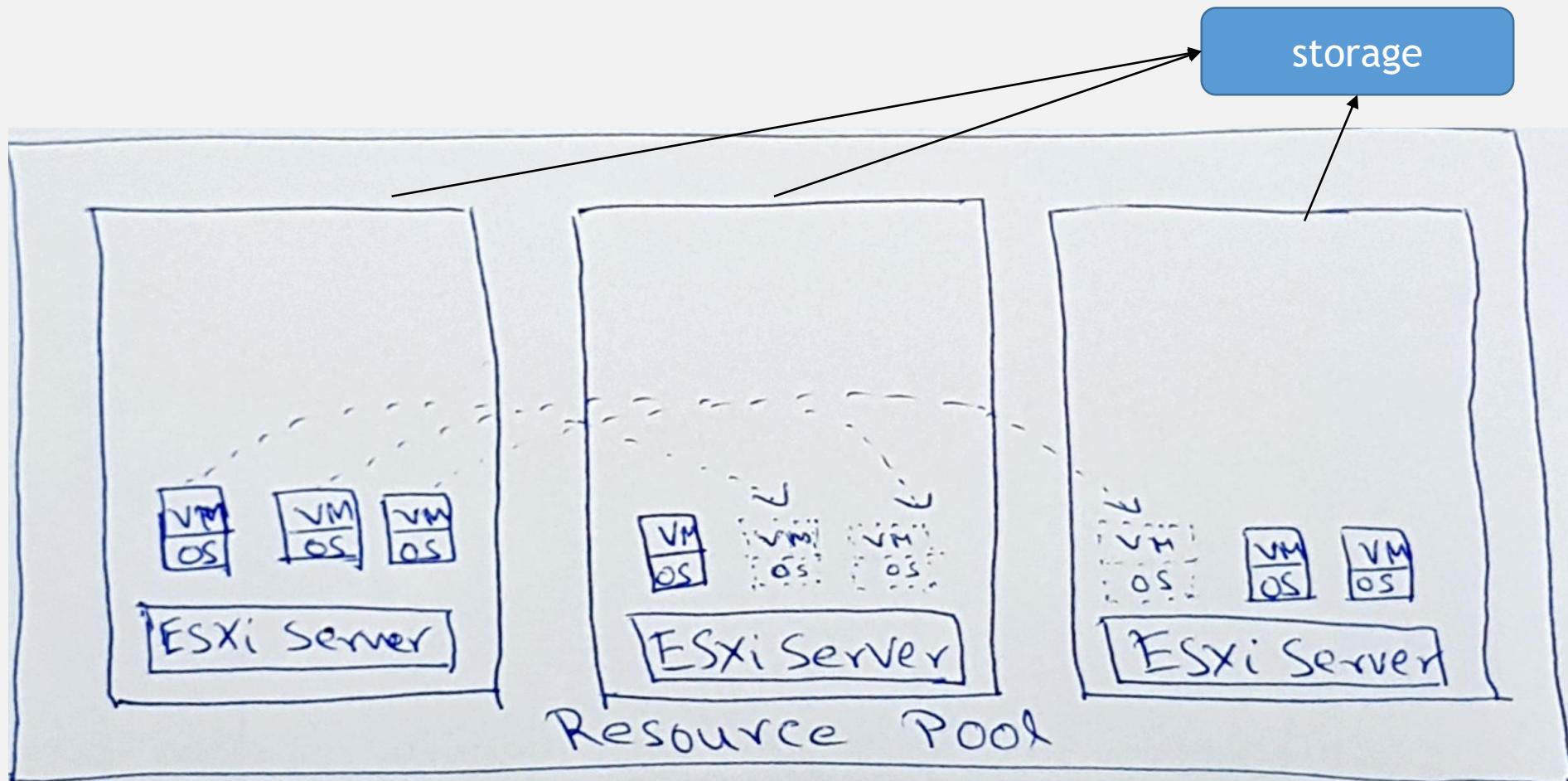
High Availability (HA)

- For High Availability, we need following things
 - Cluster
 - Shared Storage
 - Vcenter server configured for the environment
- Before HA was available, the failure of a single ESXi Host meant that a large no. of virtual machine that were running on it would be down. This was referred as the “all of your eggs in one basket” issue and caused some companies not to deploy virtual servers
- Resource check- Ensure that capacity is always available in order to restart all virtual machine affected by server failure. HA continuously monitors capacity utilization and reserves spare capacity to be able to restart virtual machine.

High Availability

- In HA, when the host(for e.g. ESXi host) crashes or fails, the VM gets restarted on another host. So, there is a very small downtime which is only related to the time taken for VM to restart.
- **Automatic detection of server failure** - HA is a complete automated process and does not need any admin interference as there is no time to recover machine if host is about to crash.
- Neither passive standby ESXi host is require nor any extra VM. The VM for which its parent host is crashing it can restart on any of the other running host.
- HA does not use vmotion
- Enable HA on the cluster setting, in order to use HA

Environment



Pre-requisite for VMWare High Availability

- All host must be licensed for VMWare HA
- You need at least two host in the cluster
- All host need to be configured with static IP address. If you are using DHCP, you must ensure that the address for each host persist across reboots.
- Virtual machine must be located on shared storage(means logical storage) and not on local storage, otherwise they can not be failed-over incase of a host failure.
- All host in a Vmware HA cluster must have DNS configured.

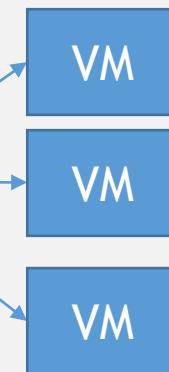
Some note on High Availability

- Note: HA works on the ‘**Master and Slave Architecture**’. When you enable HA on the cluster then election process occurs between all the host in the cluster and one host which has large number of datastore mounted has a chance to become a Master server. Once the election process complete, there will be one master server and other ESXi host are considered as the Slave server if the master server goes down or crashes then the new election process will occur.
- **HA Failover time** - This is measured from the time point vCenter server VM stopped responding to the point vSphere web client started responding to user activity again. With 64 host/6000 VM inventory, the total time is around 460 seconds(approx. 7min) with about 30-40sec for HA to get into action.

Failover time

- **HA Failover time** - This is measured from the time point Vcenter server VM stopped responding to the point vSphere web client started responding to user activity again. With 64 host/6000 VM inventory, the total time is around 460 seconds(approx. 7min) with about 30-40sec for HA to get into action.

- Web client ->Vcenter server->ESXi host



- HA works on ESXi host level, where if any of the ESXi host gets failed, HA will restart those VM onto another ESXi host.

Fault Tolerance



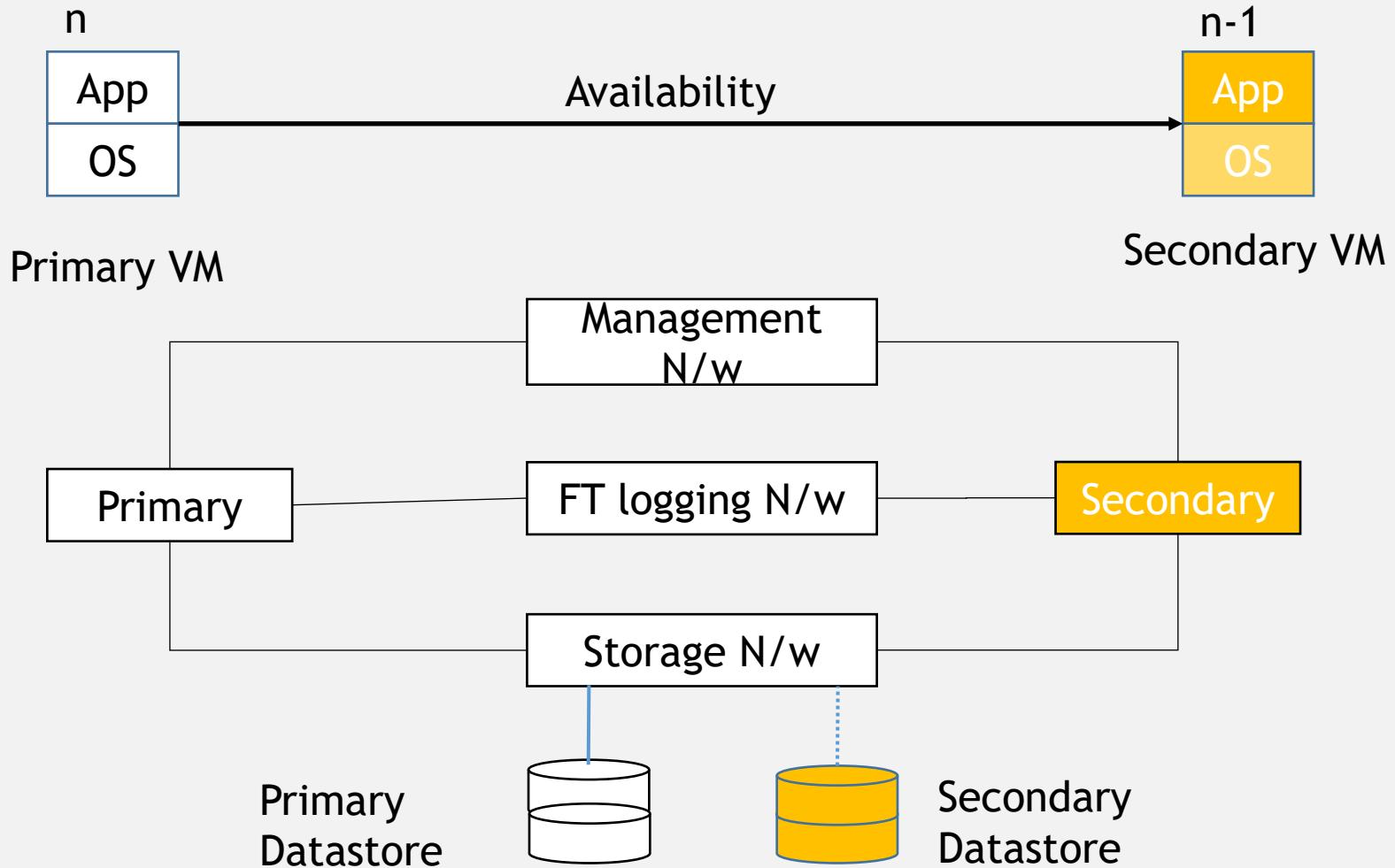
Fault Tolerance

- Aim of Fault Tolerance is similar to HA but in terms of availability it provides 0% downtime(almost) and full availability as machine does not goes down or restarts.
- This is meant for mission critical applications/servers e.g robotic surgery ARM, auto pilot system, spacecraft mission
- VMware lockstep technology is used in fault tolerance(FT)
- With FT a secondary VM is created on another host using distributed resource scheduler. This VM is exact replica of the primary VM.
- A FT virtual machine and its secondary copy are not allowed to run on the same host. This restriction ensures that a host failure cannot result in the loss of both VM.

Fault Tolerance

- Primary and secondary work in lockstep i.e the lockstep technology captures the current state and events of primary VM and sends them to secondary VM. If primary goes down, instantly secondary VM takes over and continue its operation.
- It requires extra standby VM, therefore it is a costlier solution.

Fault Tolerance



Fault Tolerance

- Fault Tolerance avoid ‘Split Brain’ Situation which can leads to two active copies of a VM after recovery from a failure.
- FT works on VM level. Therefore, you can enable or disable FT on VM
- The primary and secondary VM continuously exchange ‘heartbeat’. This exchange allows the VM pair to monitor the status of one another to ensure that FT is continually maintained.

Virtual Machine Template

- You can convert a fully configured virtual machine into a virtual machine template.
- This can be used to rapidly deploy large numbers of new VM that as configured like the original VM.

Thanks!



Cloud and big data technology

Distributed Resource Scheduler

INF-2220

Dilip K. Prasad



Distributed Resource Scheduler

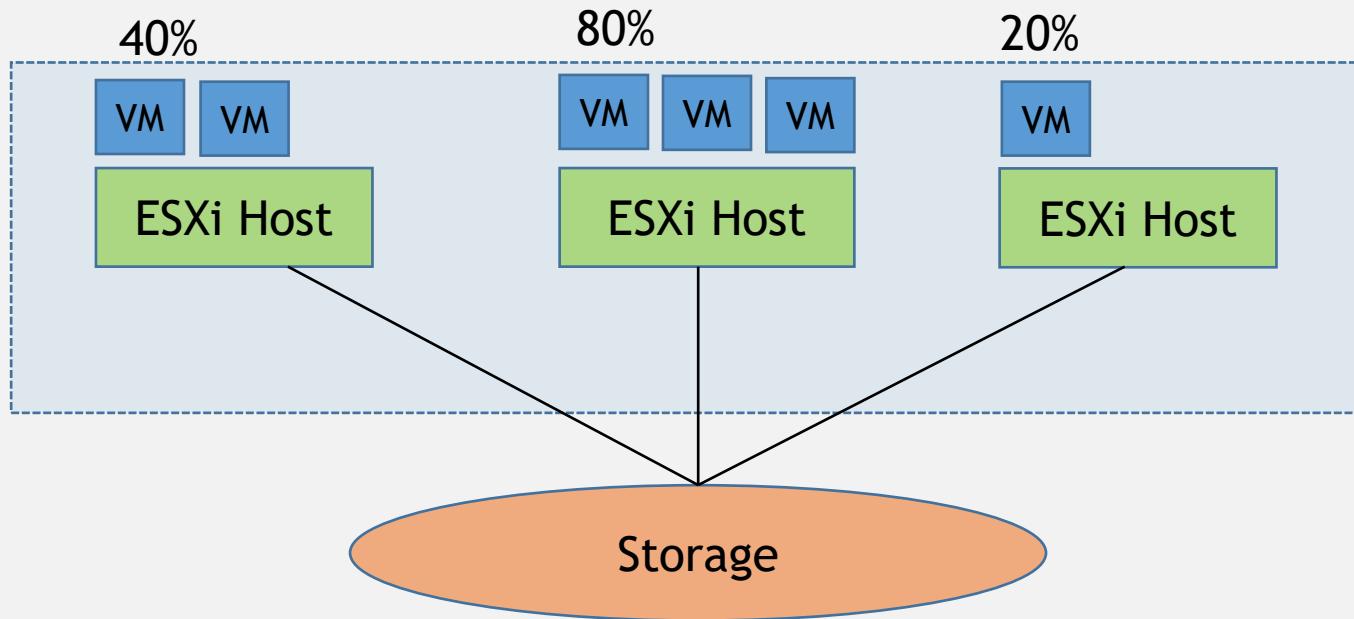
- Manual DRS
 - Partial DRS
 - Fully automated DRS



Distributed Resource Scheduler

- Distributed resource scheduler(DRS) is a feature of cluster which is managed by vcenter server. It balances load of VM across ESXi host.
- A DRS enabled cluster has following resource management capabilities:
 1. Initial VM placement
 2. Load balancing
 3. Power management
- Depending on how end-users are using applications on virtual machines, VMs constantly expands and contracts throughout the day, week or month. The physical hosts becomes over utilized or under utilized based on VM utilization and no of VM running over it.
- Vmotion is a primary requisite of DRS

Distributed Resource Scheduler



Distributed Resource Scheduler

- Main goal of DRS is to -
 - Keep all ESXi servers in the cluster healthy and well utilized by dynamically/automatically moving VMs across the ESXi host
 - Provide VMs with enough resources all the time to keep them running in most efficient ways
- Conduct Zero-downtime server maintenance
- By default, DRS checks in every 5 minutes to see if the cluster workload is balanced or not.

Types of DRS

- **Manual DRS**- When a DRS cluster is set to manual, every time you power on a VM, the cluster prompts you to select the ESXi host where that VM should be hosted.
- **Partially automated DRS** - If you select the partially automated settings in the DRS automation settings. DRS will make an automatic decision about which host a VM should be running when it is initially POWERED ON (without prompting the user who is performing the power-on task) but will still prompt for all migration on the DRS lab. Thus, initial VM placement is automated, but migrations are still manual.
- **Fully automated DRS** - The third setting for DRS is fully automated. If this setting is enabled, then decision for initial placement without prompting and also makes automatic vmotion decision based on the selected automation level.

Thanks!



Cloud and big data technology

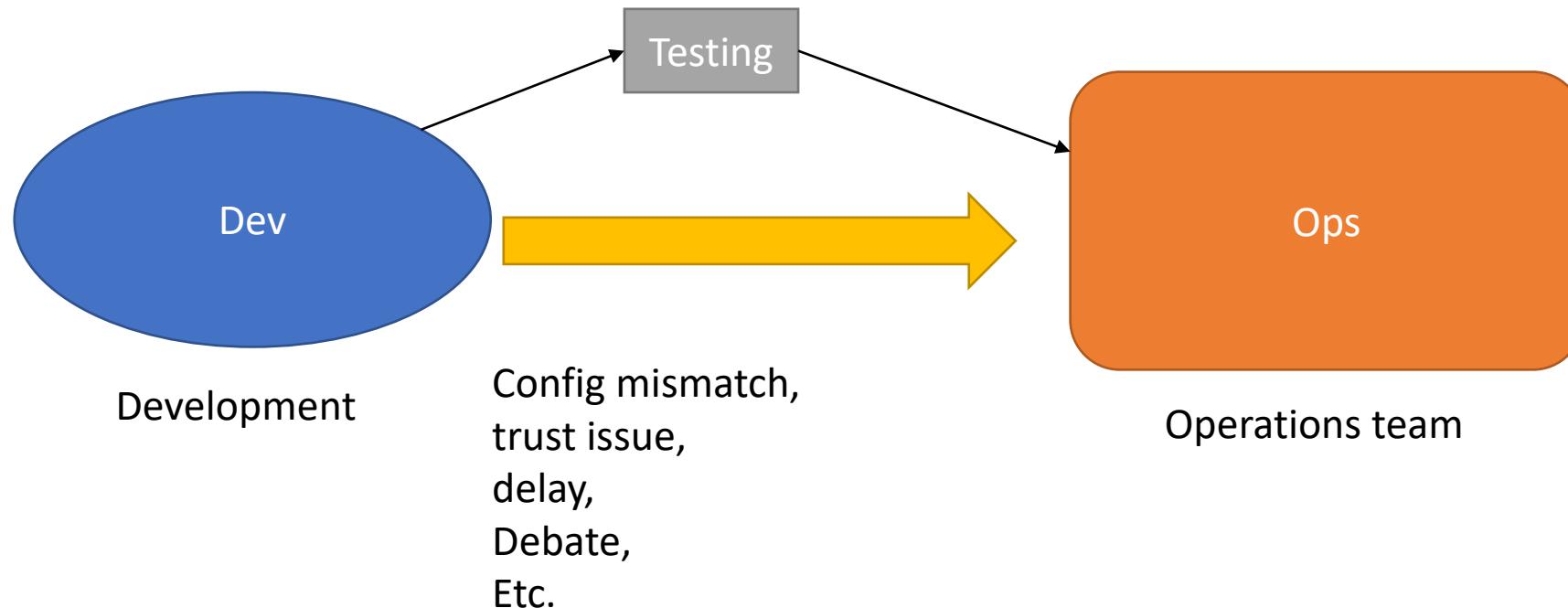
Containerization

INF-2220

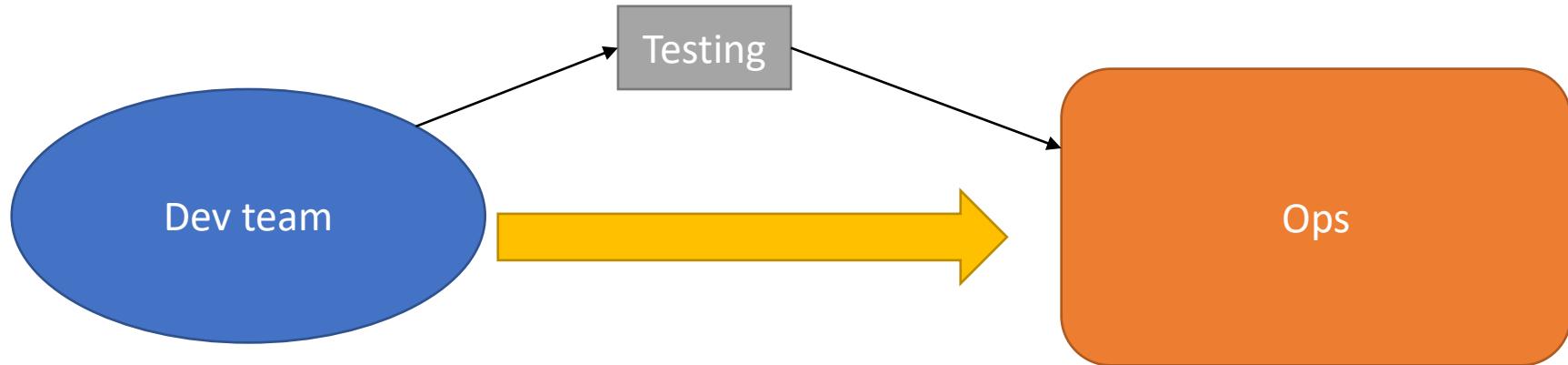
Dilip K. Prasad

What is Docker?

- Container is similar to Virtual Machine
- Docker(Docker Engine) is a tool which will create VM

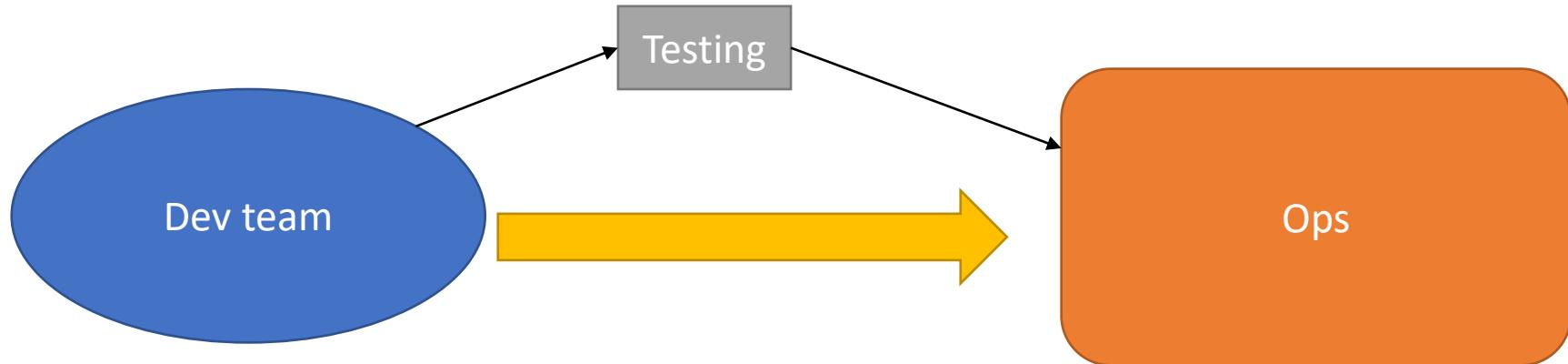


Solution of Dev-Ops mismatch problem



App+
A 2.2
B 3.1
C 1.3
D x 1.1

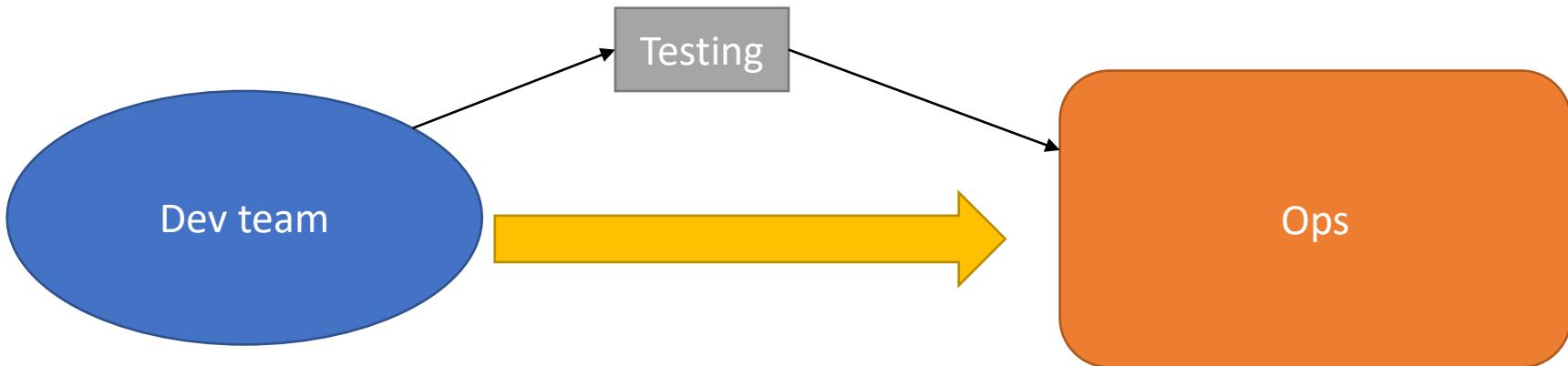
Solution of Dev-Ops mismatch problem



App+
A 2.2
B3.1
C1.3
D x 1.1

OS + s/w

Solution of Dev-Ops mismatch problem

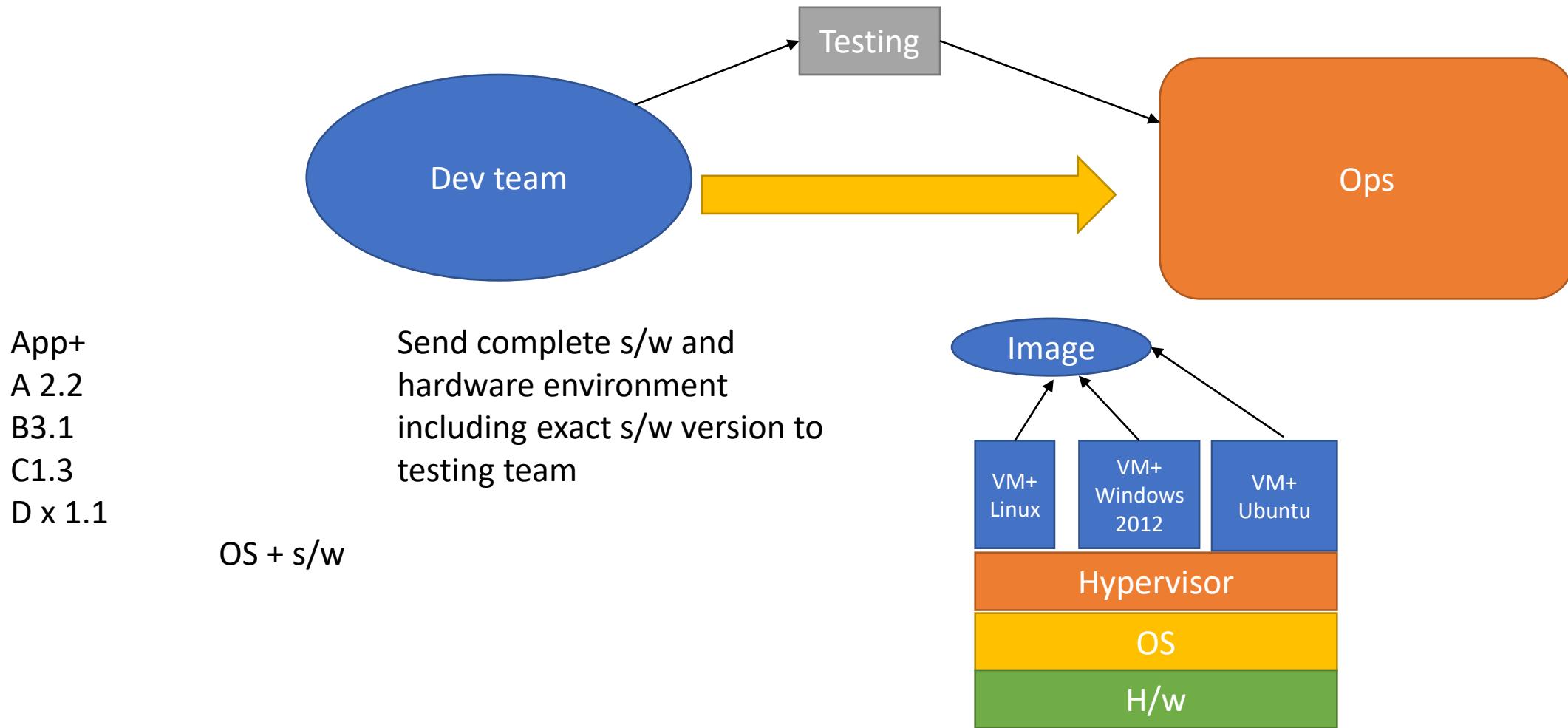


App+
A 2.2
B3.1
C1.3
D x 1.1

Send complete s/w and
hardware environment
including exact s/w version to
testing team

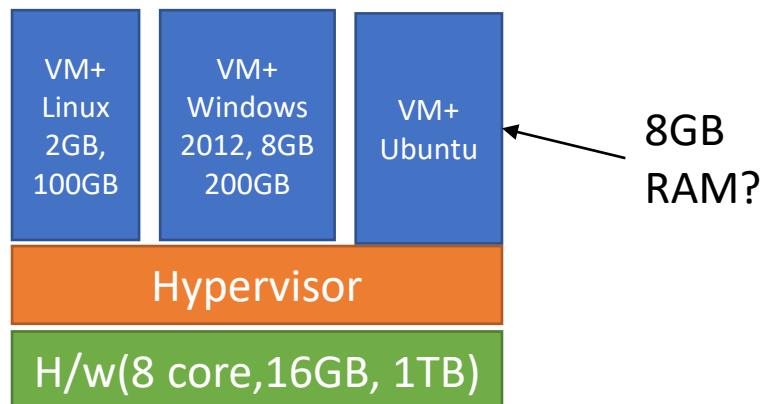
OS + s/w

Solution of Dev-Ops mismatch problem



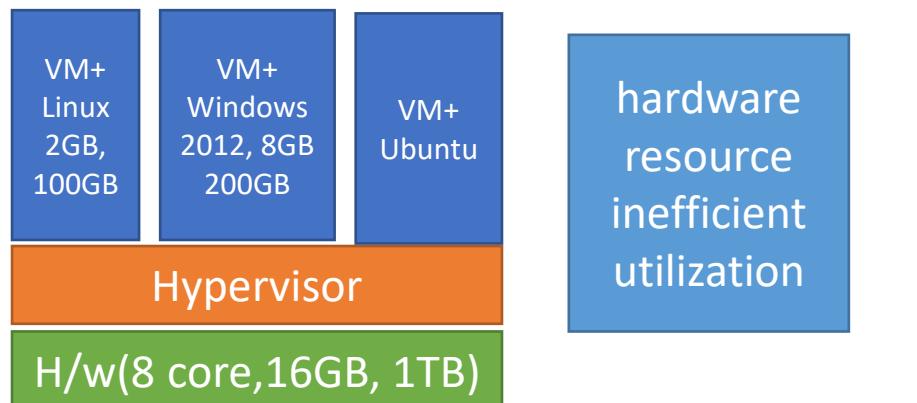
Why are we discussing everything but not docker so far?

- Advance version of virtualization is containerization



Why are we discussing everything but not docker so far?

- Advance version of virtualization is containerization

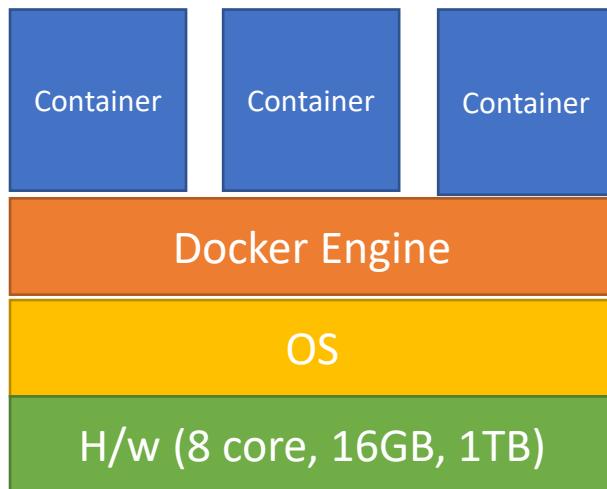


This problem
has been solved
by Docker



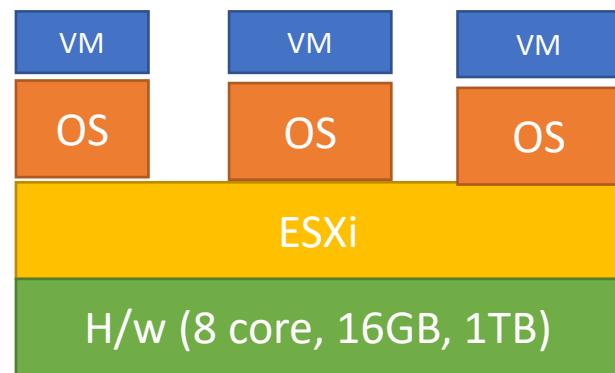
What is Docker?

- Container does not have OS. How its running a system?

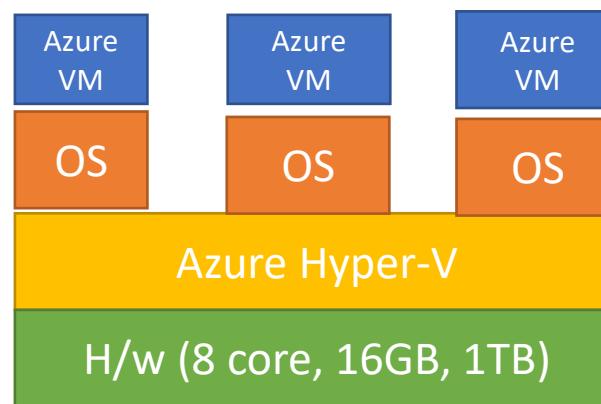


Comparison of Docker vs VMWare & Azure

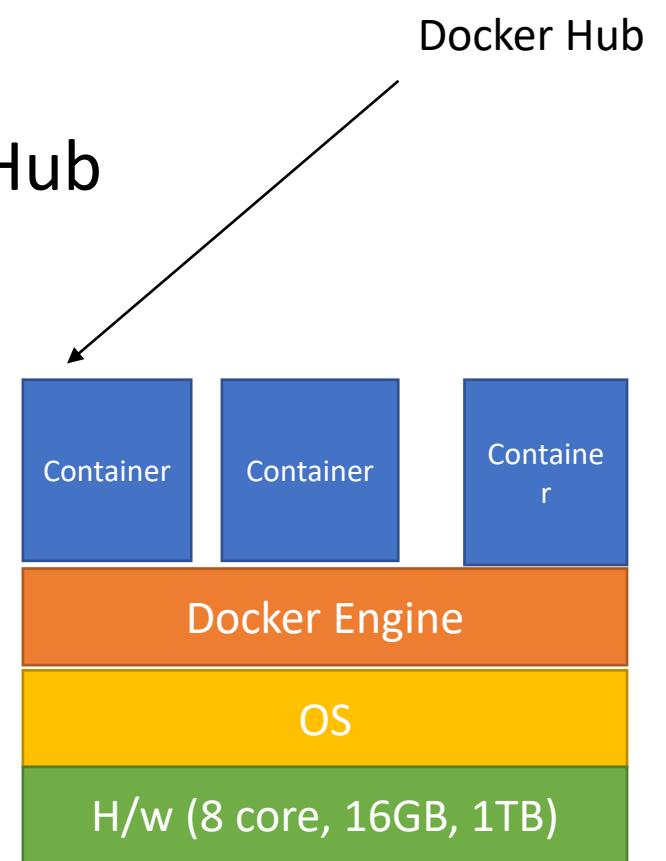
- Docker engine is very light weight compared to other virtualization option
- All s/w and dependency it import from Docker Hub



VMware



Microsoft Azure



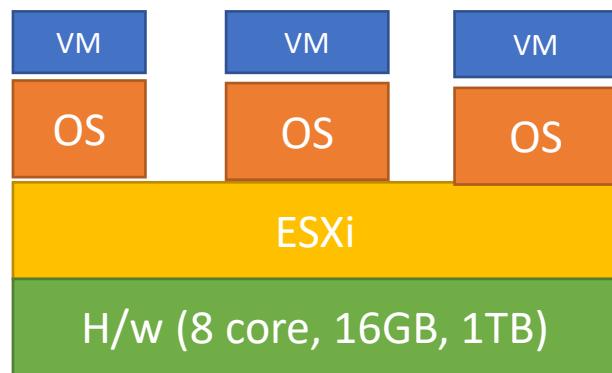
Docker

Docker Hub

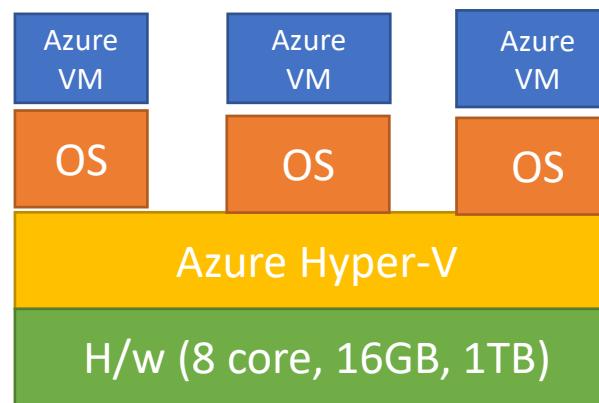
- Docker engine is very light weight compared to other virtualization option
- All s/w and dependency it import from Docker Hub

Command -> Docker run ubuntu 10.1

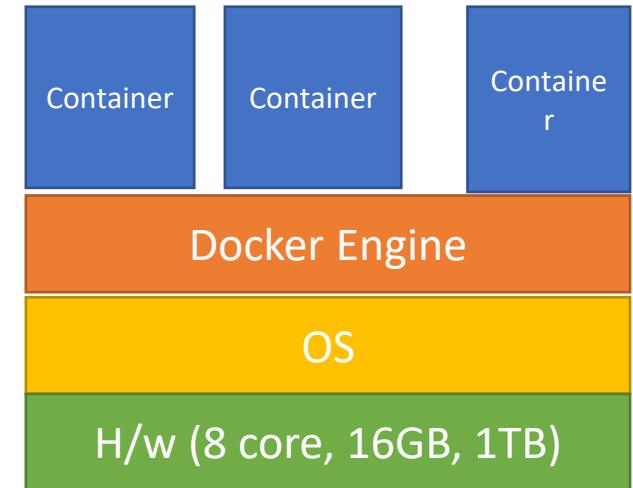
Docker Hub



VMware



Microsoft Azure



Docker

Thanks

Cloud and big data technology

Docker Architecture

INF-2220

Dilip K. Prasad

Outline

Docker

OS vs
Hardware
virtualization

Docker
Architecture

Docker
Ecosystem

Docker

- Docker was first released in 2013 by the developer Solomon Hykes and Sebastian Pahl
- Docker is an open-source centralized platform designed to create deploy and run applications.
- Docker uses Container on the host OS to run applications. It allows applications to use the same Linux kernel as system on the host computer, rather than creating a whole virtual OS.
- We can install Docker on any OS but Docker Engine runs natively on Linux distribution

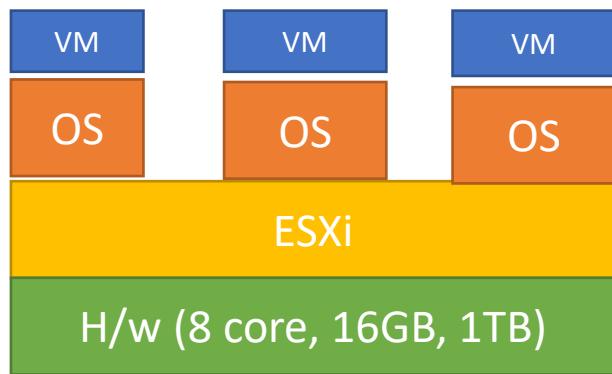
Docker is written in ‘Go’ language

Docker is a tool that performs OS level Virtualization, also known as containerization

Before Docker, many users faces problem that a particular code is running in the developer’s system but not in the user’s system

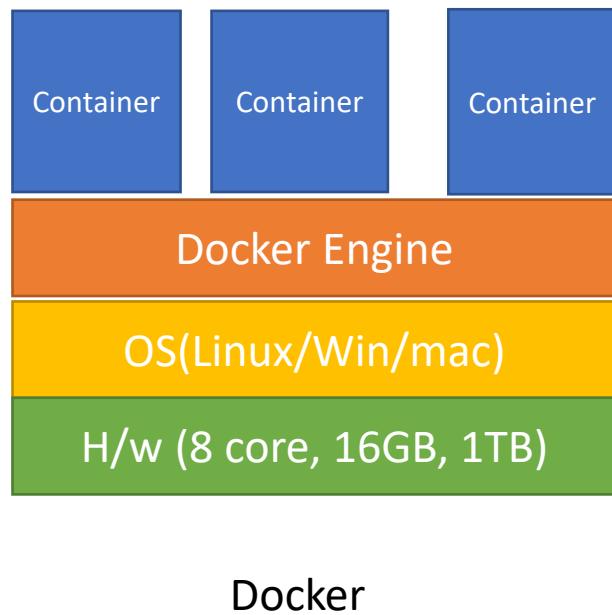
Docker is a set of Platform-as-a-Service that uses OS level virtualization whereas VMware uses Hardware level virtualization

VMWare – Hardware virtualization



VMware

Docker – OS Virtualization



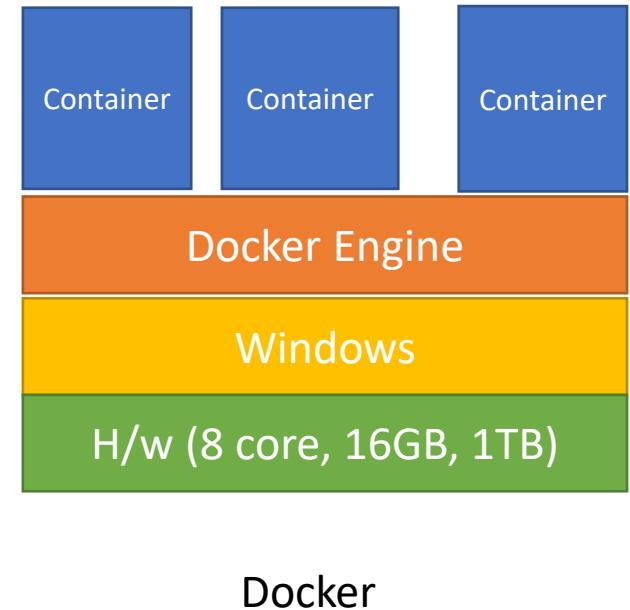
Advantages/Disadvantages of Docker

Advantages of Docker

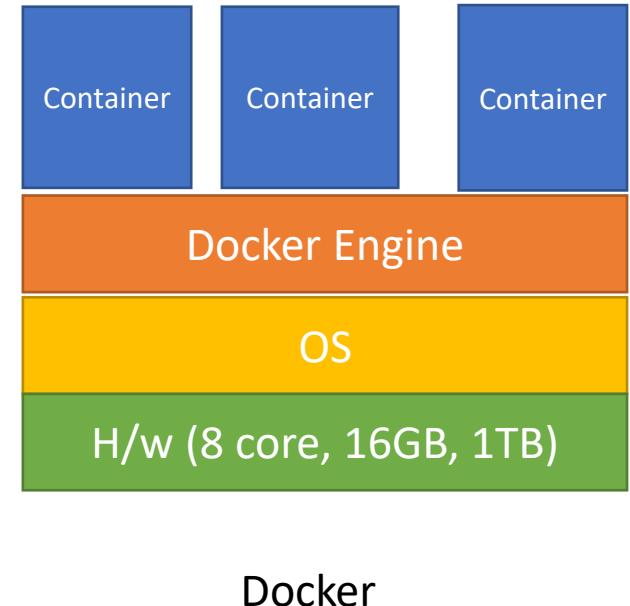
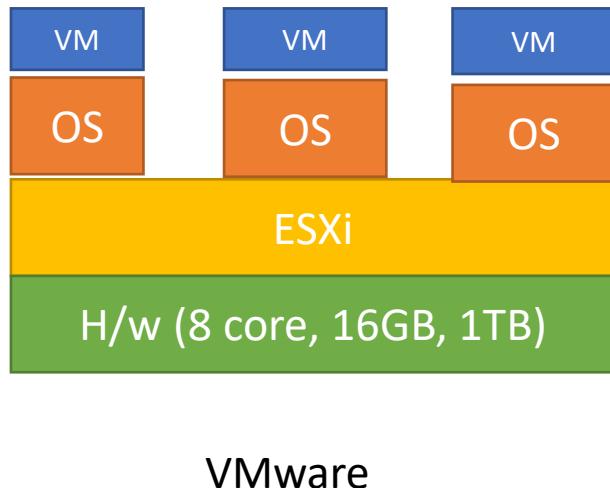
- No pre-allocation of RAM
- CI efficiency – Docker enables you to build a container image and use that same image across every step of the deployment process.
- Less cost
- It is light in weight
- It can run on physical h/w or virtual h/w or on the cloud
- You can re-use the image
- It takes very less time to create container

What about Docker on Windows

- Only windows 10 pro, enterprise, home(aug'21)
- Hyper-v should be enabled



- Image -> when it **running** then its container
- When it is **transferred** then its in the form of image



Advantages of Docker

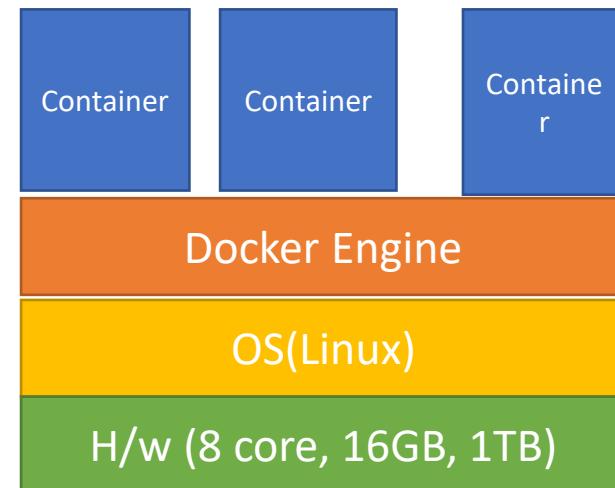
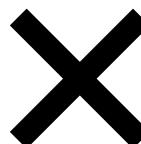
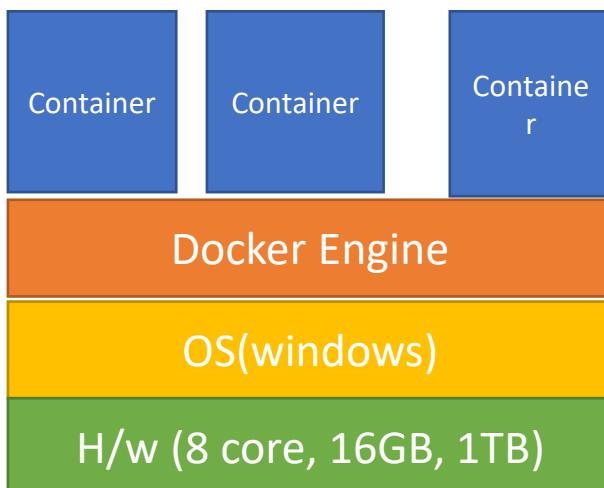
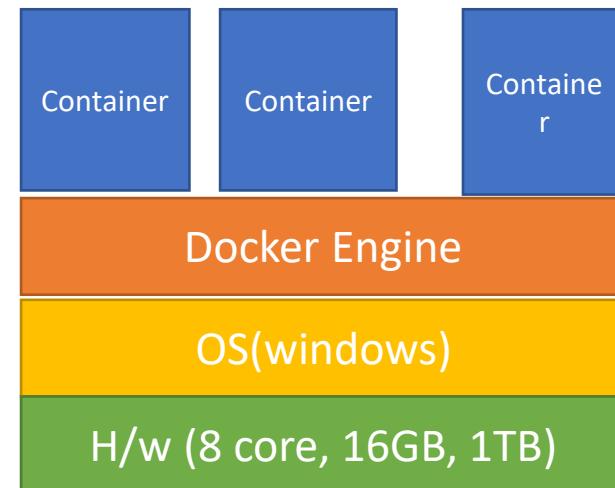
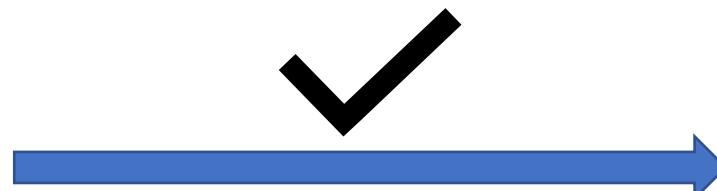
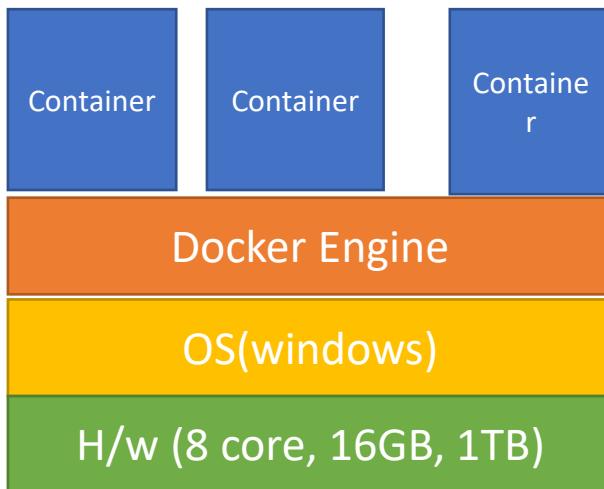
- No pre-allocation of RAM
- Continuous integration efficiency – Docker enables you to build a container image and use that same image across every step of the deployment process.
- Less cost
- It is light in weight
- It can run on physical h/w or virtual h/w or on the cloud
- You can re-use the image
- It takes very less time to create container

Disadvantages of Docker

- Docker is not a good solution for application that requires rich GUI
- Difficult to manage large number of containers
- Docker may cause cross platform compatibility sometime, means if application designed to run in a docker container on windows, then it can't run on Linux or vice-versa
- No solution for data recovery and backup
- Need to evaluate security risks before moving the application to docker

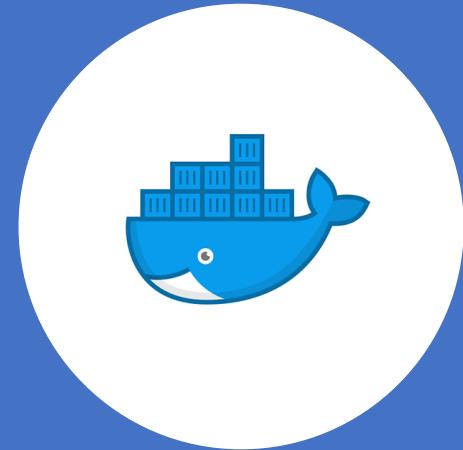
** workaround of above limitations are there but by default above are limitations

Cross compatibility poor sometime



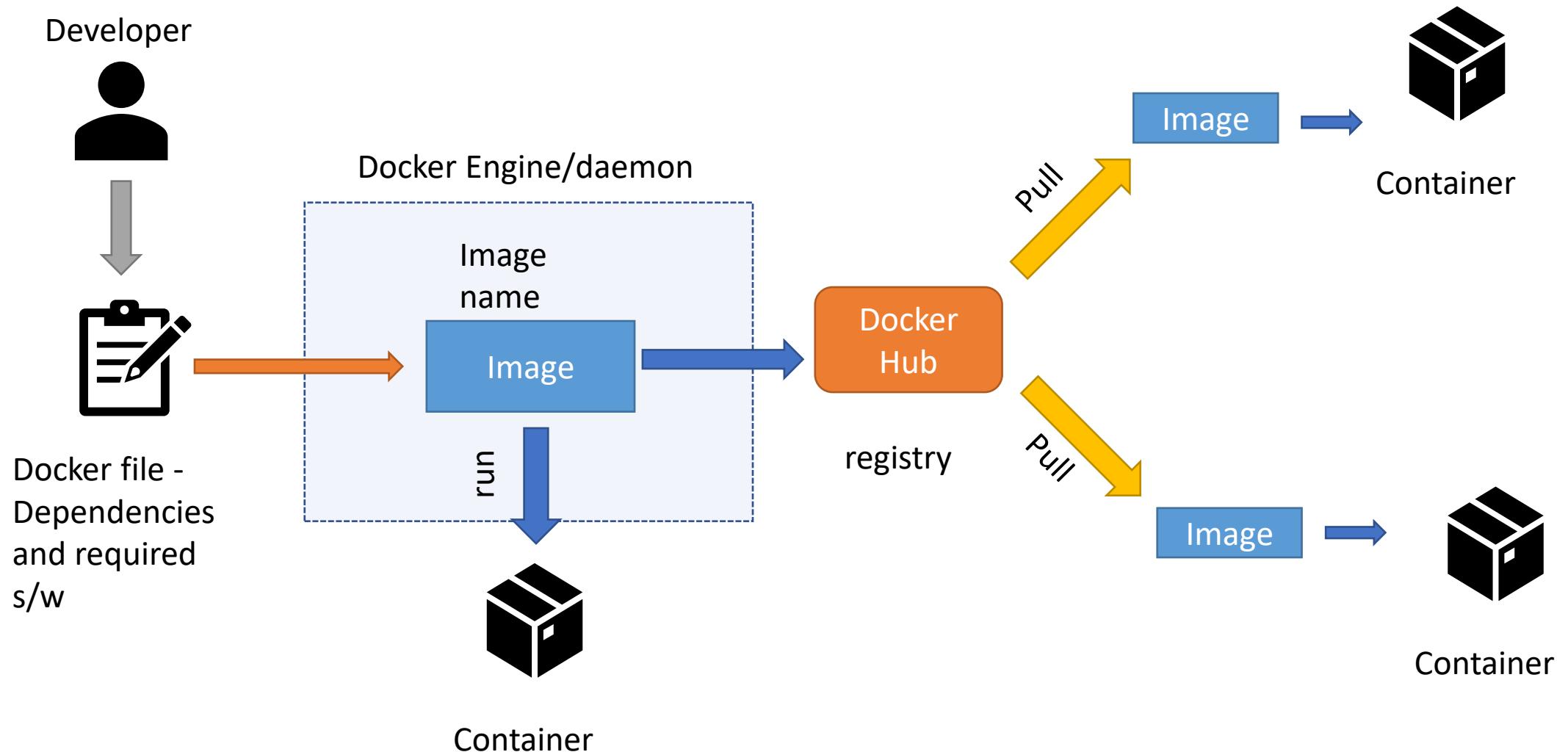
Dev

Ops/Test



Docker Architecture

Docker Architecture



- Layered file system

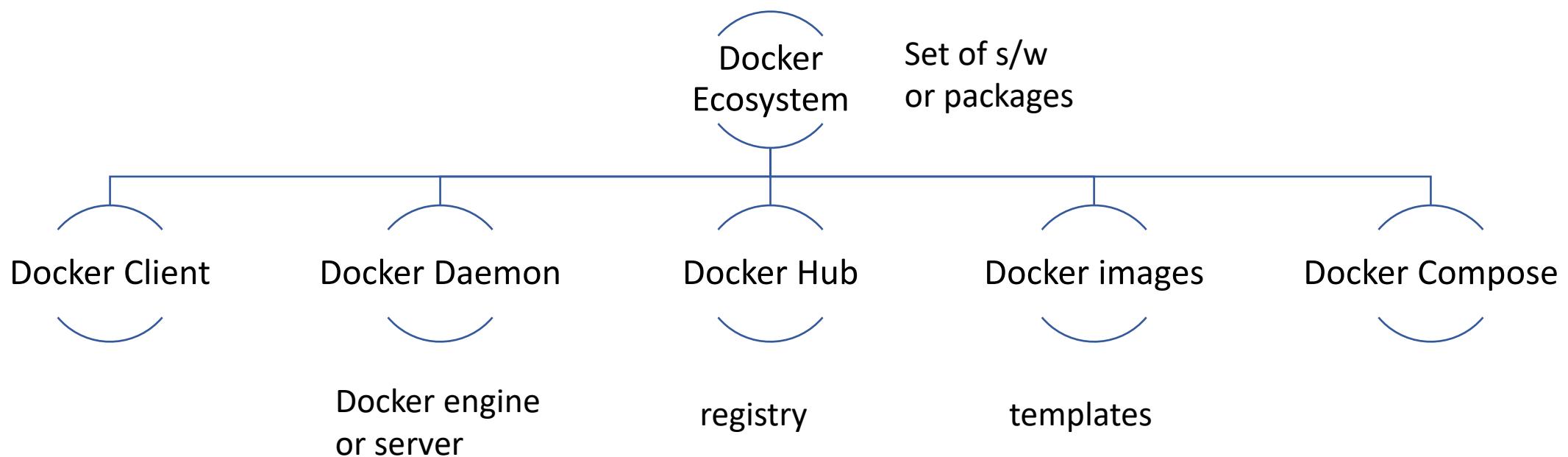


```
$ docker build -t rag004/doggif .
Sending build context to Docker daemon 57.34kB
Step 1/6 : FROM python:3
3: Pulling from library/python
955615a668ce: Pull complete
2756ef5f69a5: Pull complete
911ea9f2bd51: Pull complete
27b0a22ee906: Pull complete
8584d51a9262: Pull complete
524774b7d363: Pull complete
af193b9b3d11: Pull complete
aacb0e56e8f3: Pull complete
46cd7abc9e93: Pull complete
Digest: sha256:e6654afa815122b13242fc9ff513e2d14b00548ba6eaf4d3b03f2f261d85272d
```



Container

Docker Ecosystem



Docker Daemon

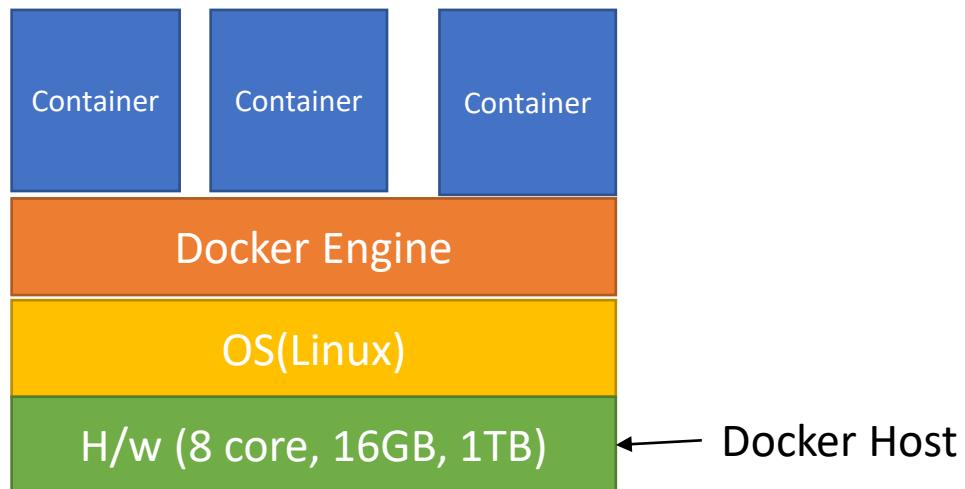
- Docker daemons runs on the host OS
- It is responsible for running Containers to , manages docker services
- Docker Daemon can communicate with other daemons

Docker Client

- Docker users can interact with Docker daemon through a client(CLI)
- Docker client uses commands and REST API to communicate with Docker Daemon
- When a client runs any server command on the Docker client terminal, the Client terminal sends these Docker commands to Docker Daemon
- It is possible for Docker client to communicate with more than one Daemon

Docker Host

- Docker host is used to provide an environment to execute and run applications.
- It contains the docker daemon, images, containers, networks and storages.



Docker Hub/Registry

- Docker registry manages and stores the docker images
- There are two types of registries in the Docker
 - 1. **Public Registry** – Public Registry is also called as Docker Hub
 - 2. **Private Registry** – It is used to share images within the enterprise

Docker Images

- Docker images are the read only binary templates used to create Docker Containers
- Or
- Single file with all dependencies and configuration required to run a program

Ways to create an Image or pull an image

1. Take image from Docker Hub
2. Create image from Docker file
3. Create image from existing Docker Containers

Docker Container

- Container hold the entire packages that is needed to run the application
- Or
- In other words, we can say that the image is a template, and the container is a copy of that template
- Container is like a Virtual machine
- Images becomes Container when they run on Docker Engine

Docker commands

- Docker images
 - To see all images present in your local m/c
- Docker search ubuntu
 - To find images in docker hub
- Docker pull centos
 - To download image from dockerhub to local m/c
- Docker run –it –name dilip ubuntu /bin/bash
 - To give name to container
 - To check service is start or not
- Service docker status

- Docker start dilip
 - To start container name dilip
- Docker attach dilip
 - To go inside container
- Docker ps –a
 - To see all container
- Docker ps
 - To see only running container
- Docker stop dilip
 - To stop container
- Docker rm dilip
 - To delete container



DevOps
topics to
explore

SDLC in a nutshell

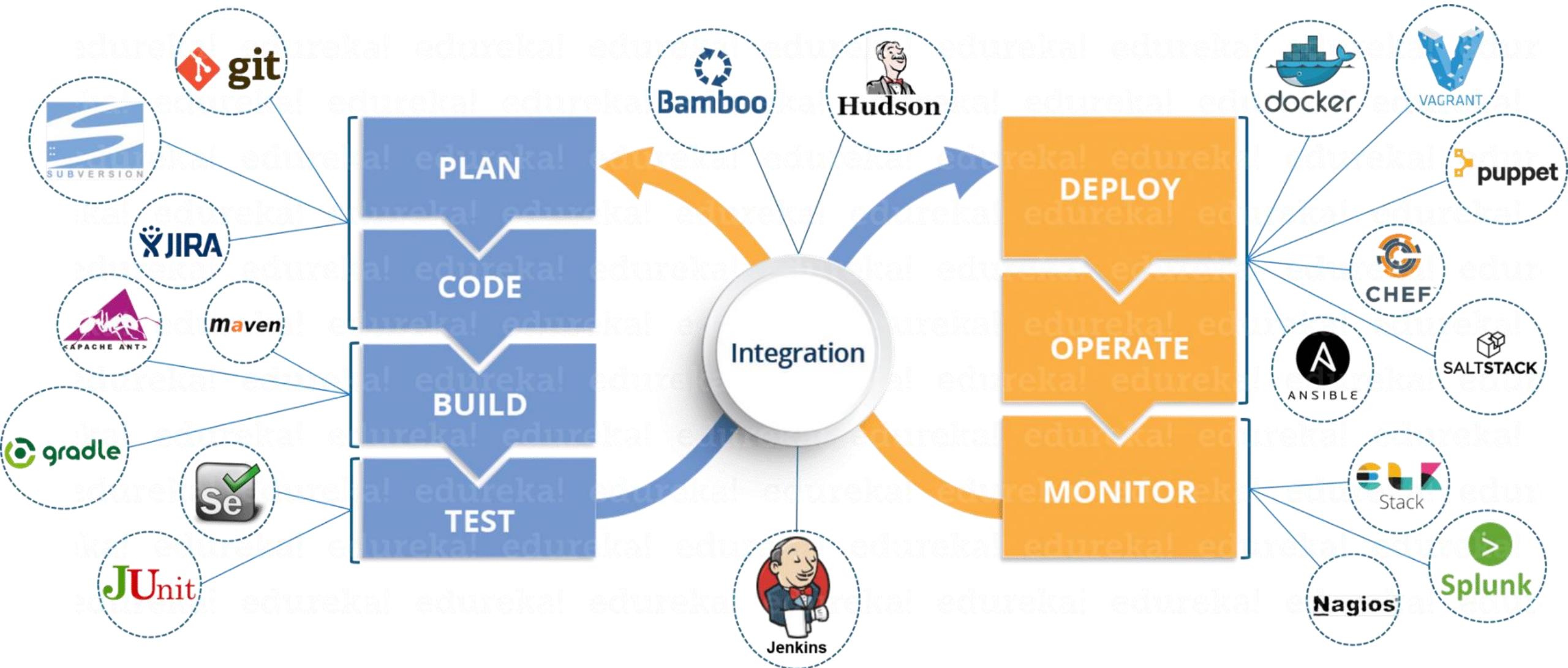
Development

- Developer
- Build
- Test
- Quality Assurance

Operations

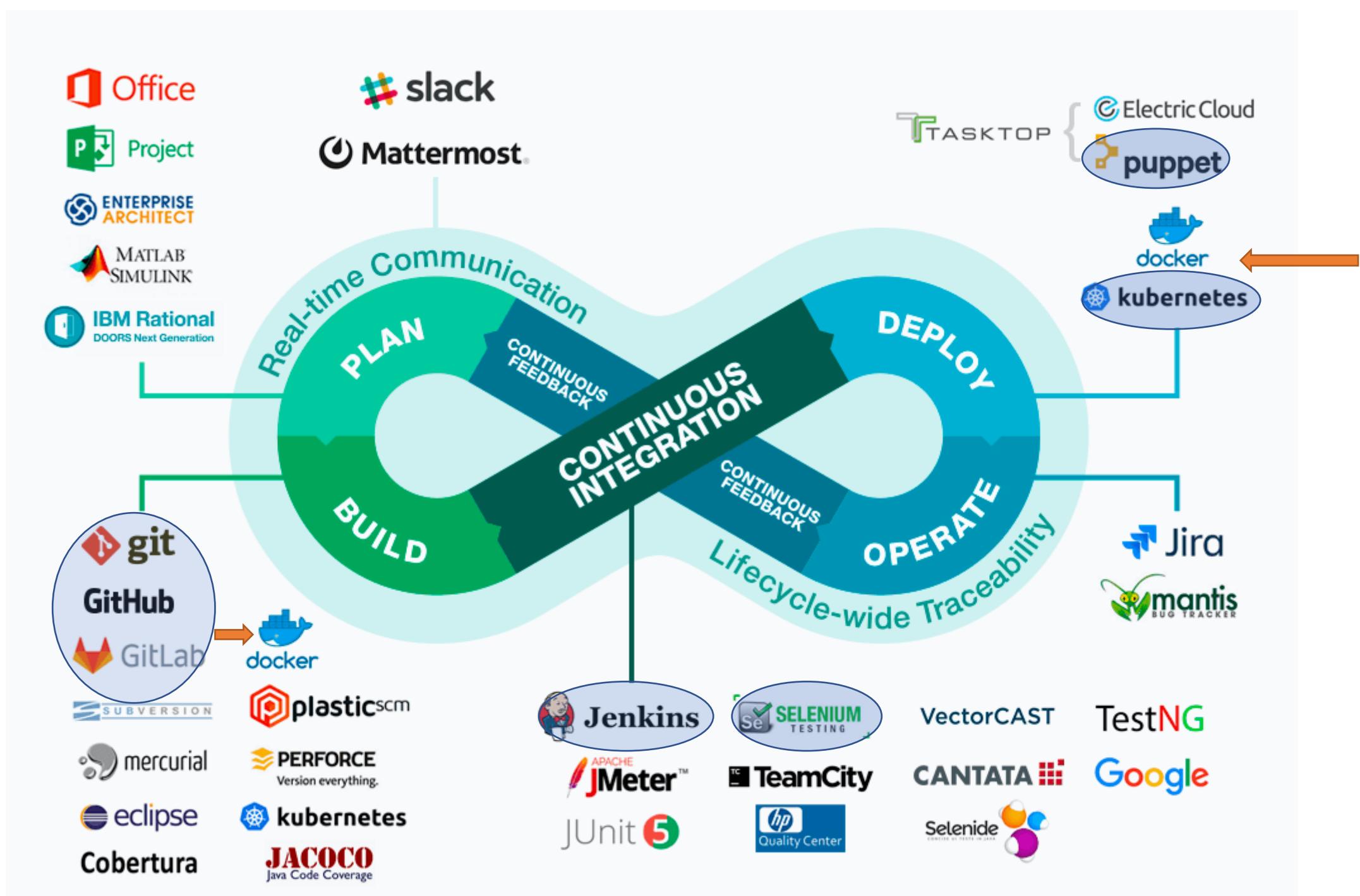
- Deploy
- Maintenance
- Monitoring

Dev-ops lifecycle

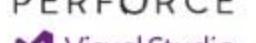
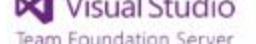
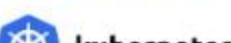
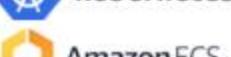
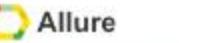


Dev-Ops what to explore further if interested

- Git – plan and code
- Jenkins – CI- continuous integration
- Chef – configuration management
- Ansible - Deployment
- Maven – build automation tool
- Selenium - testing automation
- Nagios – continuous monitoring



The DevOps Tool Ecosystem

Scripts	CI/CD tools	Build tools	Source Code Management Tools	
 Perl  Groovy  BASH	 Jenkins  TeamCity  circleci	 Bamboo  Azure DevOps  GitLab	 BuildMaster  Maven  Gradle	 Bitbucket  PERFORCE  Visual Studio Team Foundation Server  Subversion  GitHub
Deploy Tools	Virtualization & Containerization	Reports, Statistics & Analytics	Test Automation	Static Code Analysis Tools
 puppet  CakePHP  CHEF  ANSIBLE	 docker  kubernetes  Amazon ECS  MESOS  Microsoft Hyper-v	 Tableau  Allure ReportPortal.io solarwinds  Nagios  dynatrace  zenoss  New Relic	 JUnit  Tosca  Selenium  Katalon  cypress.io  SoapUI  UFT One	 sonarqube  CODACY  ReSharper  coverity 

Cloud and big data technology

Big Data Algorithms
INF 2220
Dilip K. Prasad

Big Data Algorithms & Analytics

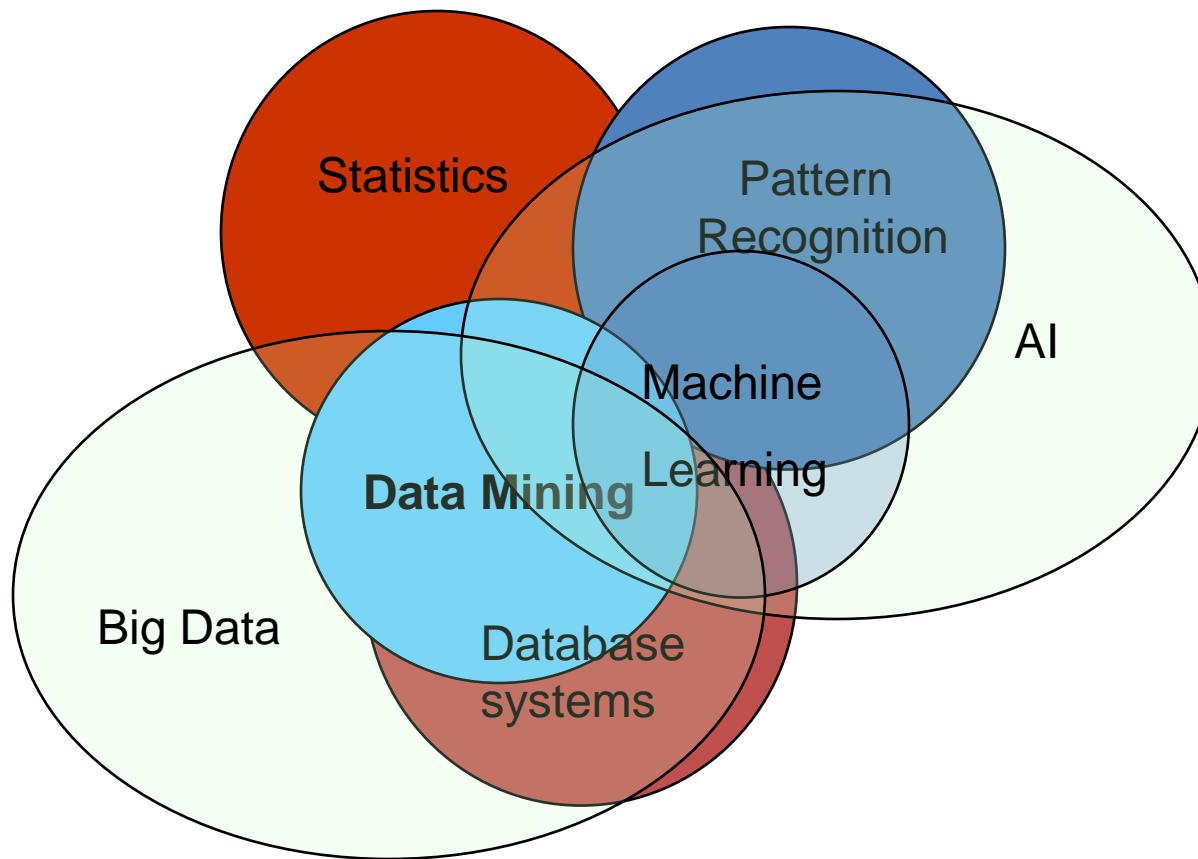
- Exploratory Data Analysis
(preprocessing/understanding the data)
- Linear Classification
- Linear Regression
- Decision Tree
- Apriori
- Information retrieval
- K-means Clustering
- EM Algorithm
- PageRank & HITS
- Collaborative Filtering

Big Data Algorithm types

- **Supervised learning**
 - Classification
 - Regression/Prediction
- **Unsupervised learning**
 - Clustering
- **Semi-supervised learning**
- **Association Analysis**
- **Reinforcement learning**

Machine Learning & Data analysis

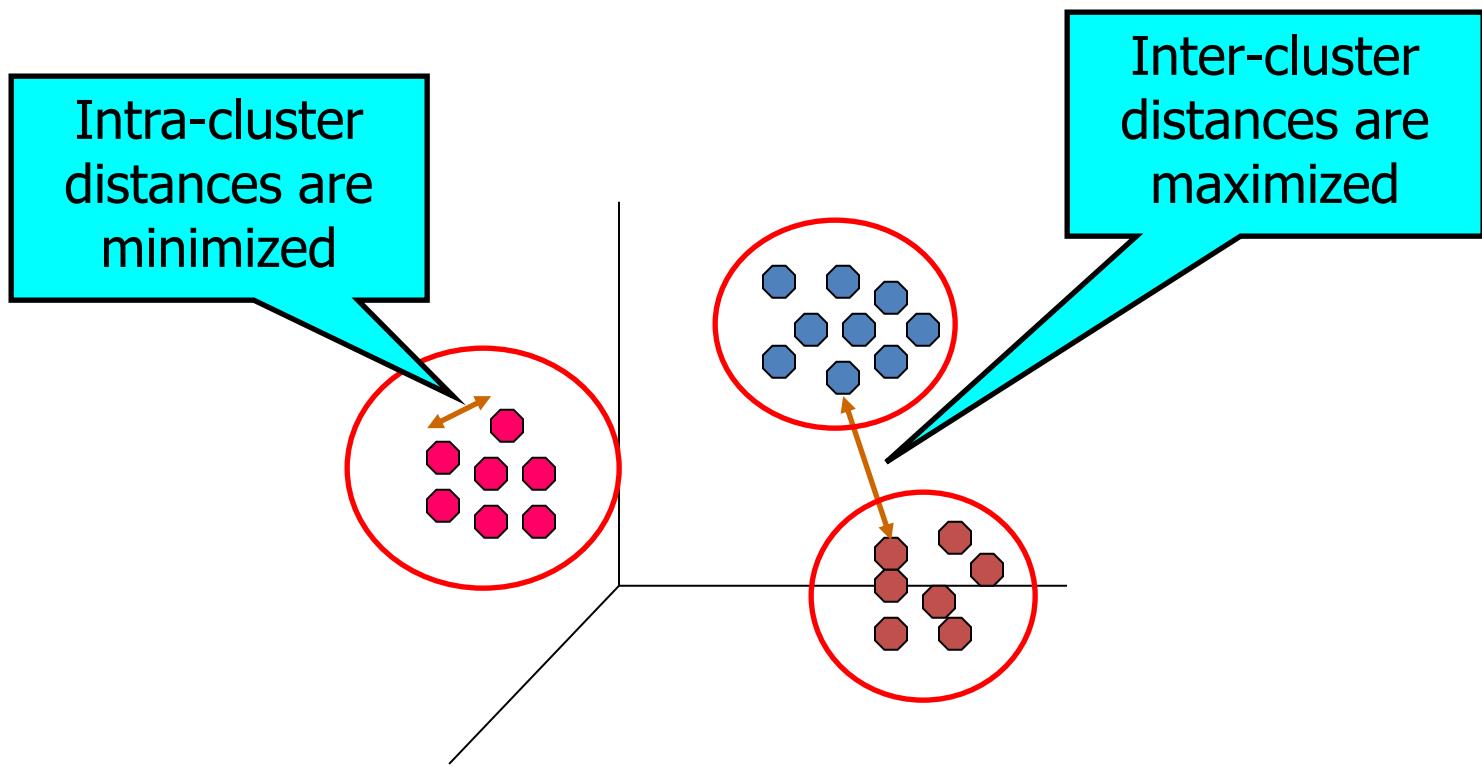
- **Machine learning** focuses on prediction, based on known properties learned from the training data.
- **Big Data analysis** focuses on the discovery of (previously) unknown properties in the data. This is the analysis step of Knowledge Discovery in Databases.
- **Big Data analysis** uses many machine learning methods, but often with a slightly different goal in mind
- **Machine learning** also employs big data analysis/data mining methods as "unsupervised learning" or as a preprocessing step to improve learner accuracy.



Unsupervised Learning: Cluster Analysis

What is Cluster Analysis?

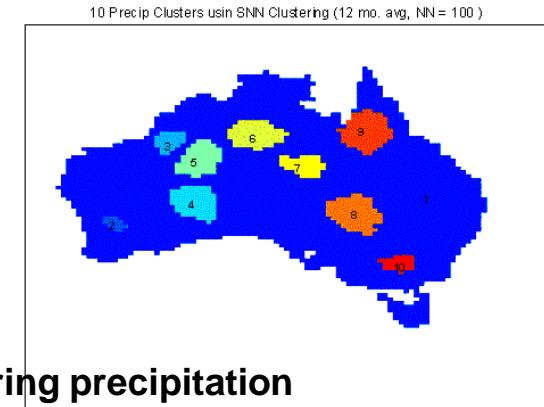
- Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



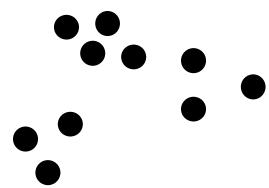
Applications of Cluster Analysis

- **Understanding**
 - Group related documents for browsing, group genes and proteins that have similar functionality, or group stocks with similar price fluctuations
- **Summarization**
 - Reduce the size of large data sets

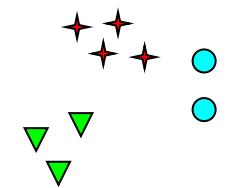
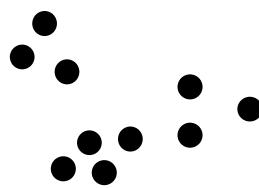
	<i>Discovered Clusters</i>	<i>Industry Group</i>
1	Applied-Matl-DOWN,Bay-Network-Down,3-COM-DOWN, Cabletron-Sys-DOWN,CISCO-DOWN,HP-DOWN, DSC-Comm-DOWN,INTEL-DOWN,LSI-Logic-DOWN, Micron-Tech-DOWN,Texas-Inst-Down,Tellabs-Inc-Down, Natl-Semiconduct-DOWN,Oracl-DOWN,SGI-DOWN, Sun-DOWN	Technology1-DOWN
2	Apple-Comp-DOWN,Autodesk-DOWN,DEC-DOWN, ADV-Micro-Device-DOWN,Andrew-Corp-DOWN, Computer-Assoc-DOWN,Circuit-City-DOWN, Compaq-DOWN, EMC-Corp-DOWN, Gen-Inst-DOWN, Motorola-DOWN,Microsoft-DOWN,Scientific-Atl-DOWN	Technology2-DOWN
3	Fannie-Mae-DOWN,Fed-Home-Loan-DOWN, MBNA-Corp-DOWN,Morgan-Stanley-DOWN	Financial-DOWN
4	Baker-Hughes-UP,Dresser-Inds-UP,Halliburton-HLD-UP, Louisiana-Land-UP,Phillips-Petro-UP,Unocal-UP, Schlumberger-UP	Oil-UP



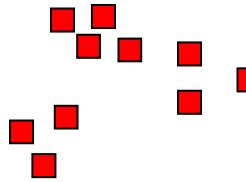
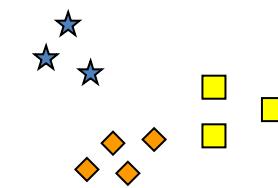
Notion of a Cluster can be Ambiguous



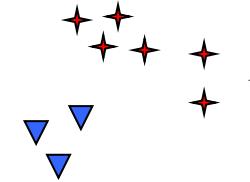
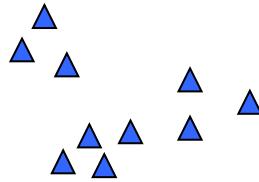
How many clusters?



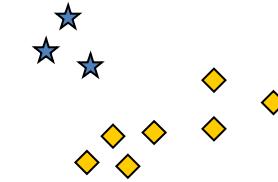
Six Clusters



Two Clusters



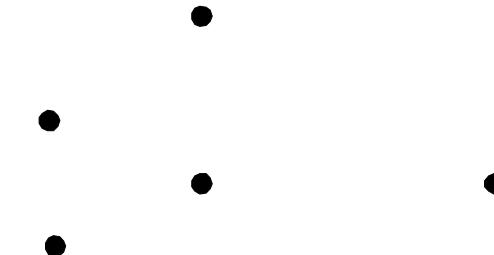
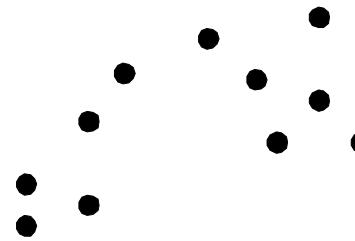
Four Clusters



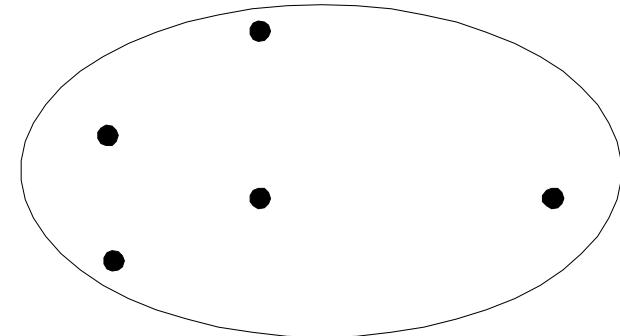
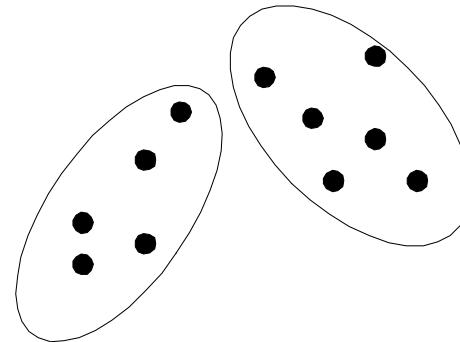
Types of Clusterings

- A **clustering** is a set of clusters
- Important distinction between **hierarchical** and **partitional** sets of clusters
- **Partitional Clustering**
 - A division data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset
- **Hierarchical clustering**
 - A set of nested clusters organized as a hierarchical tree

Partitional Clustering

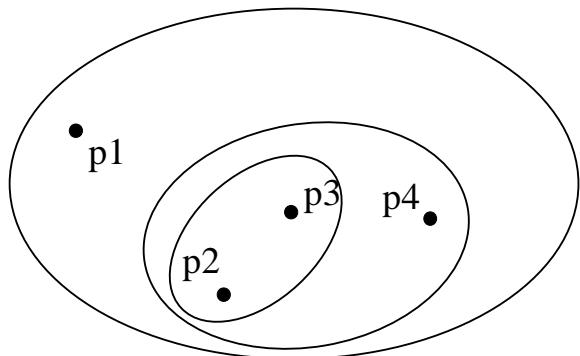


Original Points

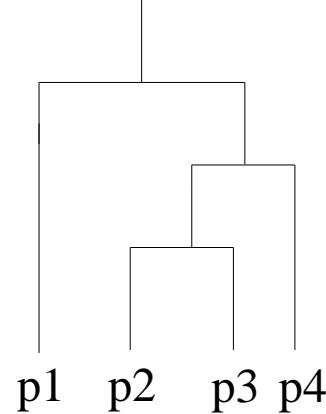


A Partitional Clustering

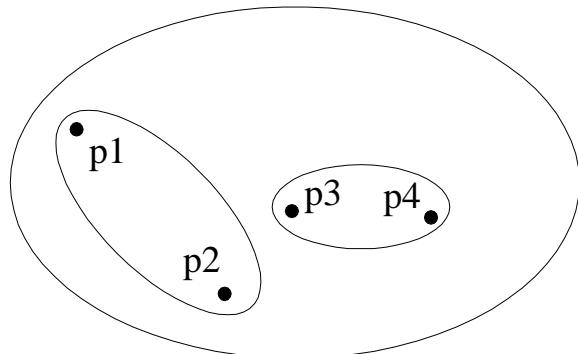
Hierarchical Clustering



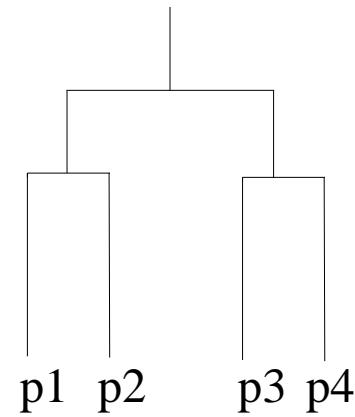
Traditional Hierarchical Clustering



Traditional Dendrogram



Non-traditional Hierarchical Clustering



Non-traditional Dendrogram

Clustering Algorithms

- K-means and its variants
- Hierarchical clustering
- Density-based clustering (DBSCAN)

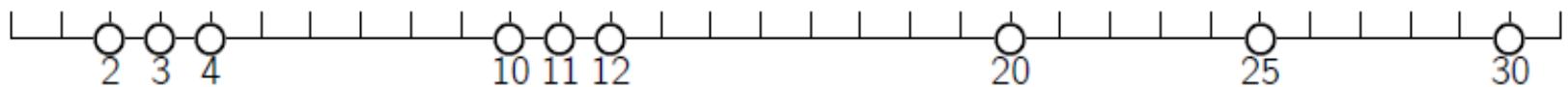
K-means Clustering

- Partitional clustering approach
 - Each cluster is associated with a **centroid** (center point)
 - Each point is assigned to the cluster with the closest centroid
- Number of clusters, K , must be specified
- The basic algorithm is very simple

-
- 1: Select K points as the initial centroids.
 - 2: **repeat**
 - 3: Form K clusters by assigning all points to the closest centroid.
 - 4: Recompute the centroid of each cluster.
 - 5: **until** The centroids don't change
-

K-means Clustering – Details

- Initial centroids are often chosen randomly.
 - Clusters produced vary from one run to another.
- The centroid is (typically) the mean of the points in the cluster.
- ‘Closeness’ is measured by Euclidean distance, cosine similarity, correlation, etc.



(a) Initial dataset

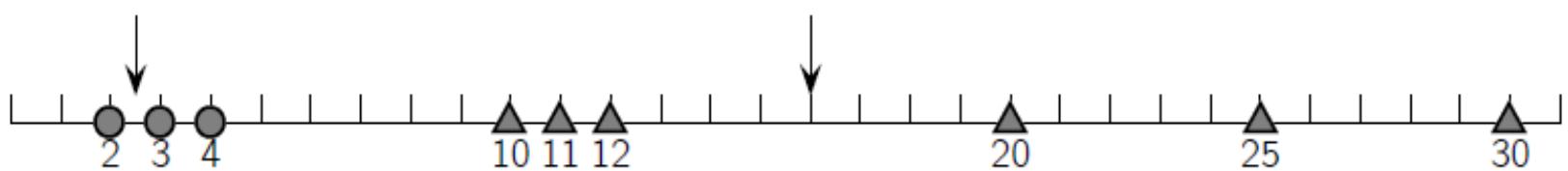
$$\mu_1 = 2 \quad \mu_2 = 4$$



(b) Iteration: $t = 1$

$$\mu_1 = 2.5$$

$$\mu_2 = 16$$



(c) Iteration: $t = 2$

$$\mu_1 = 3$$

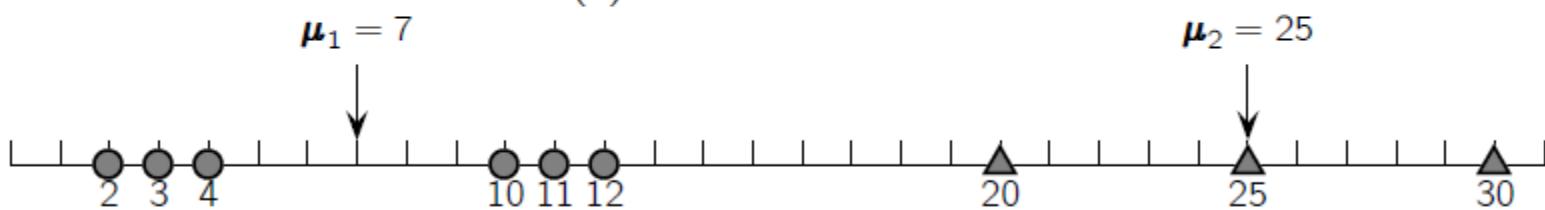
$$\mu_2 = 18$$



(d) Iteration: $t = 3$



(e) Iteration: $t = 4$



(f) Iteration: $t = 5$ (converged)

K-means Clustering – Details

- K-means will converge for common similarity measures mentioned above.
- Most of the convergence happens in the first few iterations.
 - Often the stopping condition is changed to ‘Until relatively few points change clusters’
- Complexity is $O(n * K * I * d)$
 - n = number of points, K = number of clusters,
 I = number of iterations, d = number of attributes

Evaluating K-means Clusters

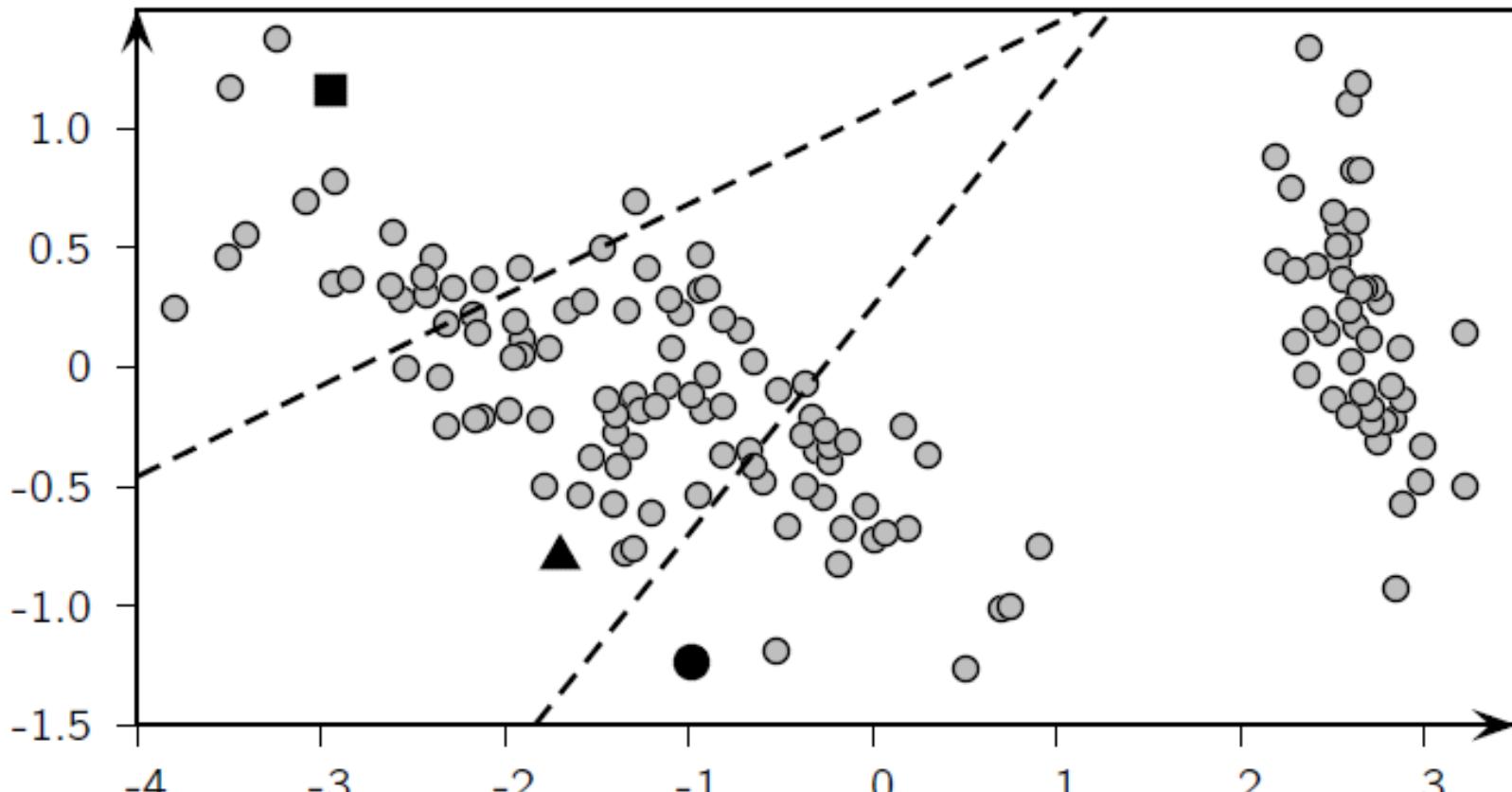
- Most common measure is Sum of Squared Error (SSE)
 - For each point, the error is the distance to the nearest cluster
 - To get SSE, we square these errors and sum them.
- x is a data point in cluster C_i and m_i is the representative point for cluster C_i
 - can show that m_i corresponds to the center (mean) of the cluster
- Given two clusters, we can choose the one with the smallest error
- One easy way to reduce SSE is to increase K , the number of clusters
 - A good clustering with smaller K can have a lower SSE than a poor clustering with higher K

Algorithm 16.1: K-means Algorithm

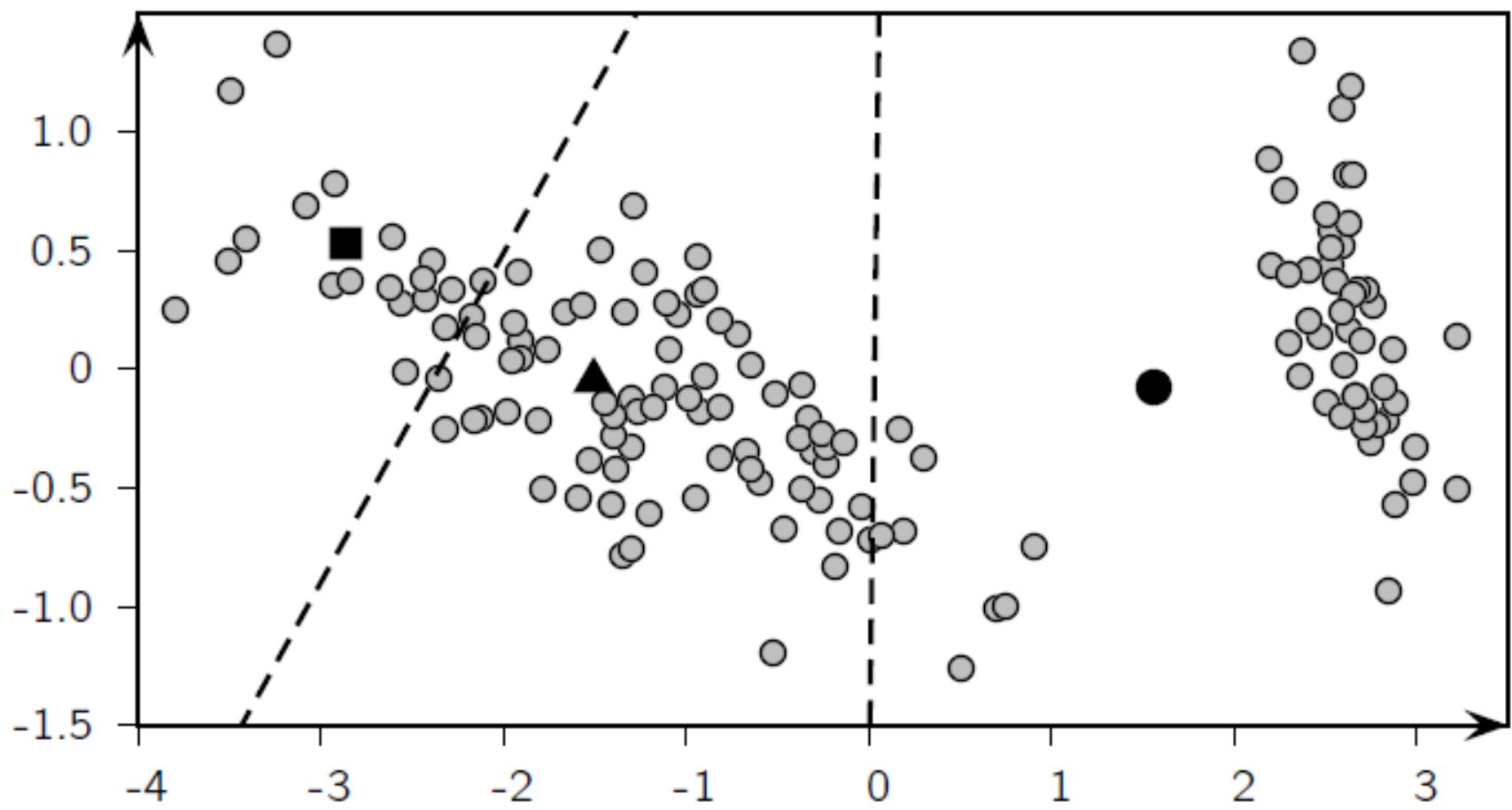
K-means (D, k, ϵ):

- 1 $t = 0$
- 2 Randomly initialize k centroids: $\mu_1^t, \mu_2^t, \dots, \mu_k^t$
- 3 **repeat**
- 4 $t = t + 1$
 // Cluster Assignment Step
- 5 **foreach** $x_j \in D$ **do**- 6 $j^* = \arg \min_i \{ \|x_j - \mu_i^t\|^2 \}$ // Assign x_j to closest centroid
- 7 $C_{j^*} = C_{j^*} \cup \{x_j\}$
- 8 // Centroid Update Step
- 9 **foreach** $i = 1$ to k **do**- 10 $\mu_i^t = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j$

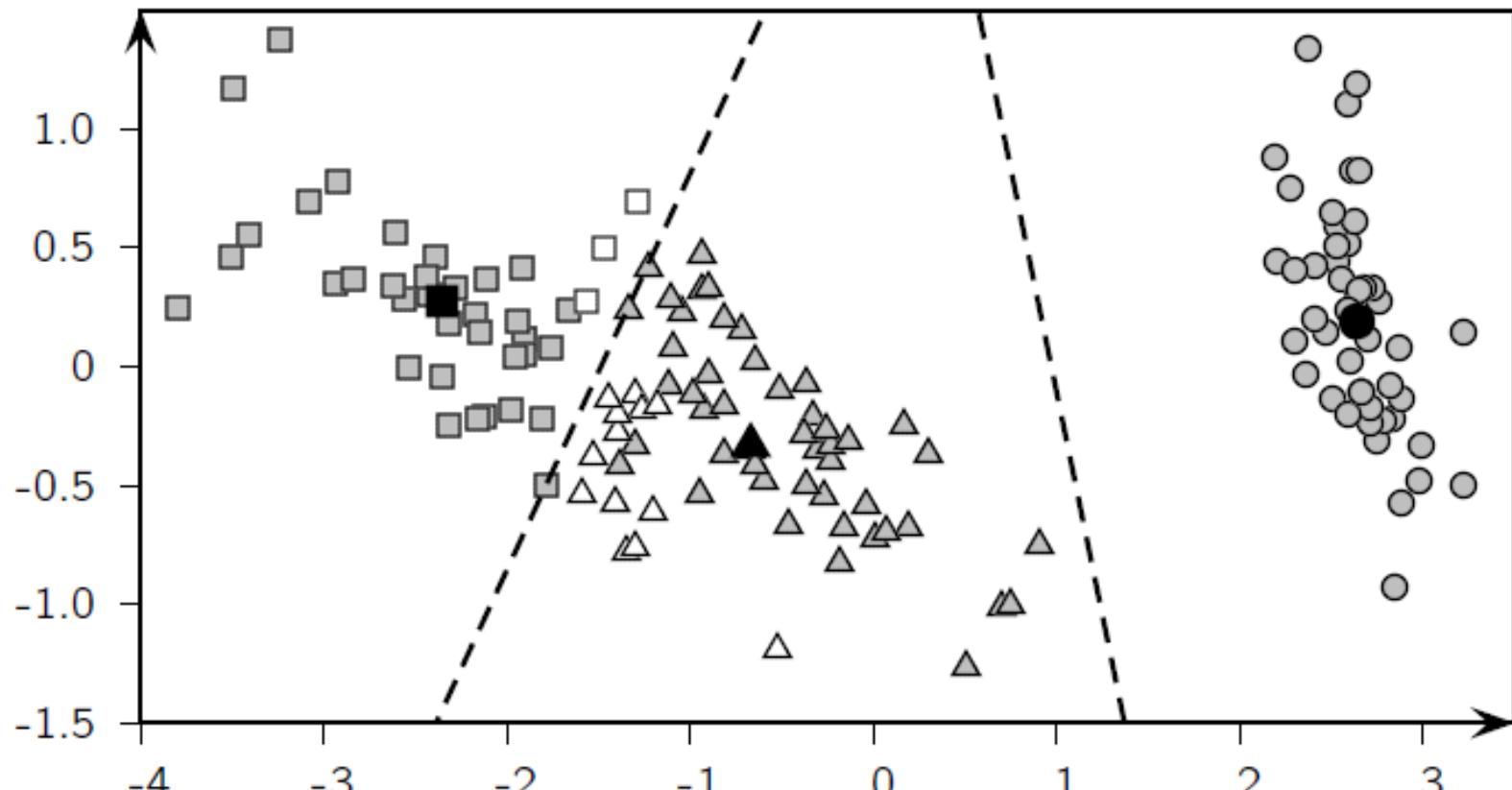
10 **until** $\sum_{i=1}^k \|\mu_i^t - \mu_i^{t-1}\|^2 \leq \epsilon$



(a) Random Initialization: $t = 0$



(b) Iteration: $t = 1$



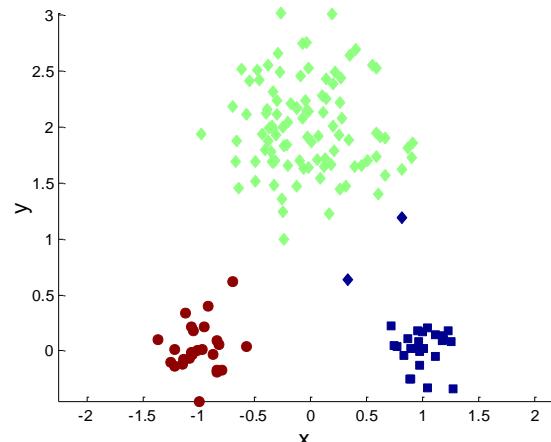
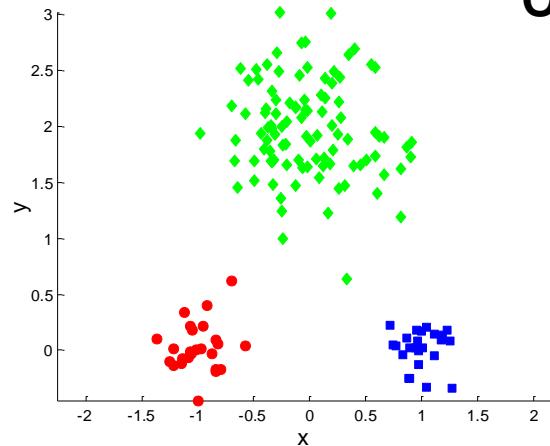
(c) Iteration: $t = 8$ (converged)

Issues and Limitations for K-means

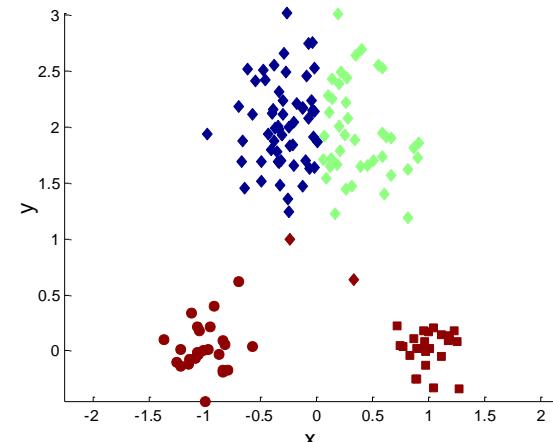
- How to choose initial centers?
- How to choose K?
- How to handle Outliers?
- Clusters different in
 - Shape
 - Density
 - Size

Two different K-means Clusterings

Original Points

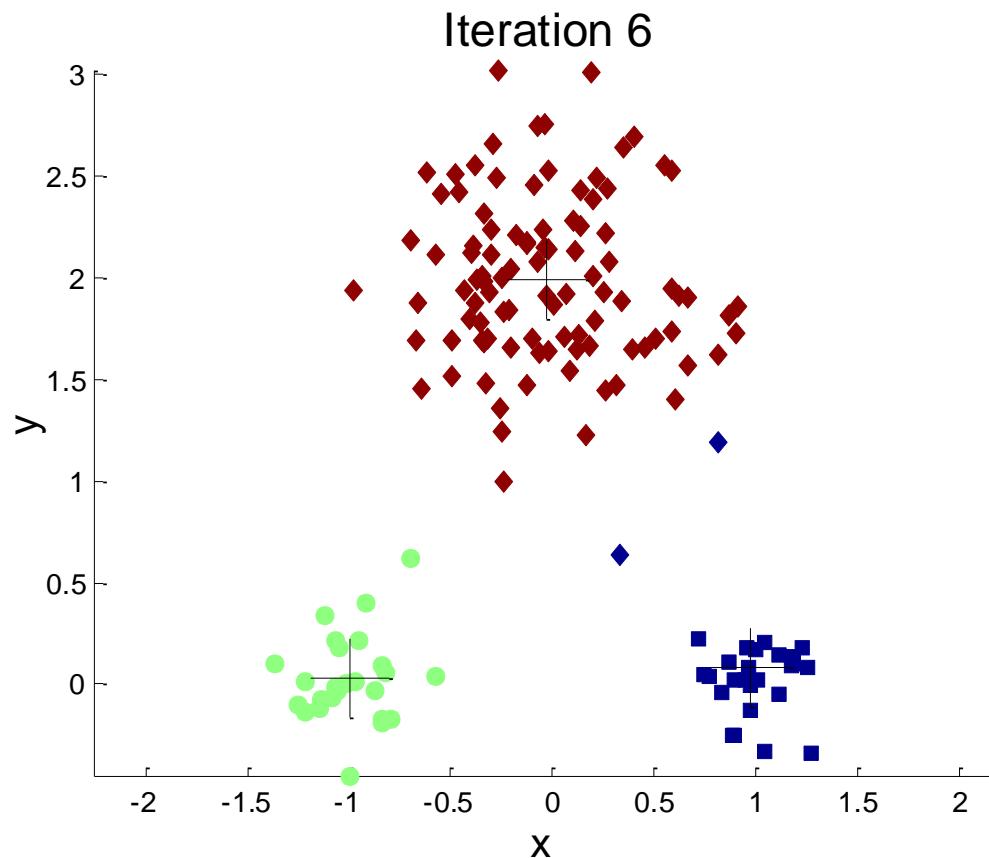


Optimal Clustering

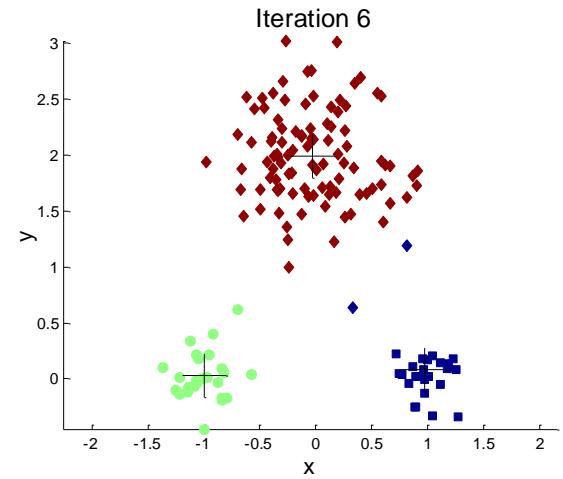
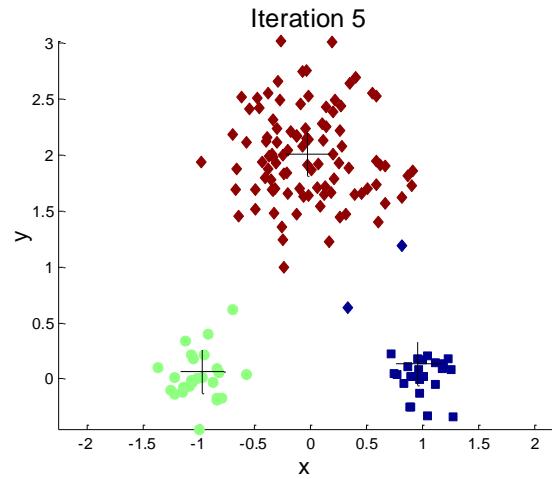
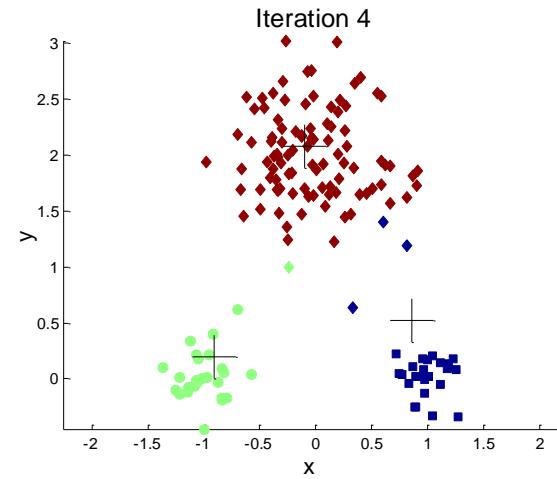
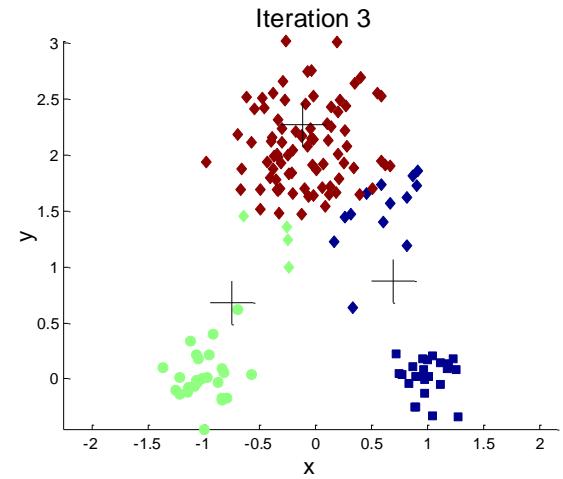
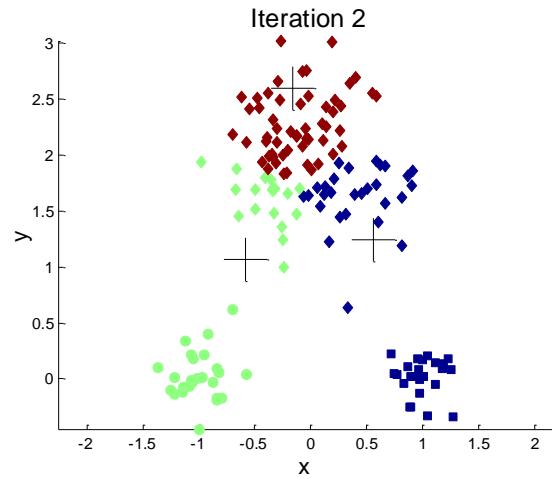
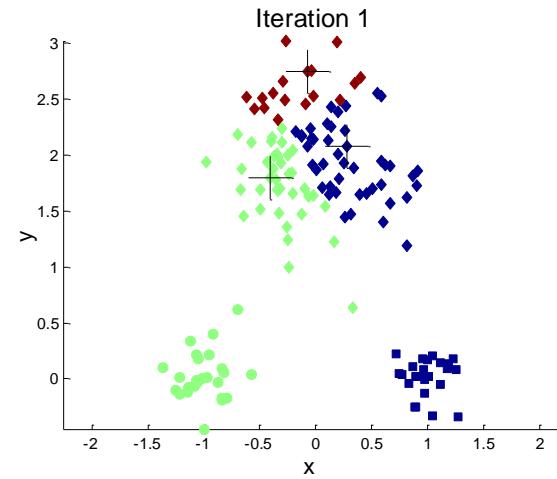


Sub-optimal Clustering

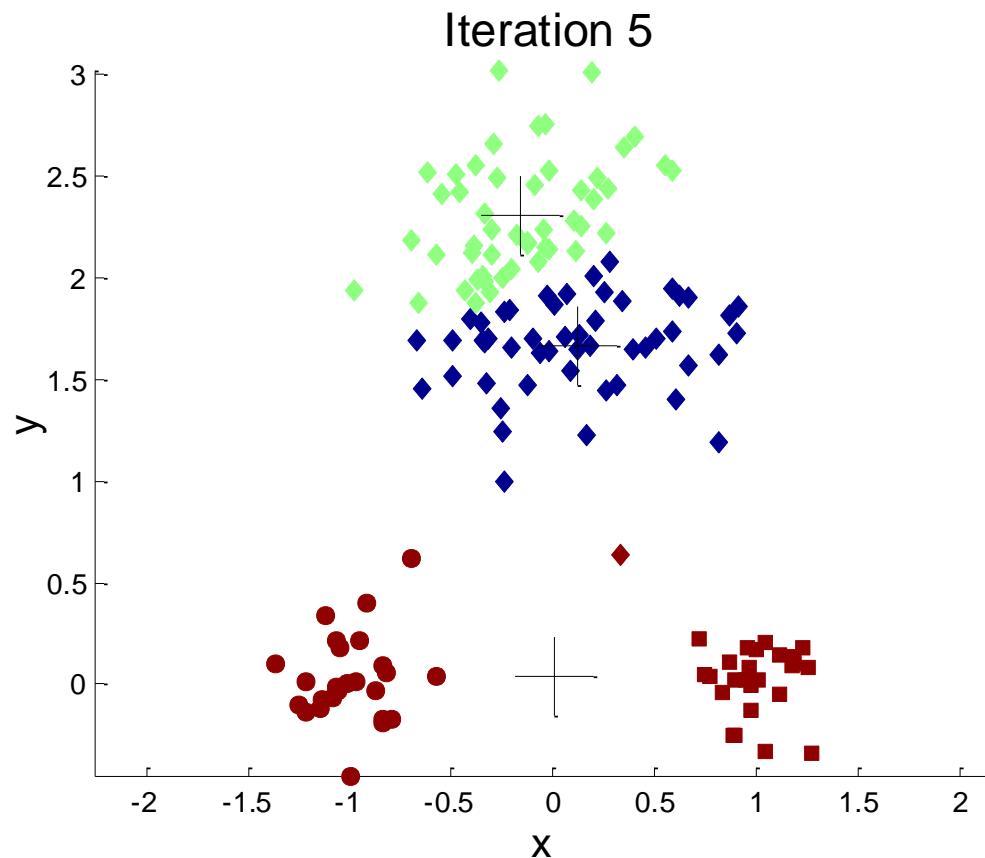
Importance of Choosing Initial Centroids



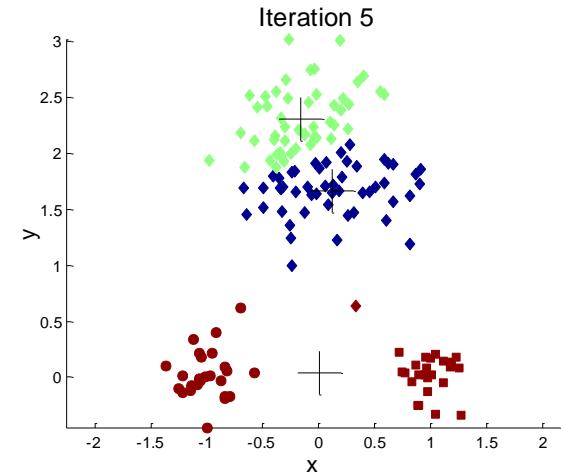
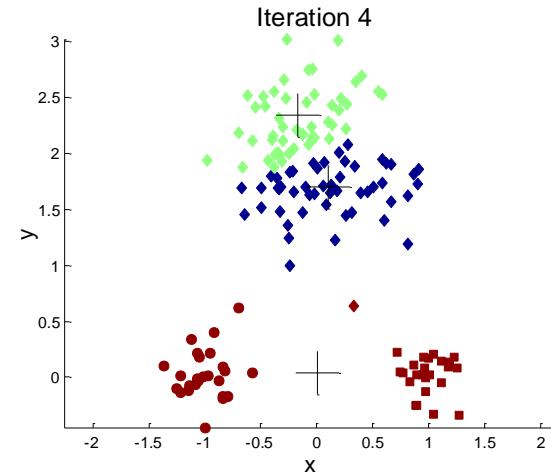
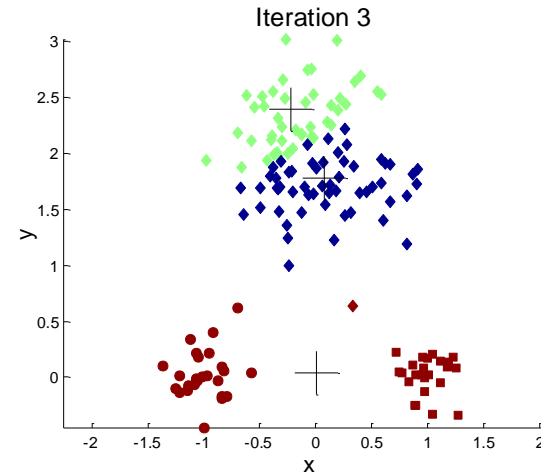
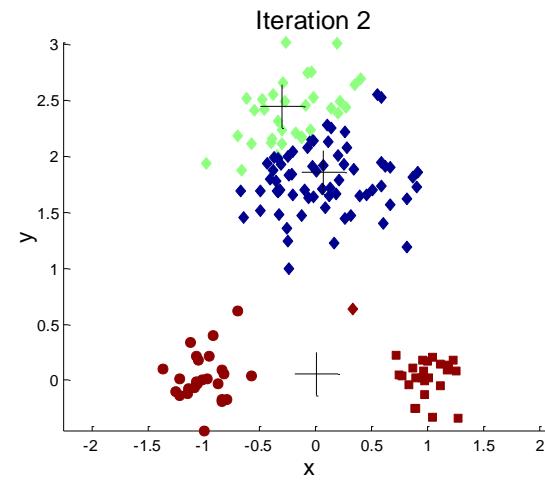
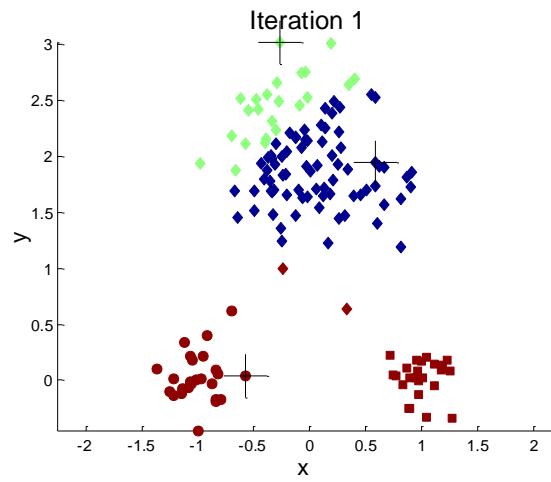
Importance of Choosing Initial Centroids



Importance of Choosing Initial Centroids ...



Importance of Choosing Initial Centroids ...



Problems with Selecting Initial Points

- If there are K ‘real’ clusters then the chance of selecting one centroid from each cluster is small.
 - Chance is relatively small when K is large
 - If clusters are the same size, n , then

$$P = \frac{\text{number of ways to select one centroid from each cluster}}{\text{number of ways to select } K \text{ centroids}} = \frac{K!n^K}{(Kn)^K} = \frac{K!}{K^K}$$

- For example, if $K = 10$, then probability = $10!/10^{10} = 0.00036$
- Sometimes the initial centroids will readjust themselves in ‘right’ way, and sometimes they don’t
- Consider an example of five pairs of clusters

Solutions to Initial Centroids Problem

- Multiple runs
 - Helps, but probability is not on your side
- Sample and use hierarchical clustering to determine initial centroids
- Select more than k initial centroids and then select among these initial centroids
 - Select most widely separated
- Postprocessing
- Bisecting K-means
 - Not as susceptible to initialization issues

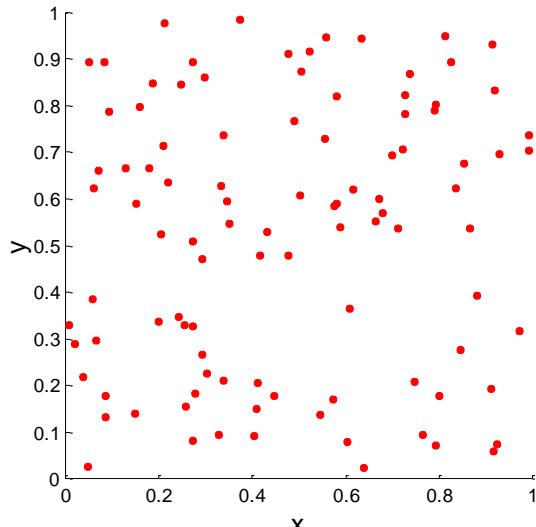
Cluster Validation

Cluster Validity

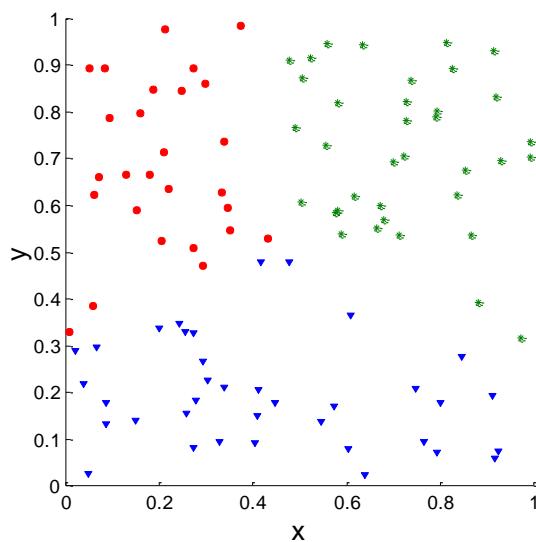
- For cluster analysis, the question is how to evaluate the “goodness” of the resulting clusters?
- But “clusters are in the eye of the beholder”!
- Then why do we want to evaluate them?
 - To avoid finding patterns in noise
 - To compare clustering algorithms
 - To compare two sets of clusters
 - To compare two clusters

Clusters found in Random Data

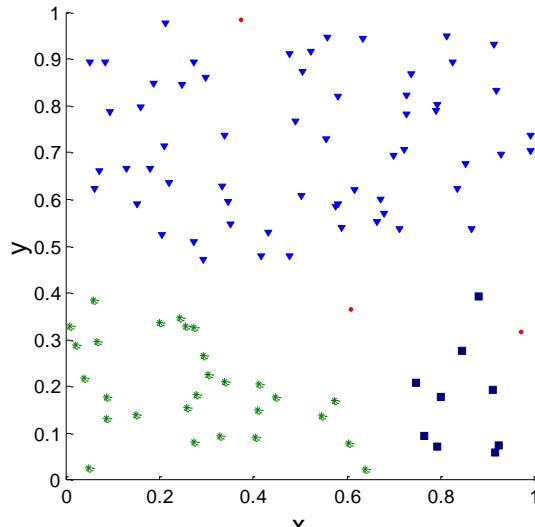
Random Points



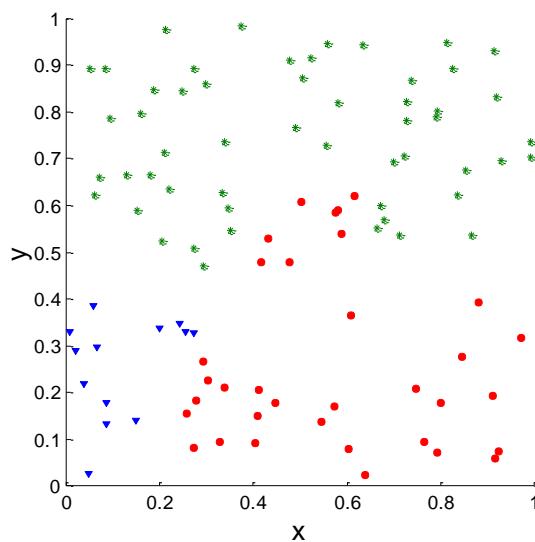
K-means



DBSCAN



Complete Link



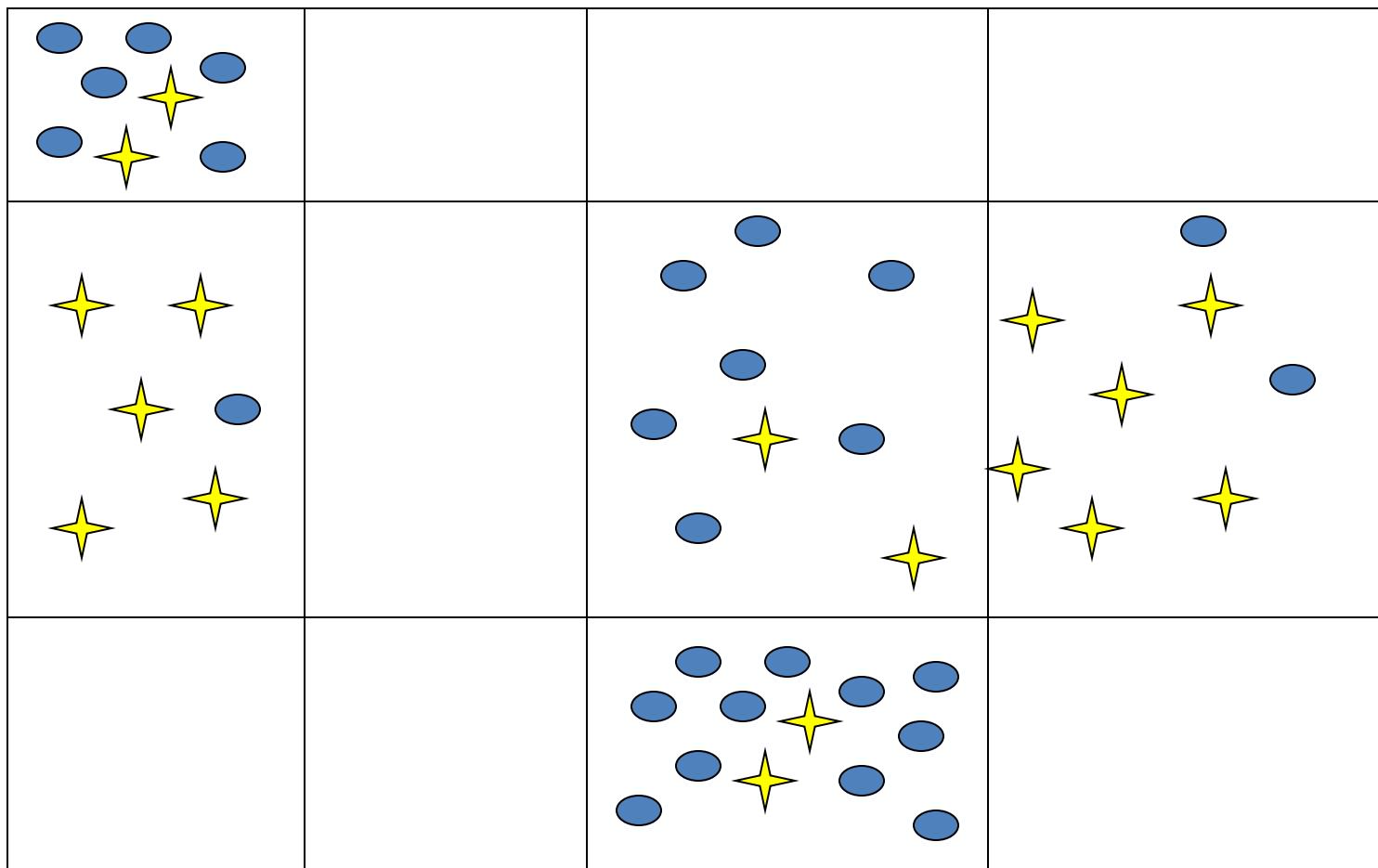
Different Aspects of Cluster Validation

1. Determining the **clustering tendency** of a set of data, i.e., distinguishing whether non-random structure actually exists in the data.
2. Comparing the results of a cluster analysis to externally known results, e.g., to externally given class labels.
3. Evaluating how well the results of a cluster analysis fit the data *without* reference to external information.
 - Use only the data
4. Comparing the results of two different sets of cluster analyses to determine which is better.
5. Determining the ‘correct’ number of clusters.

For 2, 3, and 4, we can further distinguish whether we want to evaluate the entire clustering or just individual clusters.

Classification: Basic Concepts and Decision Trees

A programming task



Classification: Definition

- Given a collection of records (*training set*)
 - Each record contains a set of *attributes*, one of the attributes is the *class*.
- Find a *model* for class attribute as a function of the values of other attributes.
- Goal: previously unseen records should be assigned a class as accurately as possible.
 - A *test set* is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

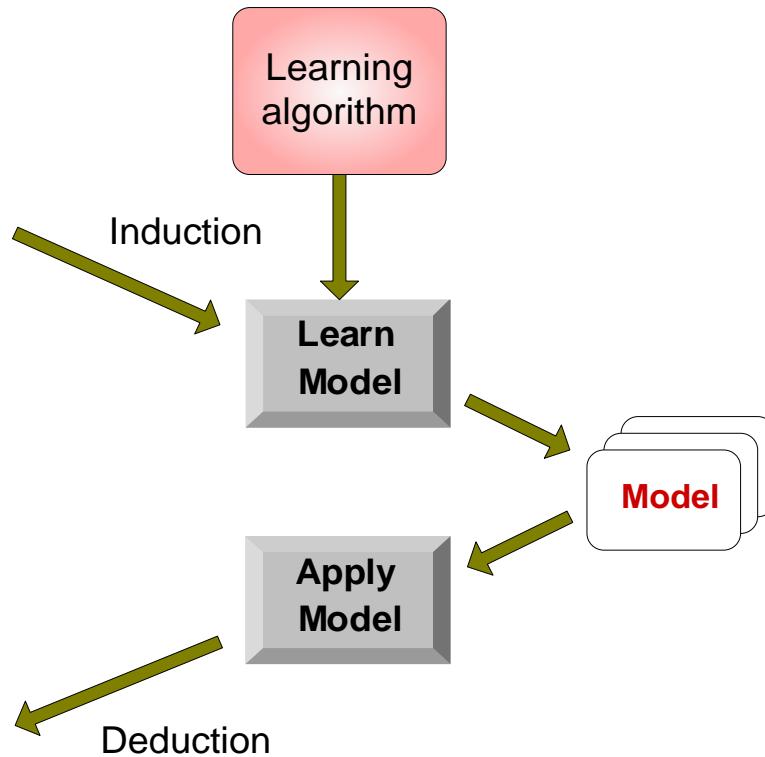
Illustrating Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

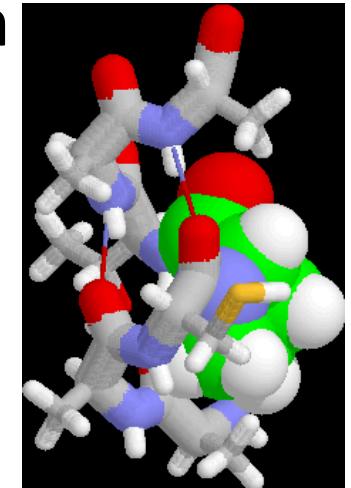
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Examples of Classification Task

- Predicting tumor cells as benign or malignant
- Classifying credit card transactions as legitimate or fraudulent
- Classifying secondary structures of protein as alpha-helix, beta-sheet, or random coil
- Categorizing news stories as finance, weather, entertainment, sports, etc



Classification Using Distance

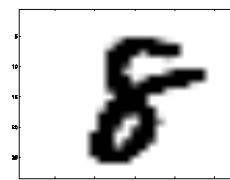
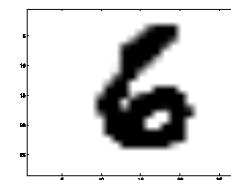
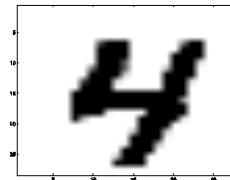
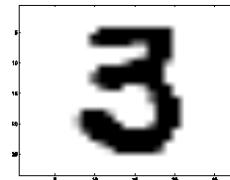
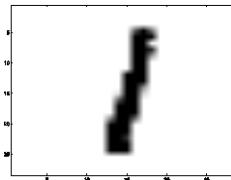
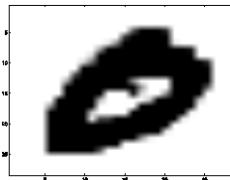
- Place items in class to which they are “closest”.
- Must determine distance between an item and a class.
- Classes represented by
 - ***Centroid***: Central value.
 - ***Medoid***: Representative point.
 - Individual points
- Algorithm: KNN

Classification Techniques

- Decision Tree based Methods
- Rule-based Methods
- Memory based reasoning
- Neural Networks
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines

A first example

Database of 20,000 images of handwritten digits, each
labeled by a human



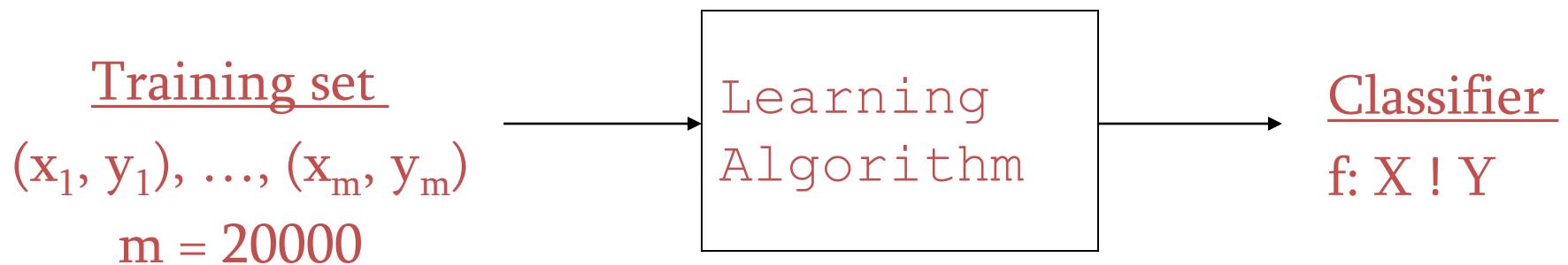
[28 x 28 greyscale; pixel values 0-255; labels 0-9]

Use these to learn a classifier which will label digit-images automatically...

The learning problem

Input space $X = \{0, 1, \dots, 255\}^{784}$

Output space $Y = \{0, 1, \dots, 9\}$



To measure how good f is: use a test set

[Our test set: 100 instances of each digit.]

A possible strategy

Input space X = $\{0, 1, \dots, 255\}^{784}$

Output space Y = $\{0, 1, \dots, 9\}$

Treat each image as a point in 784-dimensional Euclidean space

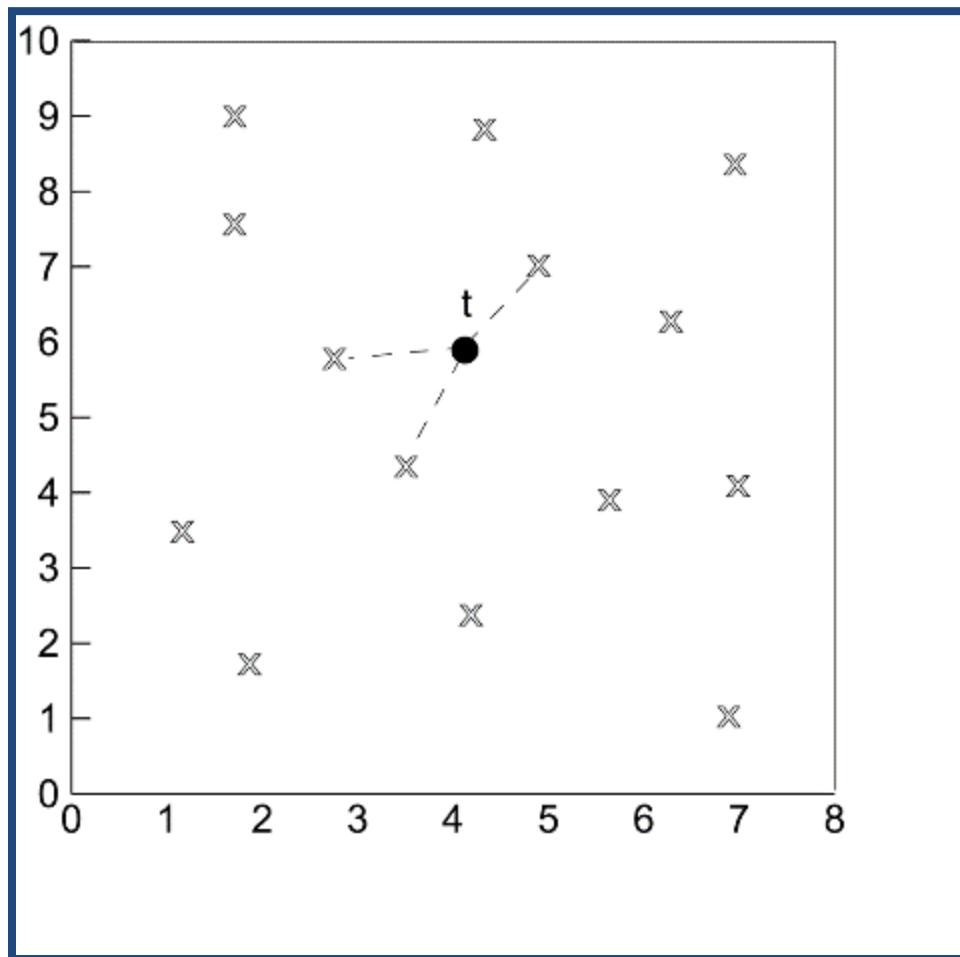
To classify a new image: find its nearest neighbor in the database (training set) and return that label

f = entire training set + search engine

K Nearest Neighbor (KNN):

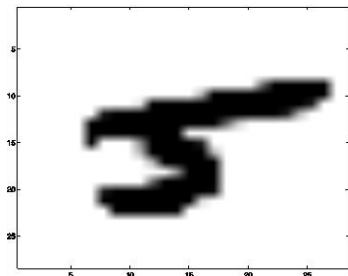
- Training set includes classes.
- Examine K items near item to be classified.
- New item placed in class with the most number of close items.
- $O(q)$ for each tuple to be classified. (Here q is the size of the training set.)

KNN

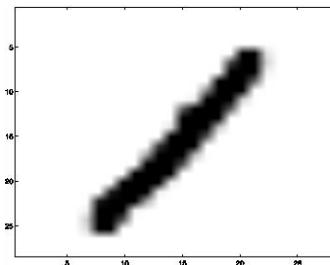
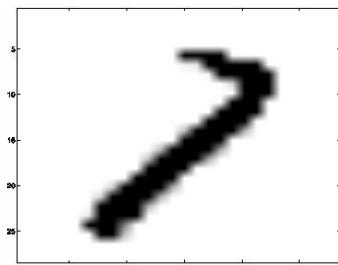
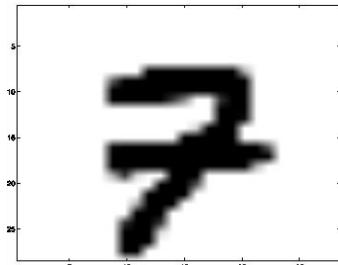
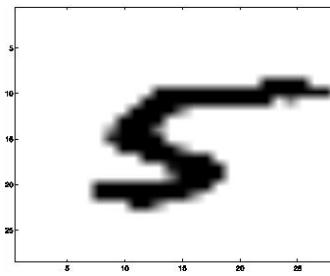


Nearest neighbor

Image to label



Nearest neighbor



Overall:

error rate = 6%

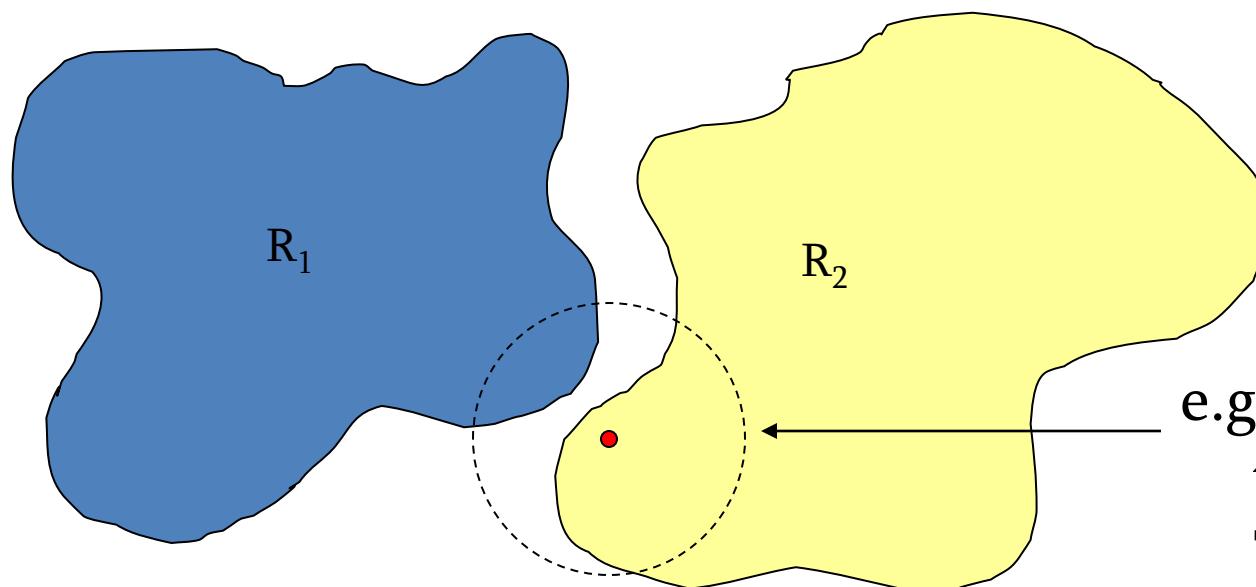
(on test set)

Question: what is
the error rate
for random
guessing?

What does it get wrong?

Who knows... but here's a hypothesis:

Each digit corresponds to some connected region of R^{784} . Some of the regions come close to each other; problems occur at these boundaries.



e.g. a random point in
this ball has only a
70% chance of
being in R_2

Nearest neighbor: pros and cons

Pros

Simple

Flexible

Excellent performance on a wide range of tasks

Cons

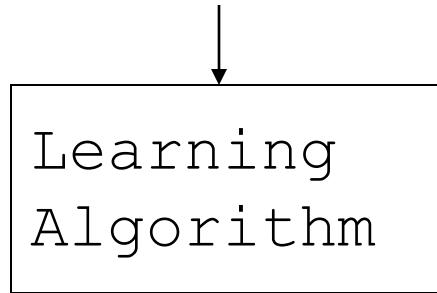
Algorithmic: time consuming – with n training points in \mathbb{R}^d , time to label a new point is $O(nd)$

Statistical: memorization, not learning!
no insight into the domain
would prefer a compact classifier

Postscript: learning models

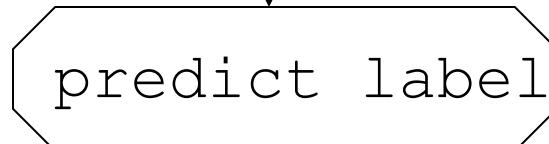
Batch learning

Training data_



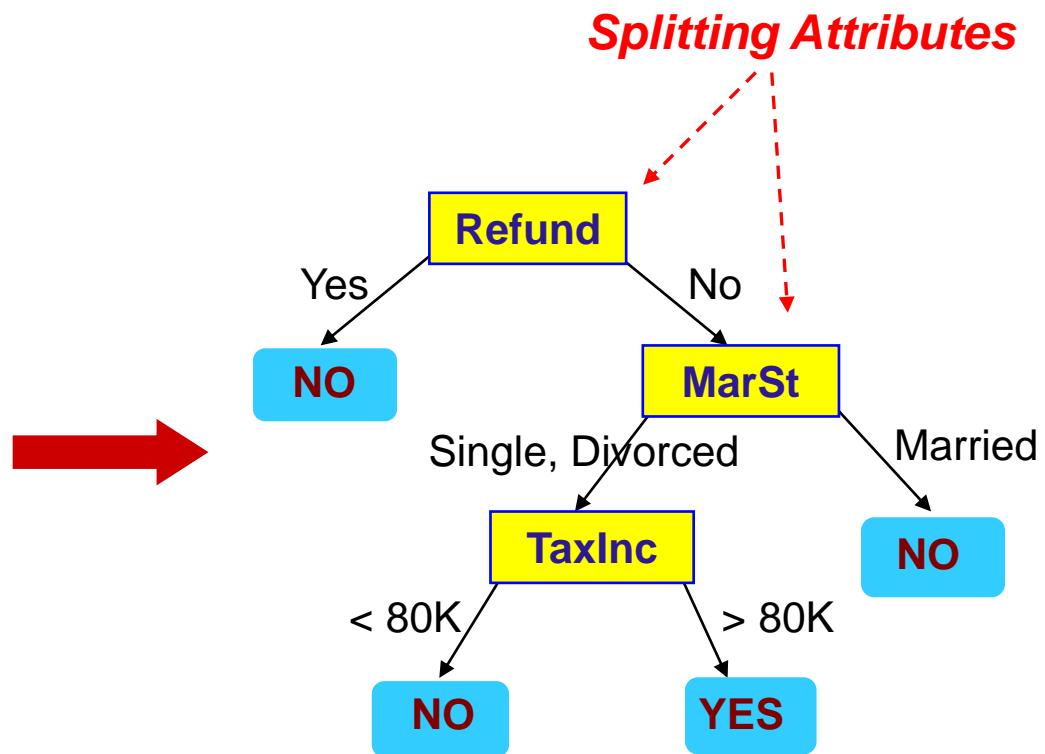
On-line learning

See a new point $x_{_}$



Example of a Decision Tree

Tid	Categorical				continuous class
	Refund	Marital Status	Taxable Income	Cheat	
1	Yes	Single	125K	No	
2	No	Married	100K	No	
3	No	Single	70K	No	
4	Yes	Married	120K	No	
5	No	Divorced	95K	Yes	
6	No	Married	60K	No	
7	Yes	Divorced	220K	No	
8	No	Single	85K	Yes	
9	No	Married	75K	No	
10	No	Single	90K	Yes	

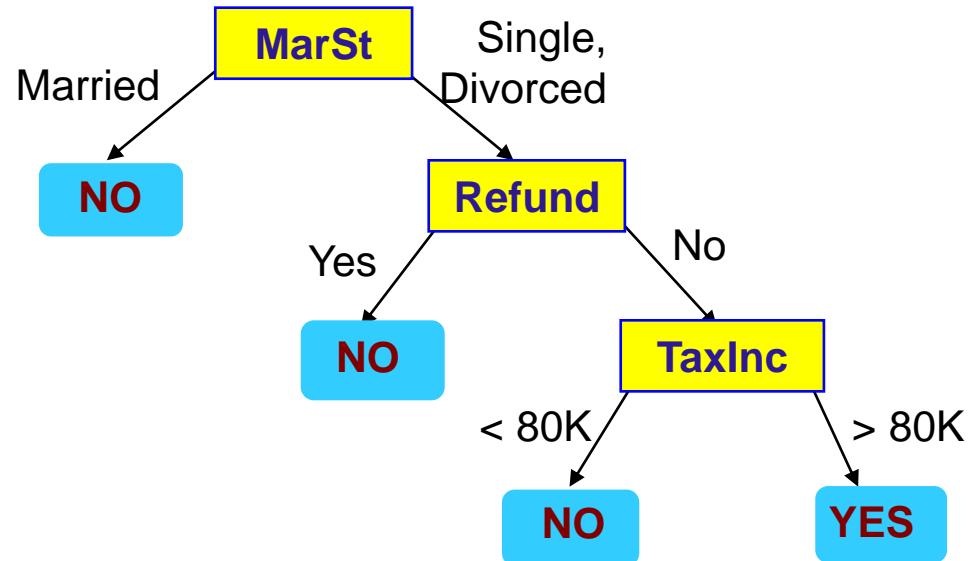


Training Data

Model: Decision Tree

Another Example of Decision Tree

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



There could be more than one tree that fits the same data!

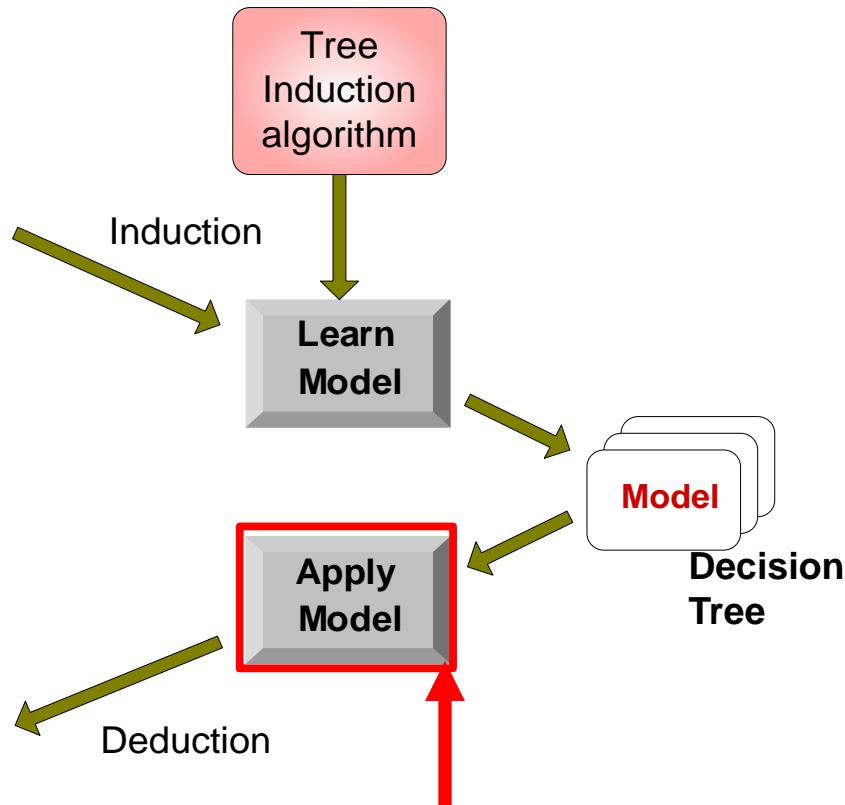
Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

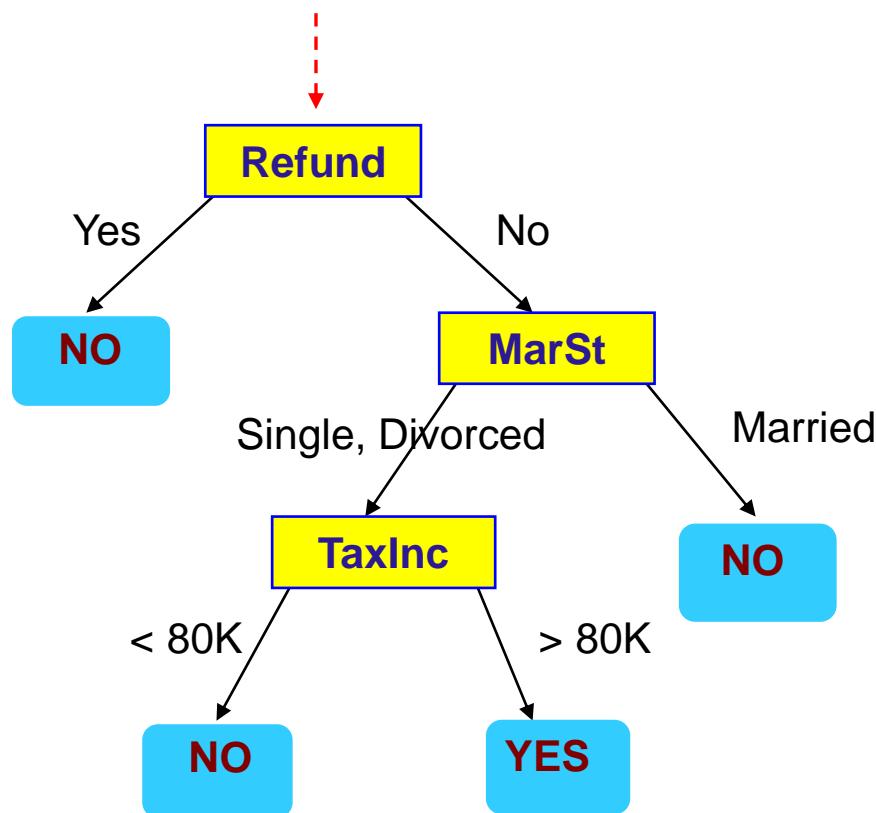
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Apply Model to Test Data

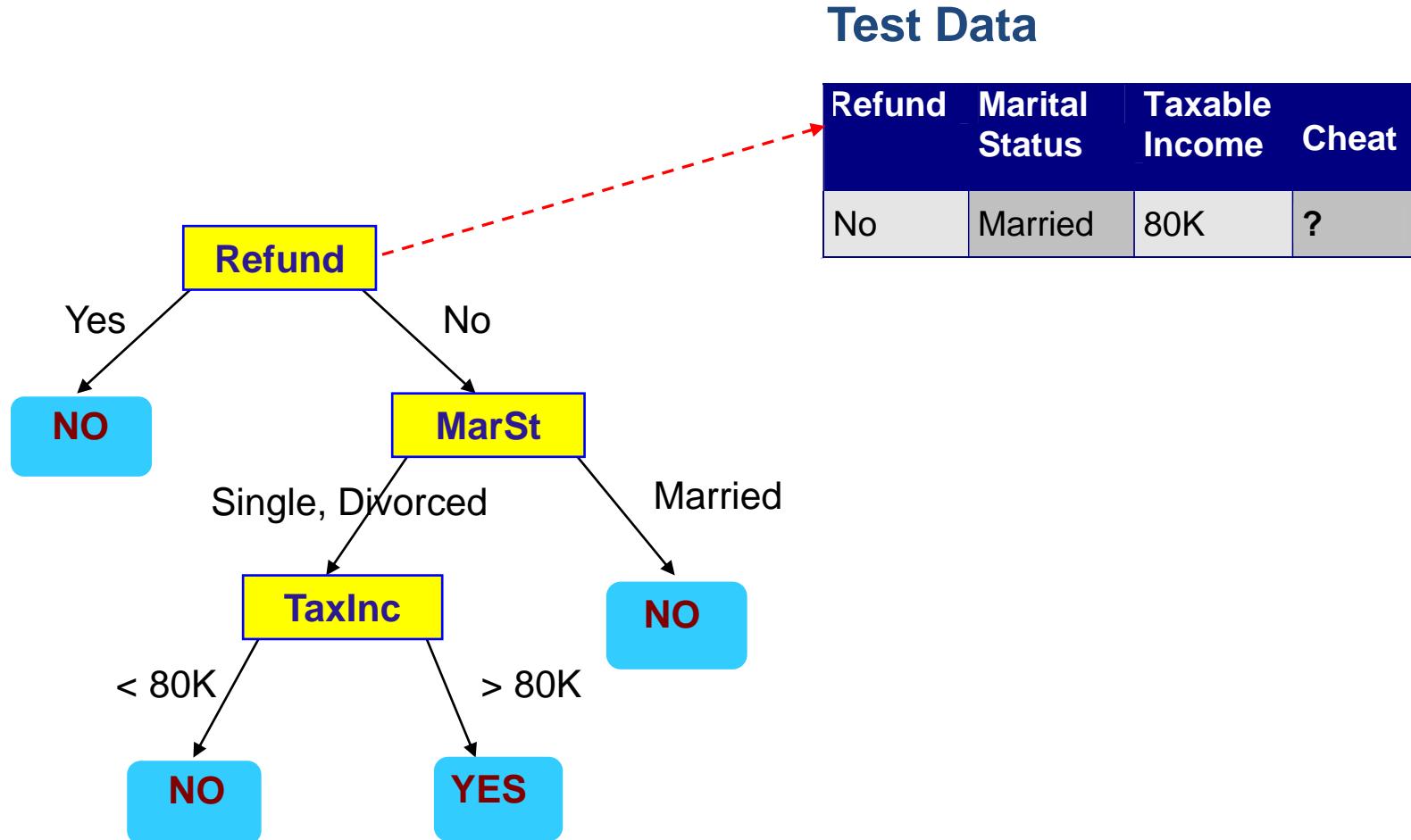
Start from the root of tree.



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

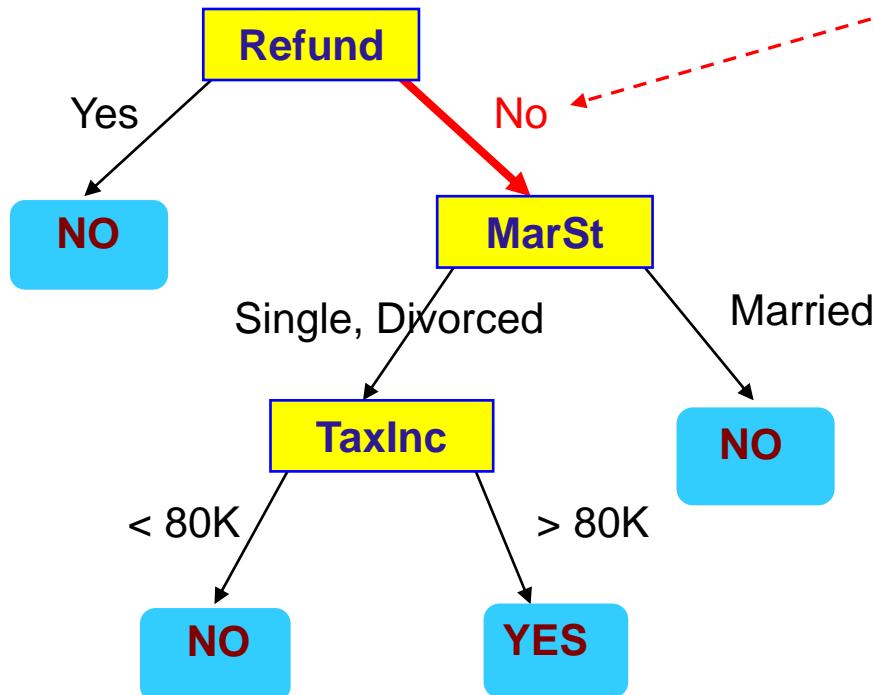
Apply Model to Test Data



Apply Model to Test Data

Test Data

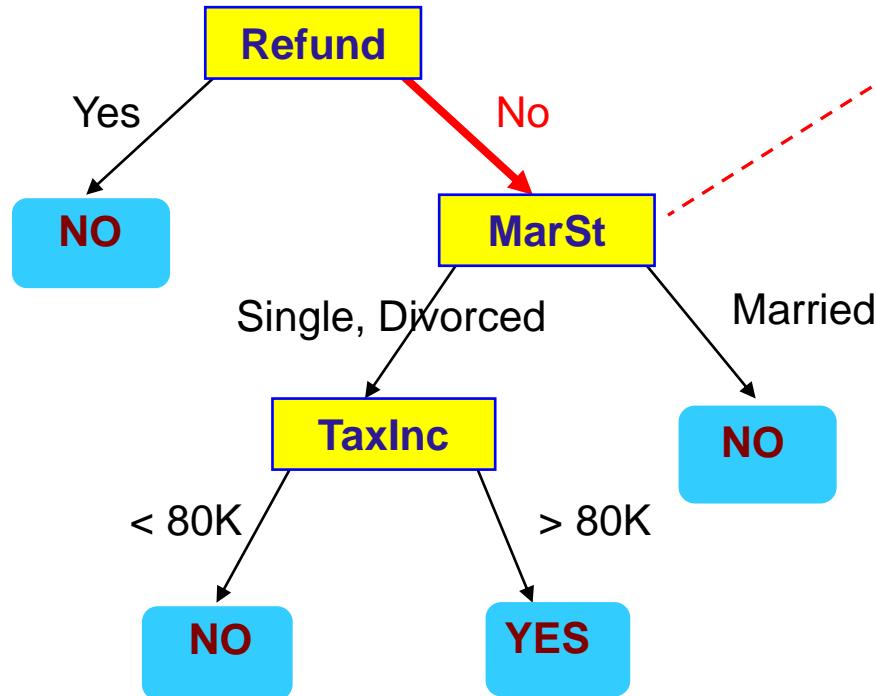
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

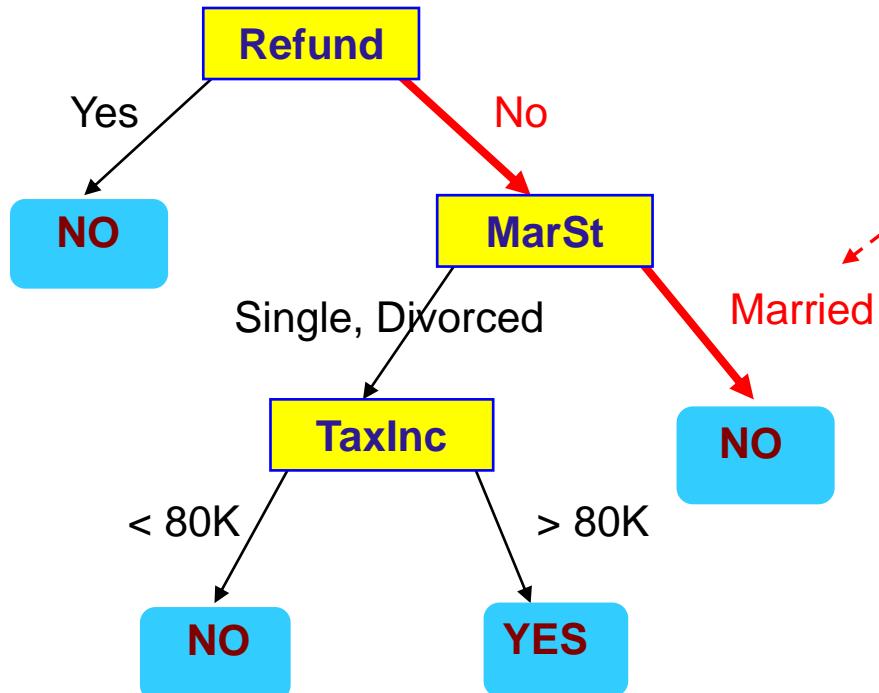
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



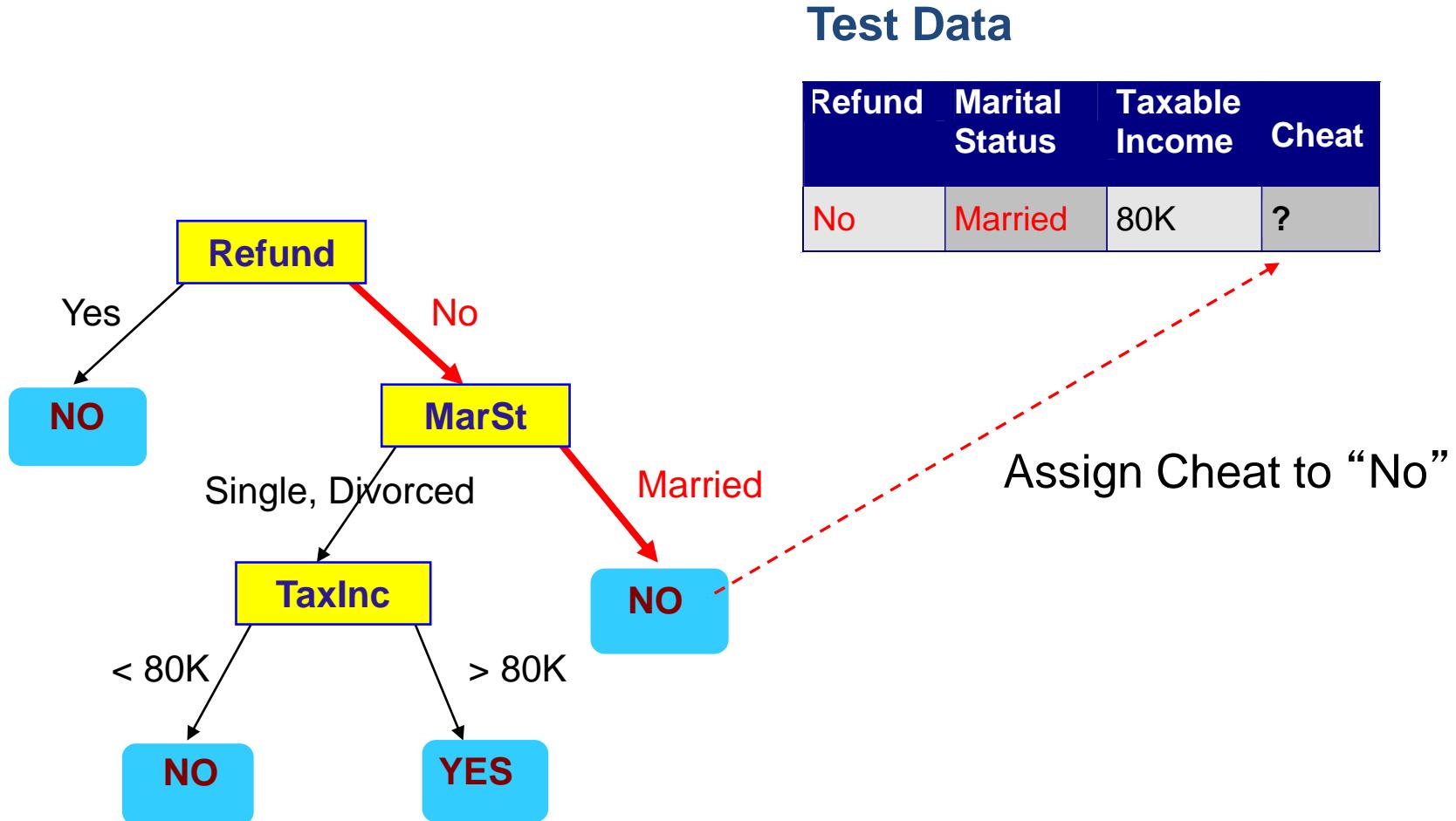
Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data



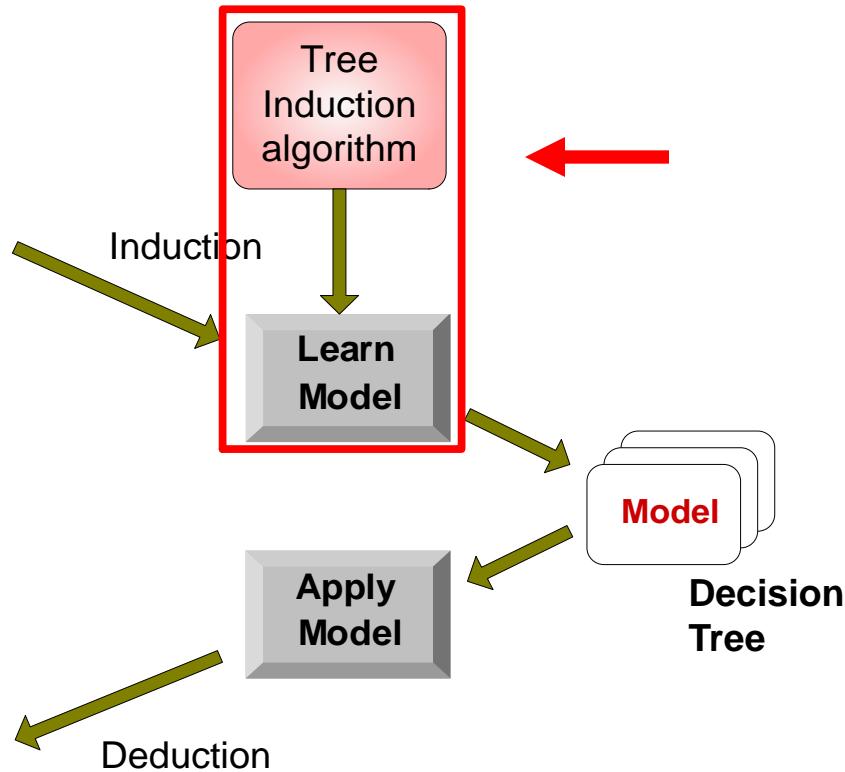
Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Decision Tree Induction

- Many Algorithms:
 - Hunt's Algorithm (one of the earliest)
 - CART
 - ID3, C4.5
 - SLIQ, SPRINT

Big Data

Dilip K. Prasad

Outline

- Intro to Big data
- ETL (Extract-Transform-Load)
- Introduction to Hadoop

Big Data

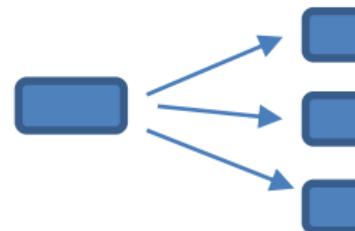
What is Big Data?

How big is the data to be termed as big data?

RDBS

Bank

NoSQL



Structured -
tabular

Unstructured
– image, audio
video, etc.

3 V's of data

Volume – amount of data

Velocity – speed of data in/out

Variety – text, image, video, etc.

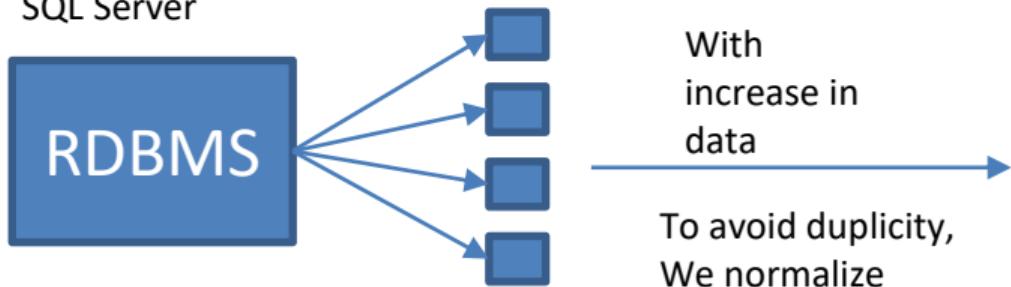
5 V(3V+value,veracity) 7V(5V + variability, visualization)

What size of the data can be termed as Big Data?

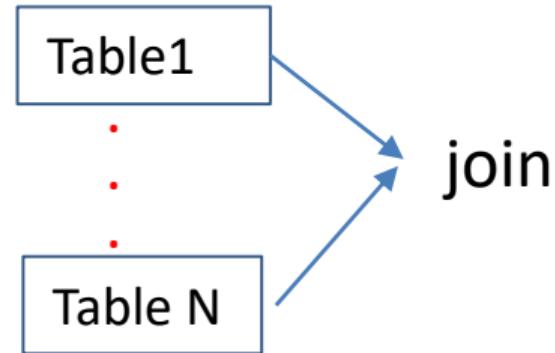
Anything can be big data depending on the system which will process the data

Big data ??

Oracle/MySQL/
SQL Server

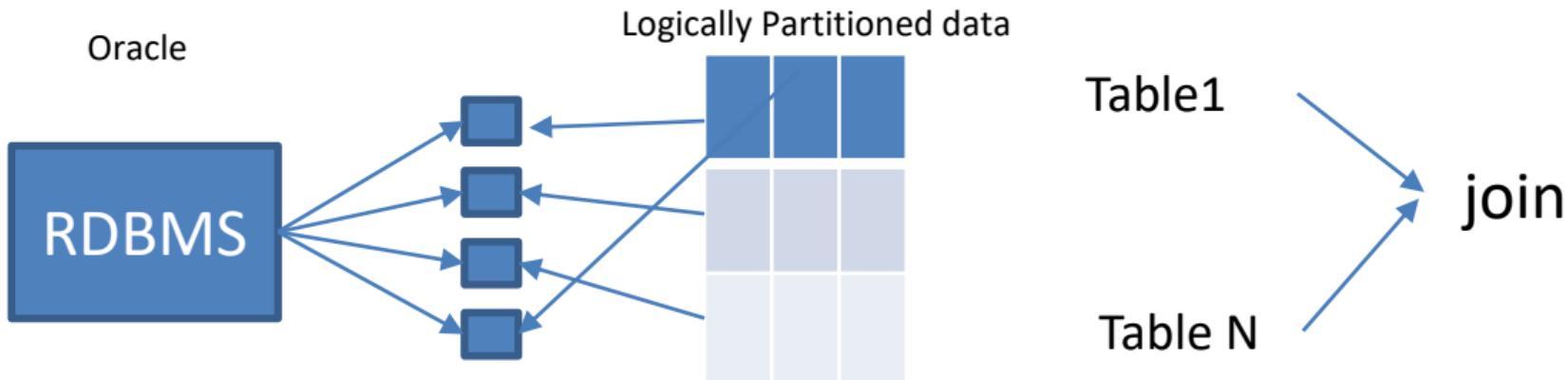


Small
table size
from KB
to TB



What to do in RDBMS if table size increases?

ETL (Extract-Transform-Load)



?

Nosql

Queries become slow with increase in size
Can not process unstructured data
Cost

ETL (Extract-Transform-Load)

Polyglot Persistence

Mysql

CRM

Oracle

Core
Banking

XML

Subsidiary

JSON

Social media

Flat File

Customer care log file

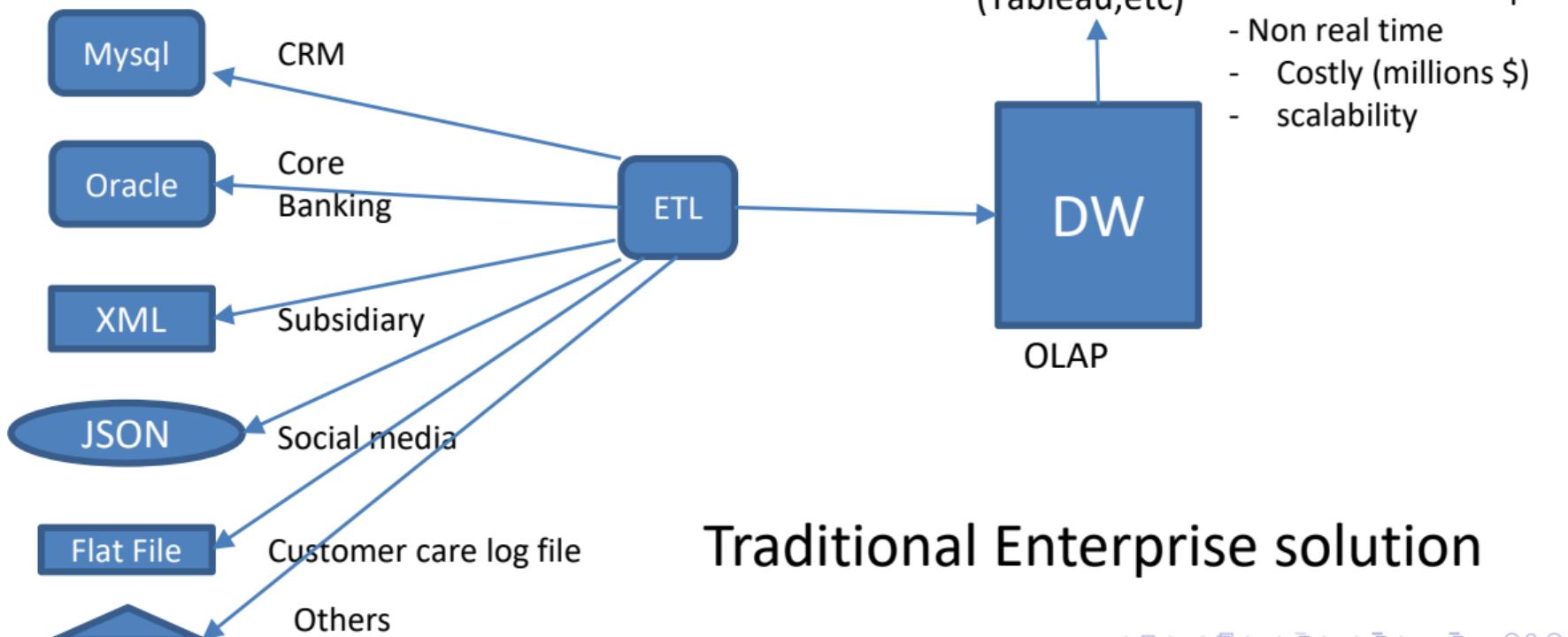


Others

DW

ETL (Extract-Transform-Load)

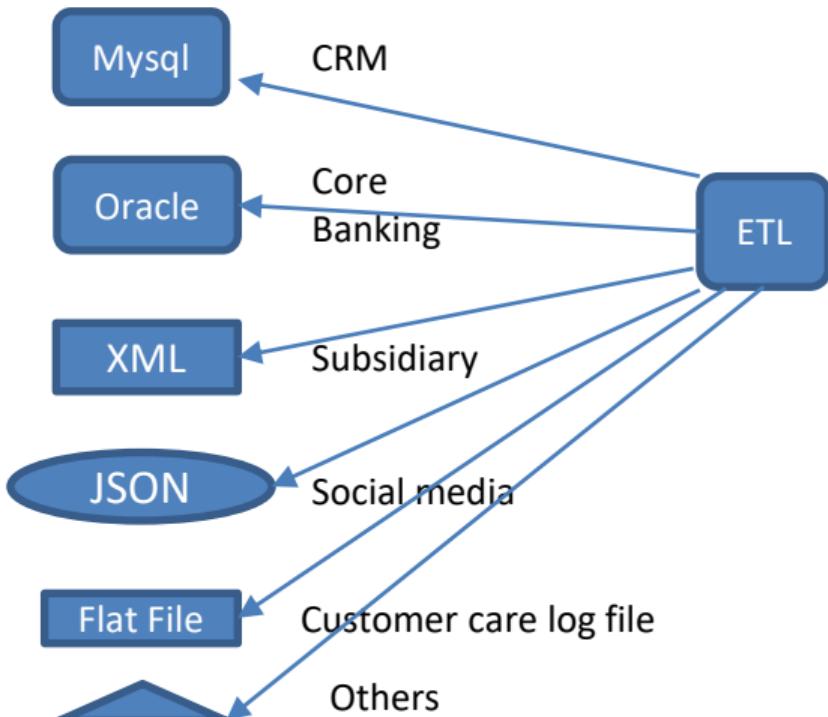
Polyglot Persistence



Traditional Enterprise solution

Storage

Polyglot Persistence



Cloudera (2008)
HortonWorks
MapR

Hadoop

BigInsights
HDInsights

Does NAS/SAN solve
our problem?

Single System → NAS

SAN

Hadoop History

The Google File System

2003

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung
Google



MapReduce: Simplified Data Processing on Large Clusters

2004

Jeffrey Dean and Sanjay Ghemawat
jeff@google.com, sanjay@google.com
Google, Inc.



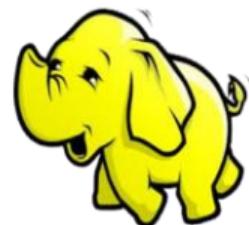
Bigtable: A Distributed Storage System for Structured Data

2006

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach,
Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber,
(c) 2006 by the authors. Licensee ACM, Inc. 1544-7555/06/0100-0001 \$5.00

Google, Inc.

Abstract
Bigtable is a distributed storage system for managing sparse data for a distributed system in a very simple way. It provides a fault-tolerant and highly available service for structured data stored by thousands of commodity servers. Many projects at Google store data in Bigtable, including search, Google Earth, and Google Earthquake. This paper describes the design of Bigtable and its performance, both in terms of data size (from TBs to terabytes) and number of columns (and hence rows).



2005: Doug Cutting and Michael J. Cafarella developed Hadoop to support distribution for the Nutch search engine project.

The project was funded by Yahoo.

2006: Yahoo gave the project to Apache Software Foundation.

Hadoop milestones

2008 - Hadoop Wins Terabyte Sort Benchmark (sorted 1 terabyte of data in 209 seconds, compared to previous record of 297 seconds)

2009 - Avro and Chukwa became new members of Hadoop Framework family

2010 - Hadoop's Hbase, Hive and Pig subprojects completed, adding more computational power to Hadoop framework

2011 - ZooKeeper Completed

2013 - Hadoop 1.1.2 and Hadoop 2.0.3 alpha.

- Ambari, Cassandra, Mahout have been added



Thanks

Hadoop & HDFS

Dilip K. Prasad

Outline

- Introduction to Hadoop
- Distributed Computing
- Hadoop Architecture
- HDFS File Storage
- Introduction to Oozie and HDFS Processing
- Hadoop Clusters
- Hadoop Ecosystem

Hadoop History

The Google File System

2003

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung
Google



MapReduce: Simplified Data Processing on Large Clusters

2004

Jeffrey Dean and Sanjay Ghemawat
jeff@google.com, sanjay@google.com
Google, Inc.



Bigtable: A Distributed Storage System for Structured Data

2006

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach,
Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber,
(bigtable-research@googlegroups.com)

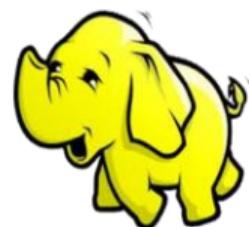
Google, Inc.



2005: Doug Cutting and Michael J. Cafarella developed Hadoop to support distribution for the Nutch search engine project.

The project was funded by Yahoo.

2006: Yahoo gave the project to Apache Software Foundation.



Hadoop milestones

2008 - Hadoop Wins Terabyte Sort Benchmark (sorted 1 terabyte of data in 209 seconds, compared to previous record of 297 seconds)

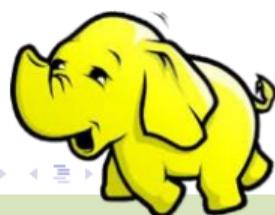
2009 - Avro and Chukwa became new members of Hadoop Framework family

2010 - Hadoop's Hbase, Hive and Pig subprojects completed, adding more computational power to Hadoop framework

2011 - ZooKeeper Completed

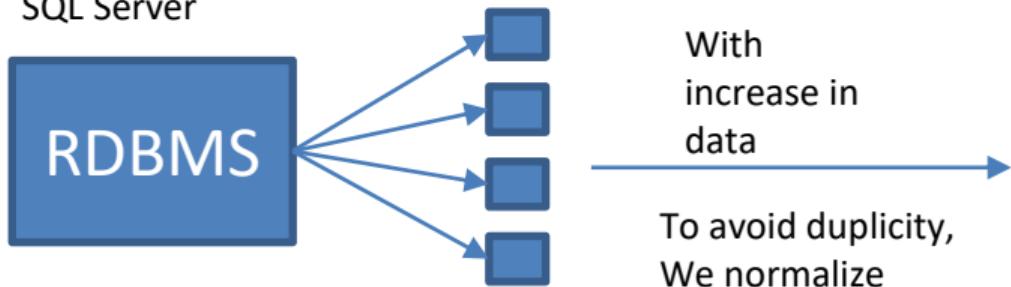
2013 - Hadoop 1.1.2 and Hadoop 2.0.3 alpha.

- Ambari, Cassandra, Mahout have been added



Big data ??

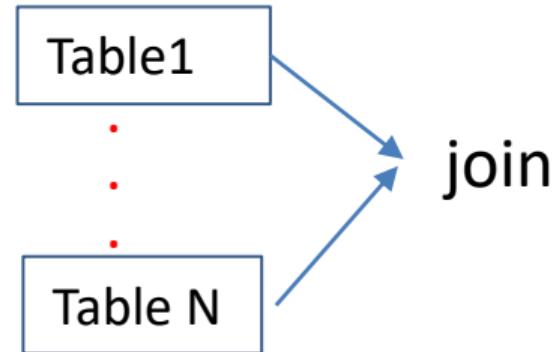
Oracle/MySQL/
SQL Server



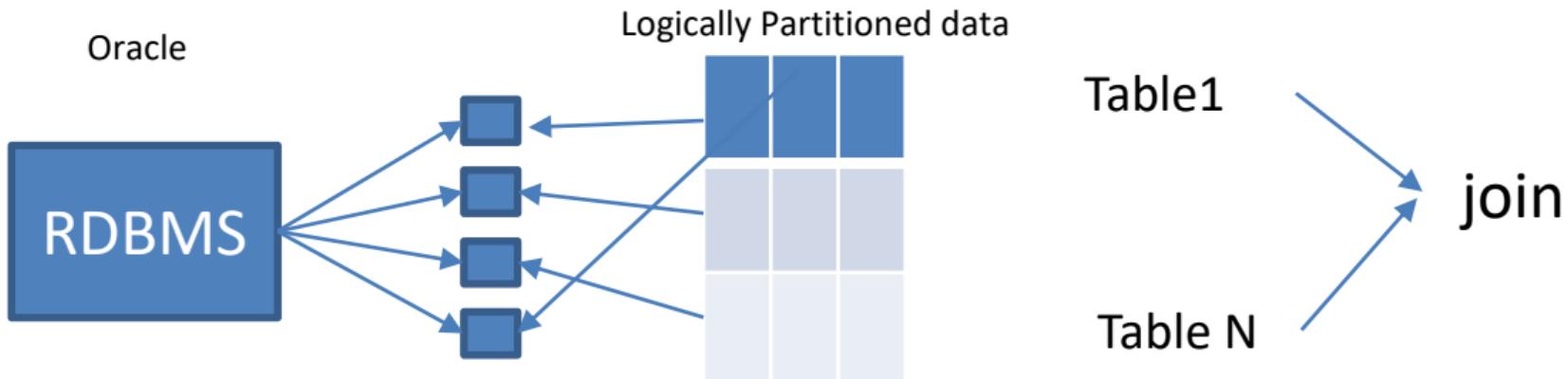
Small
table size
from KB
to TB

With
increase in
data

To avoid duplicity,
We normalize



ETL (Extract-Transform-Load)



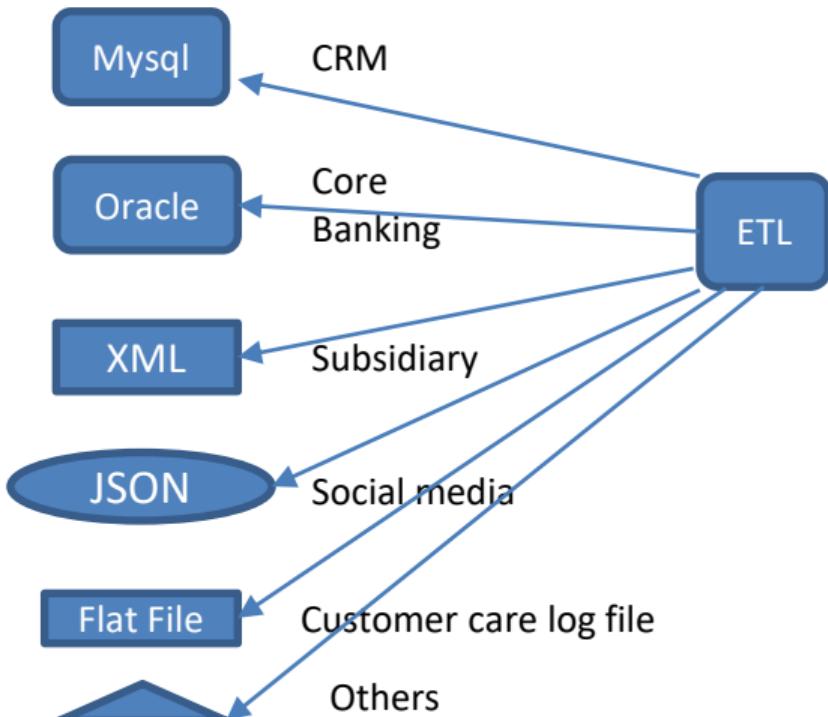
?

Nosql

Queries become slow with increase in size
Can not process unstructured data
Cost

Hadoop

Polyglot Persistence



Cloudera (2008)
HortonWorks
MapR

Hadoop

BigInsights
HDInsights

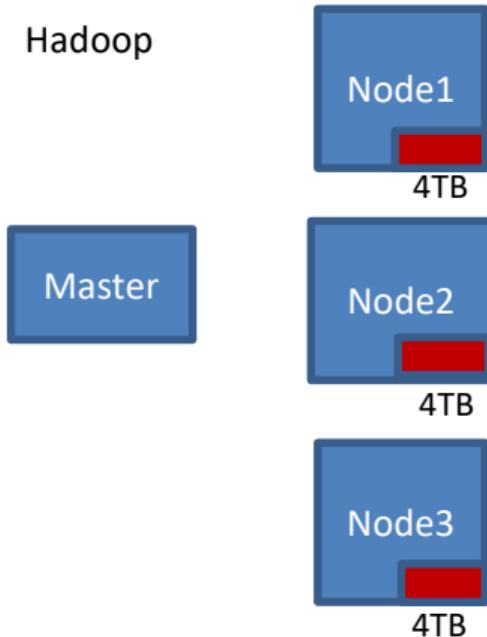
Does NAS/SAN solve
our problem?

Single System →
NAS

SAN

Distributed Computing and Hadoop

Hadoop

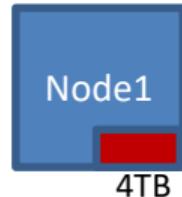


Is Hadoop a software?

Distributed Computing and Hadoop

Hadoop

Master



4TB



4TB



4TB

Hadoop is a framework

- Dynamically we can add/remove machine/node of various configuration
- We can add ~30000 machines/nodes
- Scaling out architecture
- Cheap commodity hardware used
- Not crash free!!! (reliability??)

Hadoop Architecture

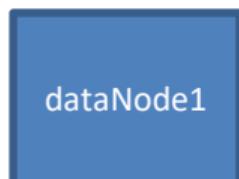
HDFS – Storage (Hadoop distributed file system)

YARN - Resource manager

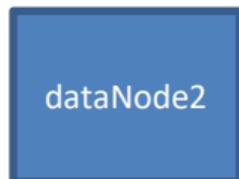
MapReduce – Processing

Access-edit-
restore – delete
old file

Hadoop



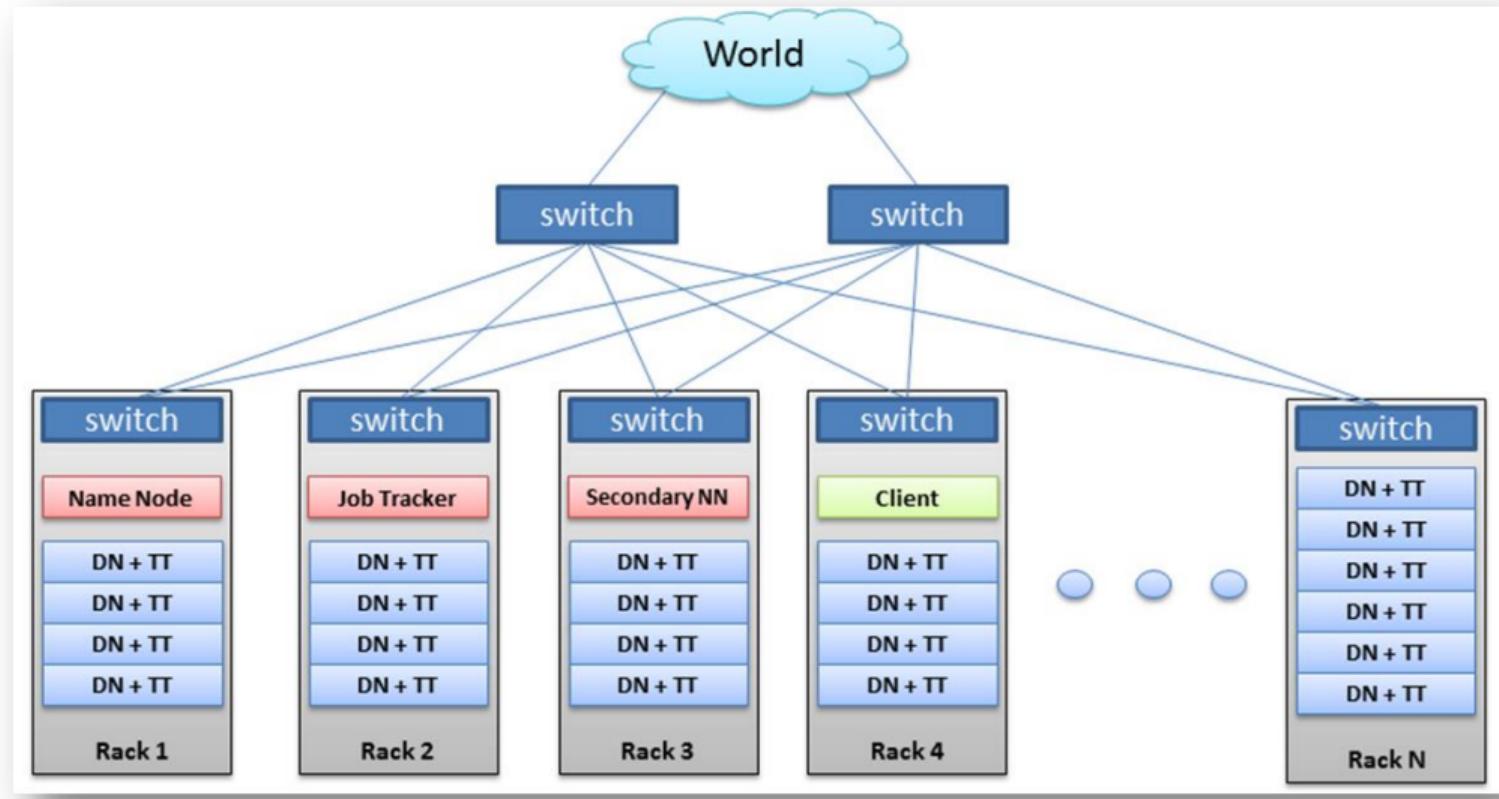
nameNode



dataNode3

HDFS you can store data in any format
You **can not** edit the data on the HDFS system
But you **can** append the data
Specific Seek is not possible
Hadoop is meant for storing large amount of data fast
Hadoop is meant for analysis of the data
Limitations of HDFS is handled in Processing step

Hadoop Architecture



Hadoop Architecture

NameNode:

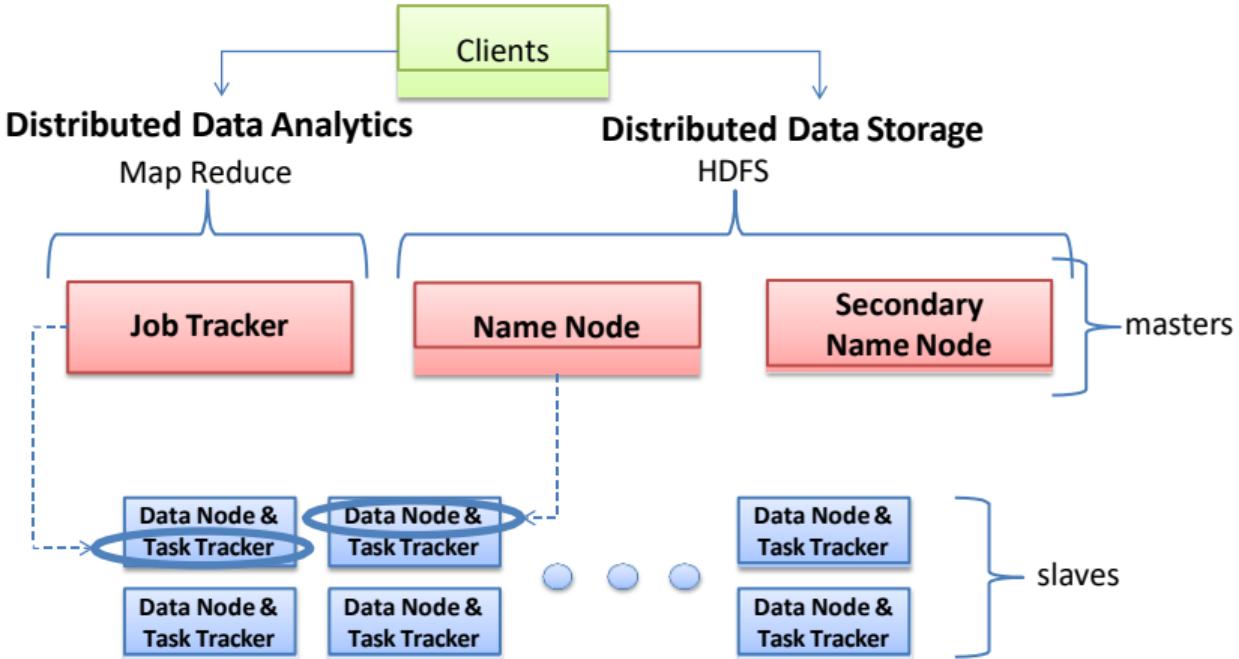
- Stores metadata for the files, like the directory structure of a typical FS.
- The server holding the NameNode instance is quite crucial, as there is only one.
- Transaction log for file deletes/adds, etc. Does not use transactions for whole blocks or file-streams, only metadata.
- Handles creation of more replica blocks when necessary, after a DataNode failure

Hadoop Architecture

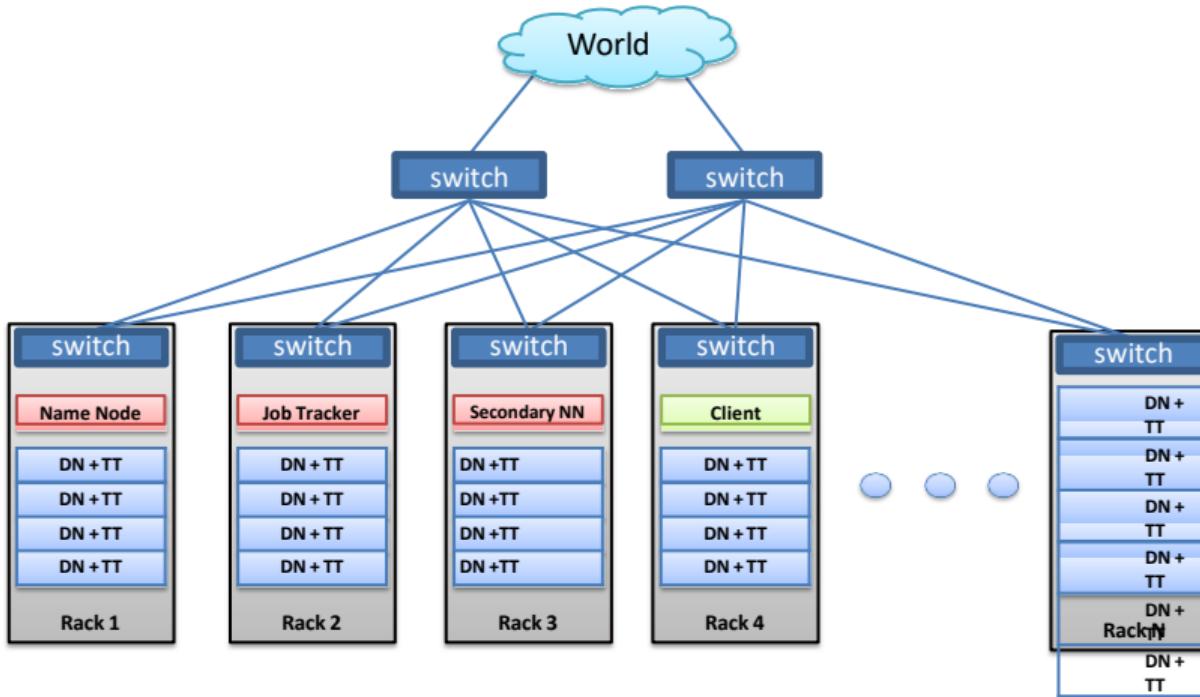
DataNode:

- Stores the actual data in HDFS
- Can run on any underlying filesystem (ext3/4, NTFS, etc)
- Notifies NameNode of what blocks it has
- NameNode replicates blocks 2x in local rack, 1x elsewhere

Hadoop Server Roles



Hadoop Cluster



Typical Workflow

- Load data into the cluster (HDFS writes)
- Analyze the data (Map Reduce)
- Store results in the cluster (HDFS writes)
- Read the results from the cluster (HDFS reads)

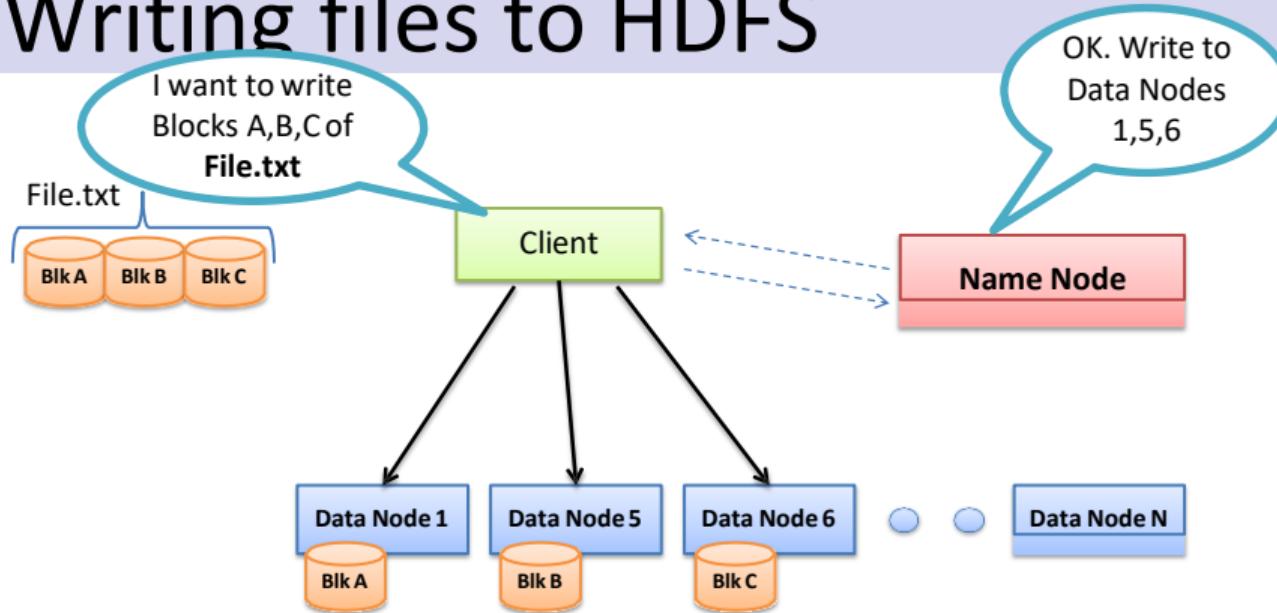
Sample Scenario:

How many times did our customers type the word “**Fraud**” into emails sent to customer service?

Huge file containing all emails sent to customer service

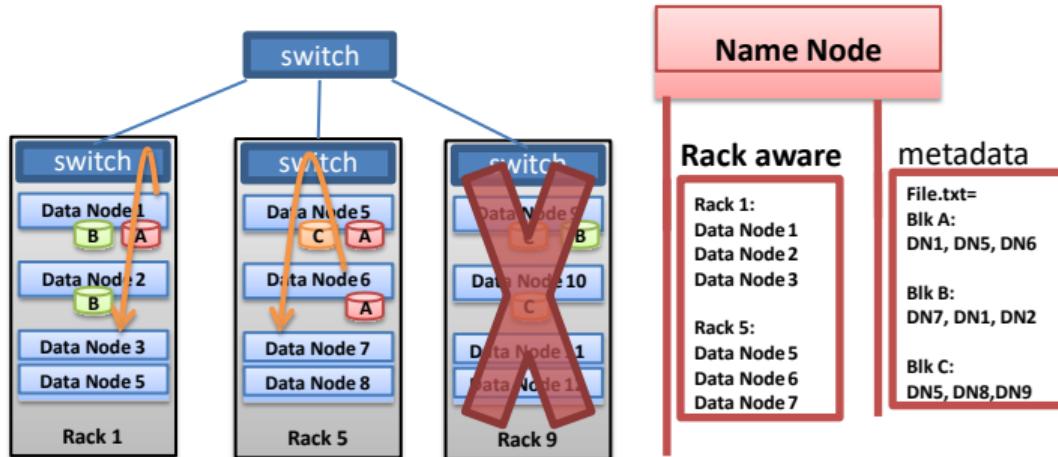


Writing files to HDFS



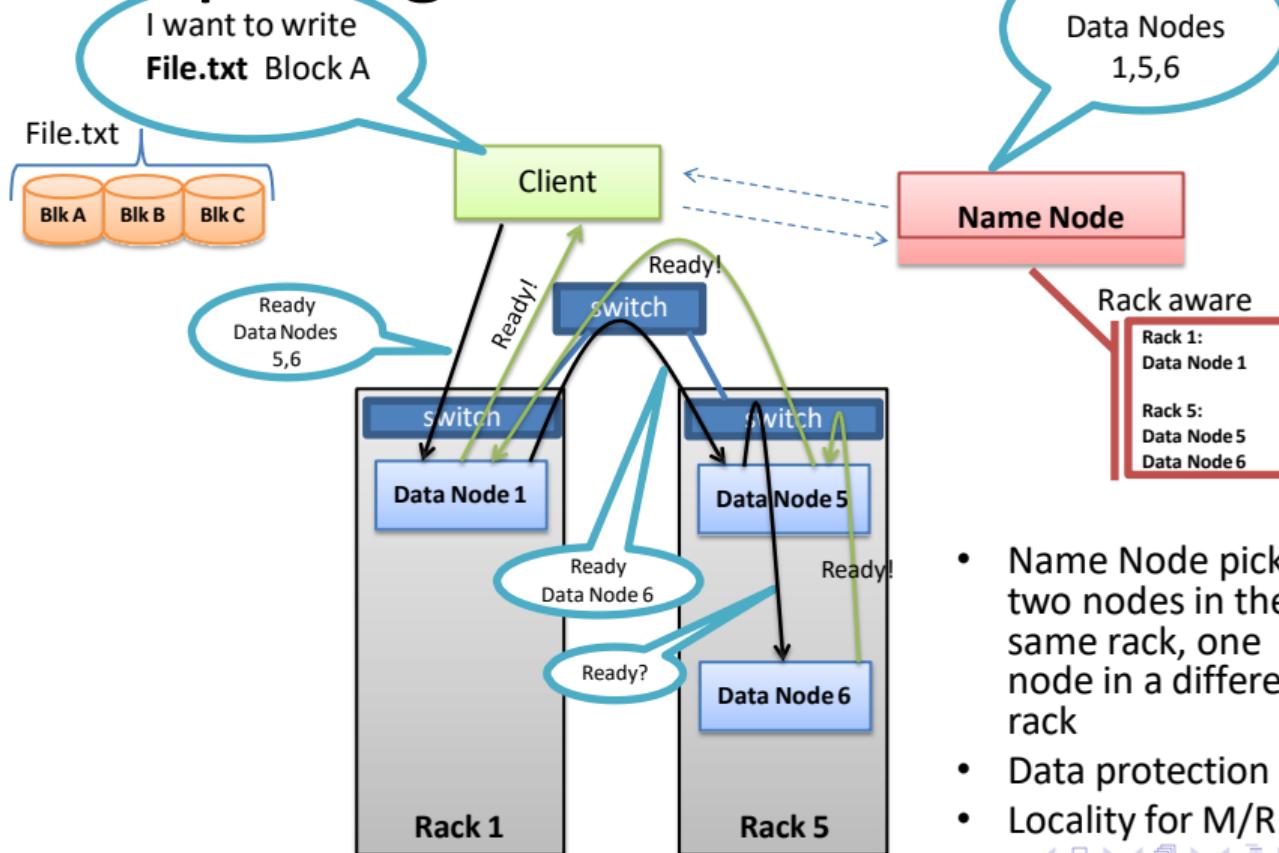
- Client consults Name Node
- Client writes block directly to one Data Node
- Data Nodes replicates block
- Cycle repeats for next block

Hadoop Rack Awareness – Why?

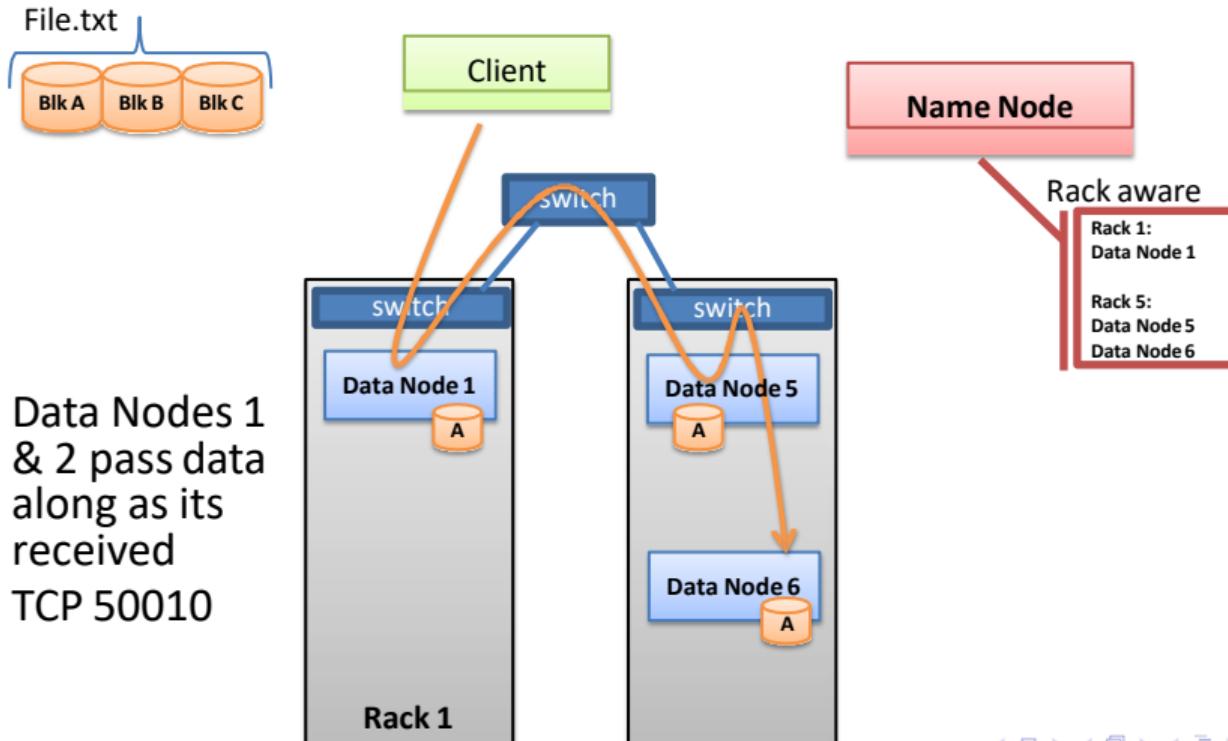


- Never lose all data if entire rack fails
- Keep bulky flows in-rack when possible
- Assumption that in-rack is higher bandwidth, lower latency

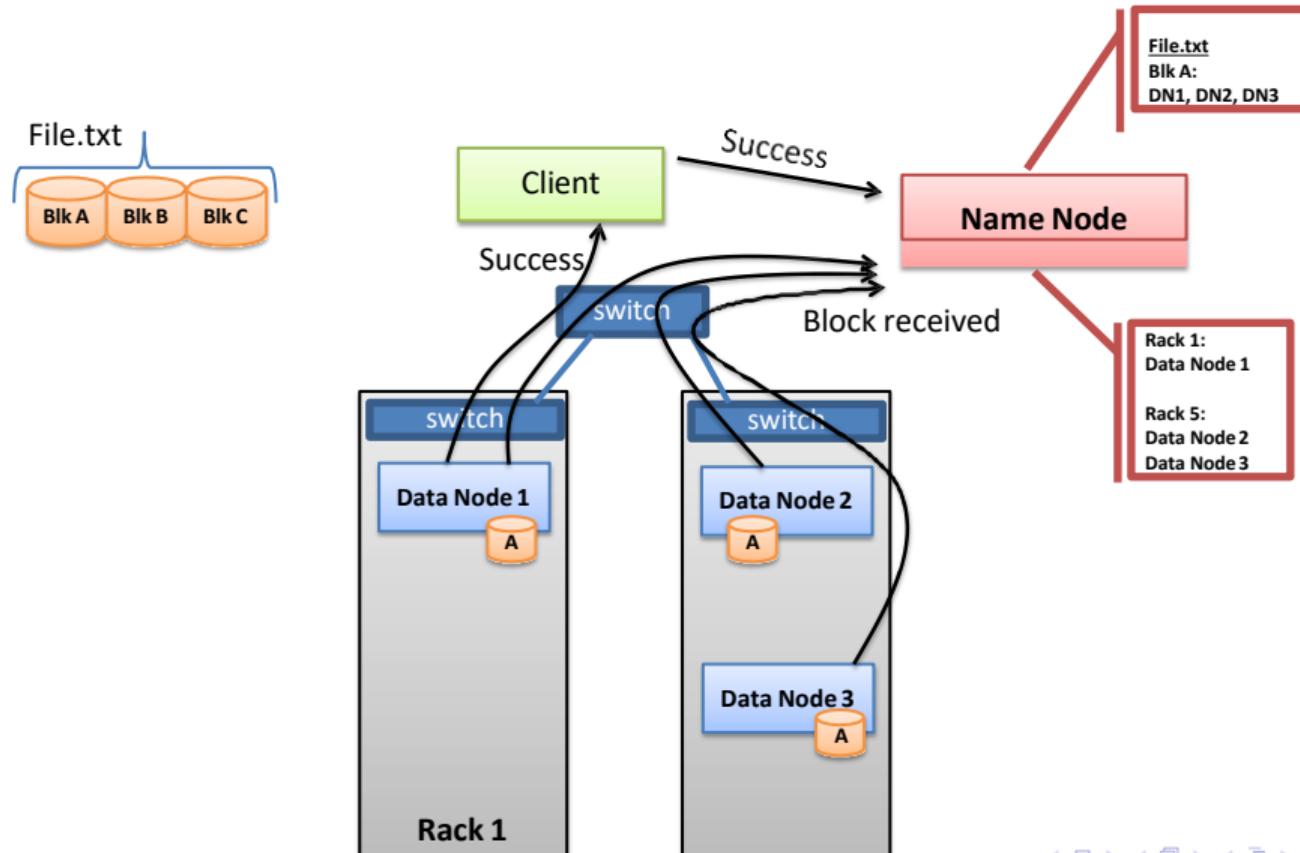
Preparing HDFS writes



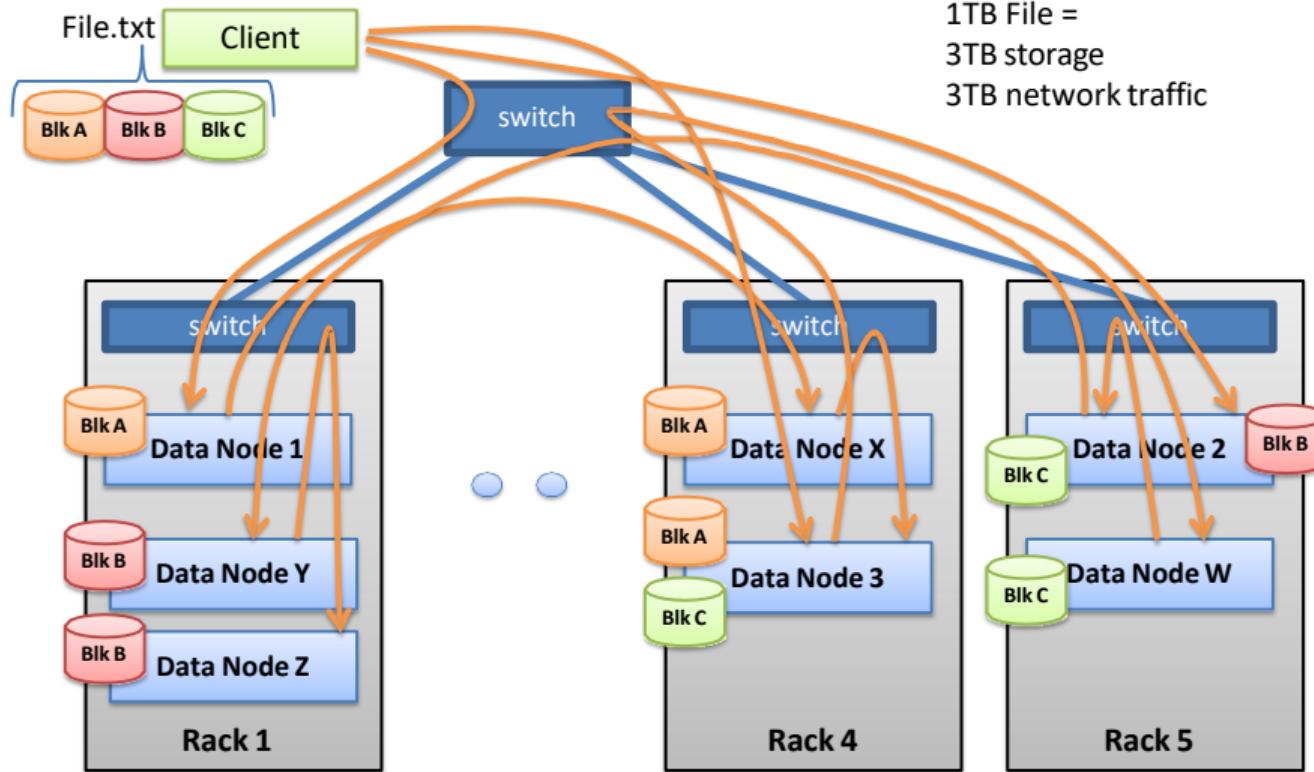
Pipelined Write



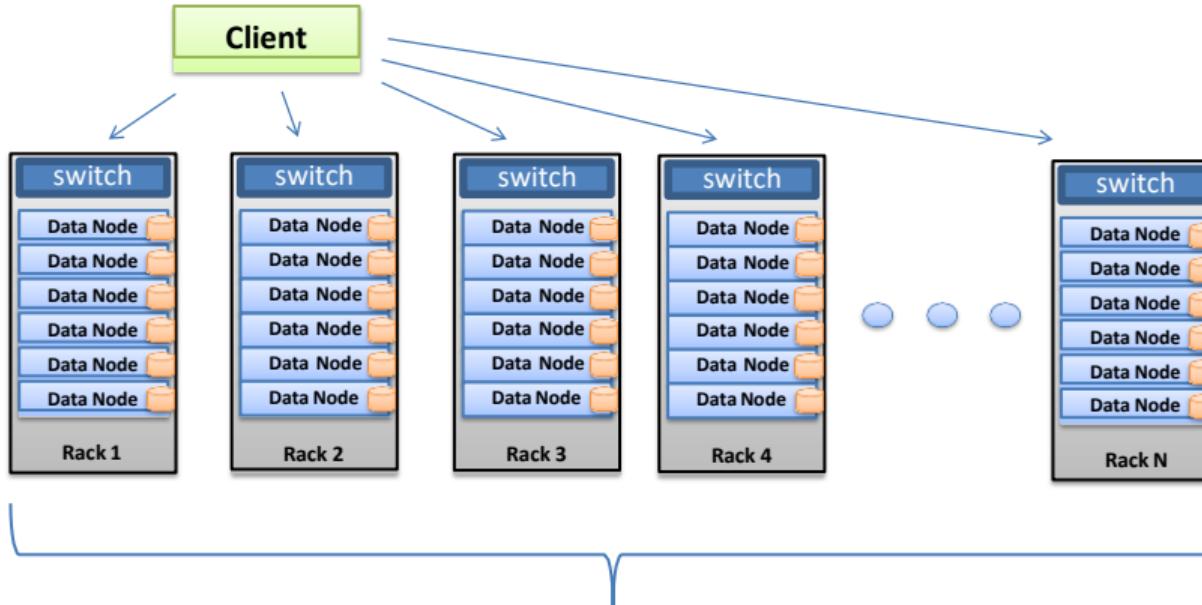
Pipelined Write



Multi-block Replication Pipeline



Client writes Span the HDFS Cluster



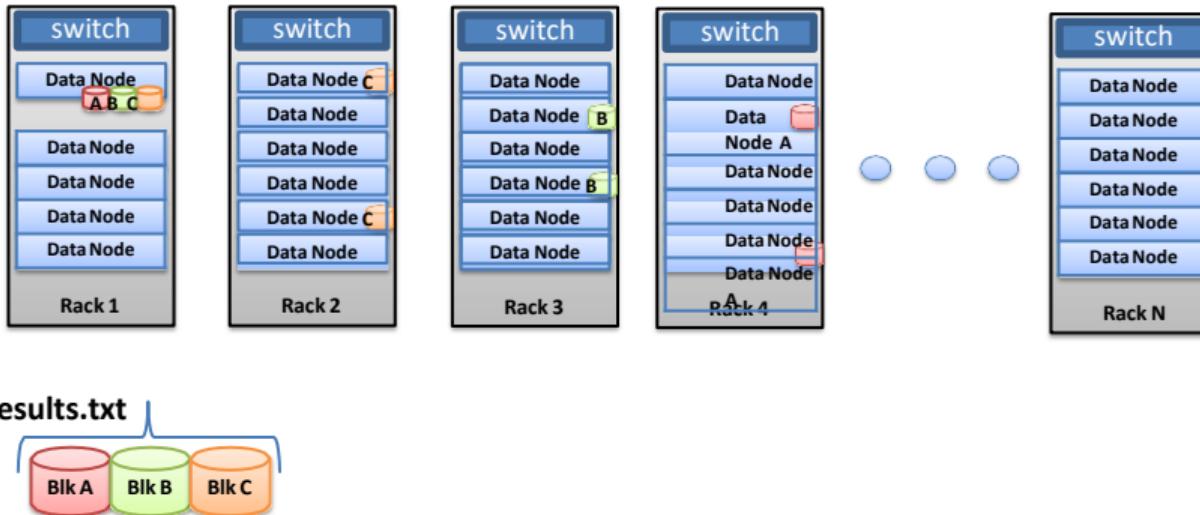
Factors:

- Block size
- File Size

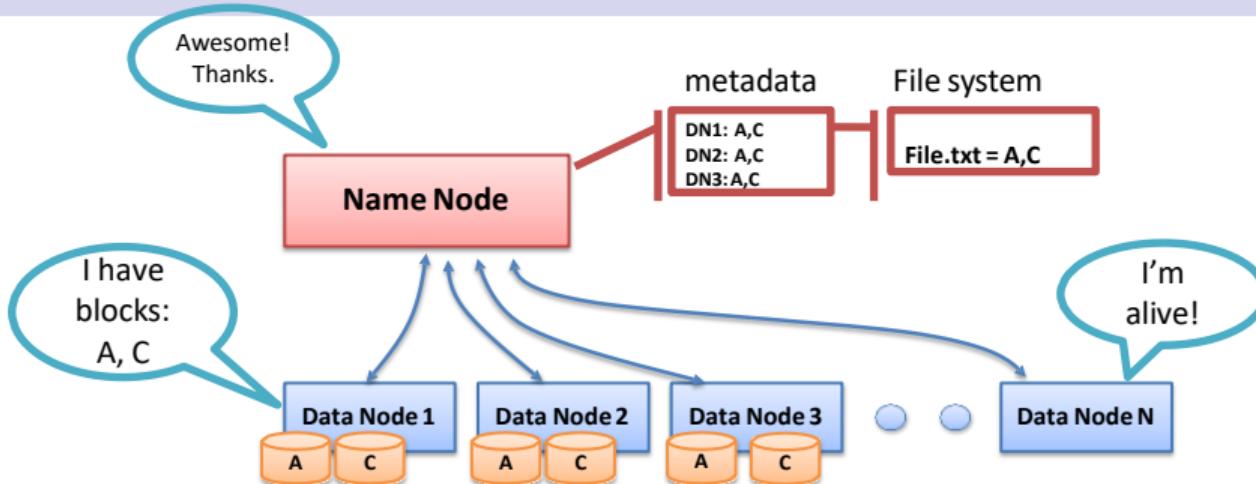
File.txt

More blocks = Wider spread

Data Node writes span itself, and other racks

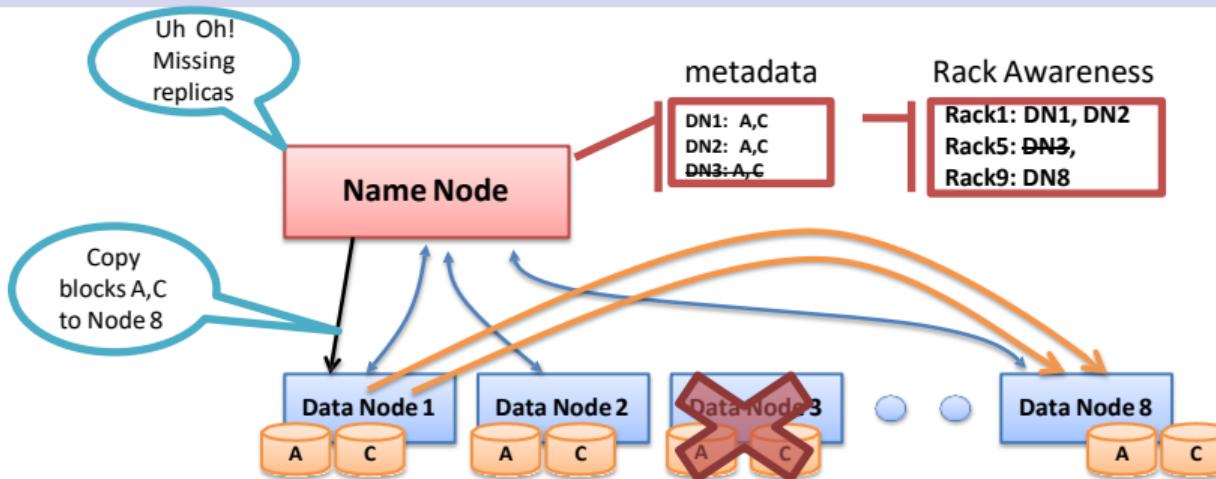


Name Node



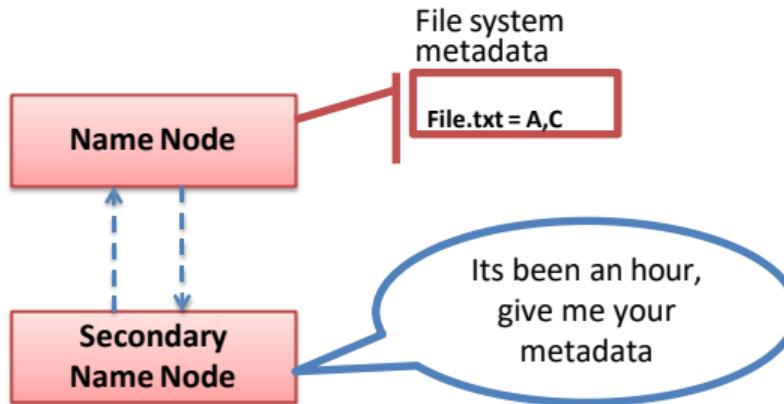
- Data Node sends Heartbeats
- Every 10th heartbeat is a Block report
- Name Node builds metadata from Block reports
- TCP – every 3 seconds
- If Name Node is down, HDFS is down

Re-replicating missing replicas



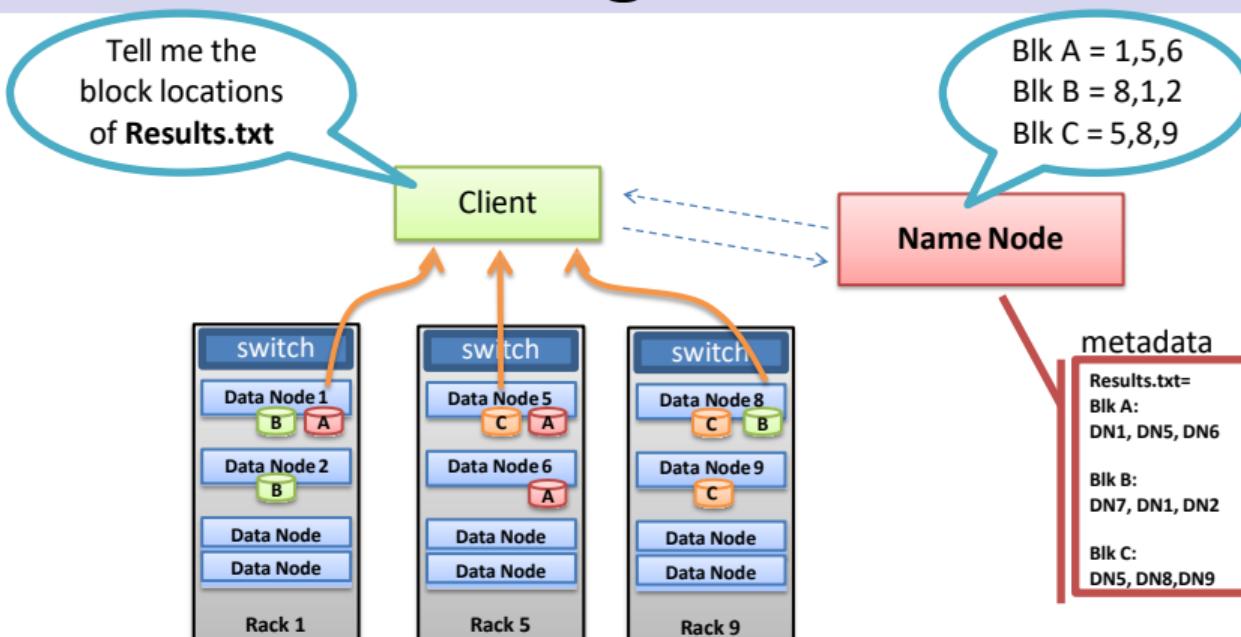
- Missing Heartbeats signify lost Nodes
- Name Node consults metadata, finds affected data
- Name Node consults Rack Awareness script
- Name Node tells a Data Node to re-replicate

Secondary Name Node



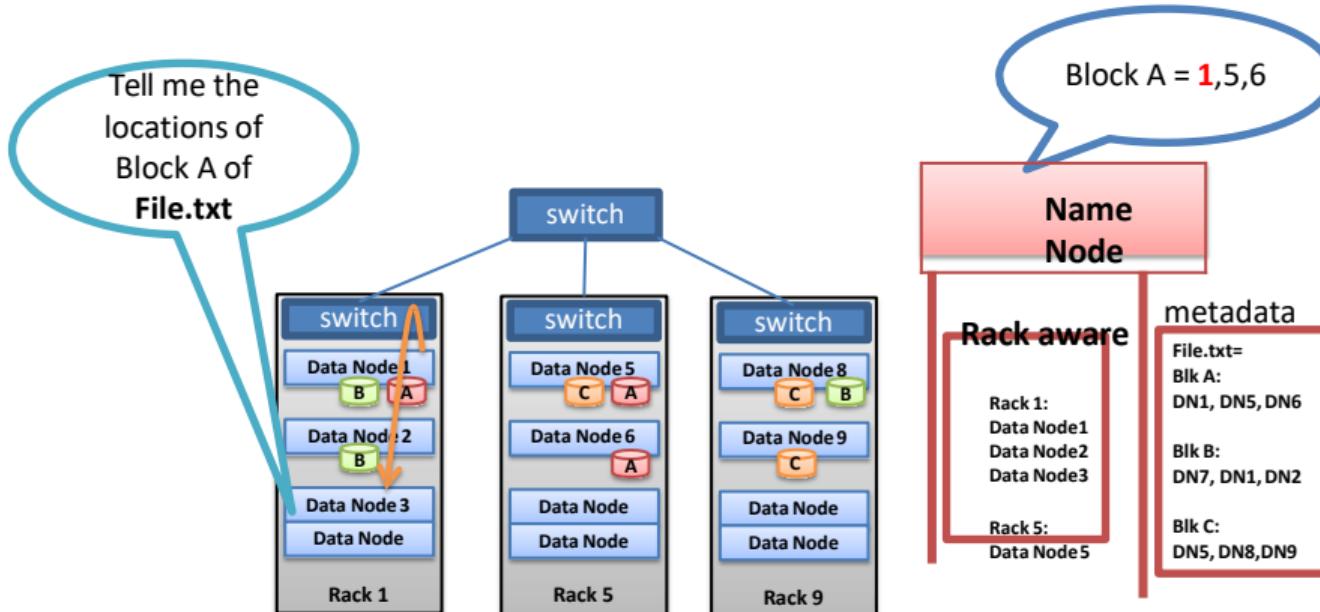
- Not a hot standby for the Name Node
- Connects to Name Node every hour*
- Housekeeping, backup of Name Node metadata
- Saved metadata can rebuild a failed Name Node

Client reading files from HDFS



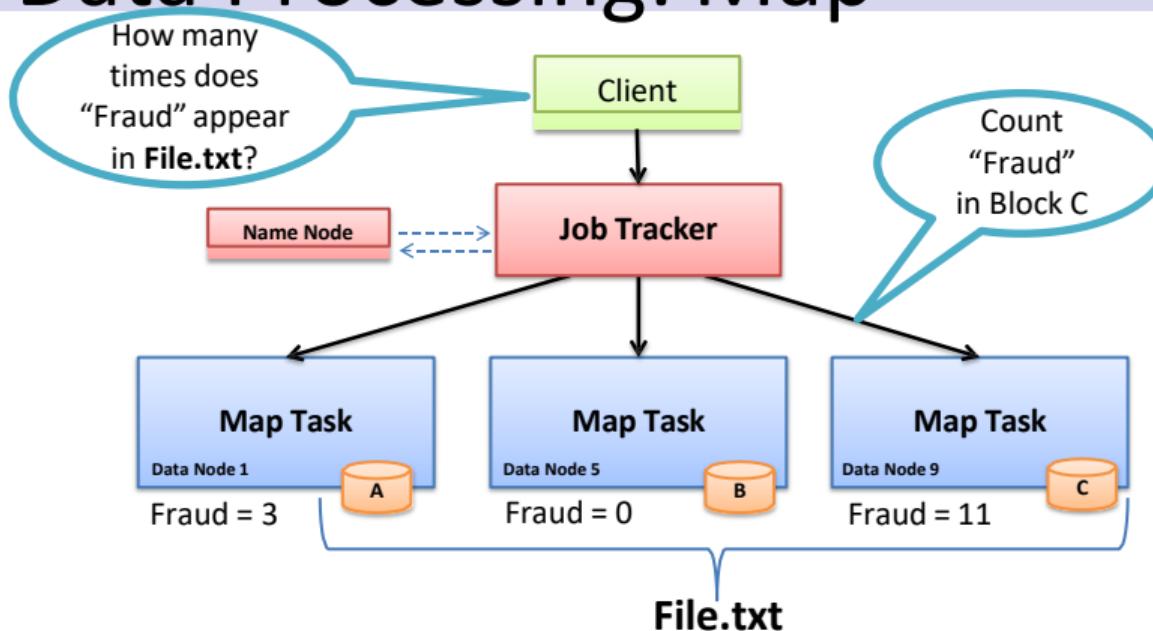
- Client receives Data Node list for each block
- Client picks first Data Node for each block
- Client reads blocks sequentially

Data Node reading files from HDFS



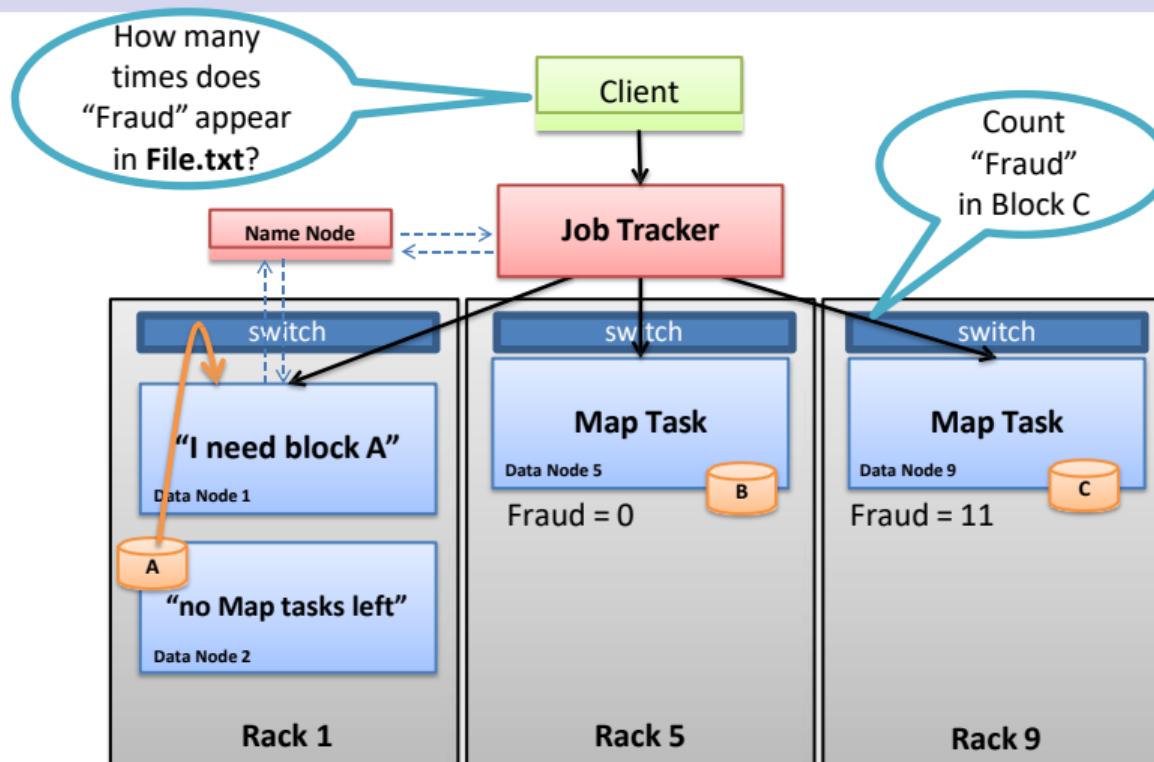
- Name Node provides rack local Nodes first
 - Leverage in-rack bandwidth, single hop

Data Processing: Map



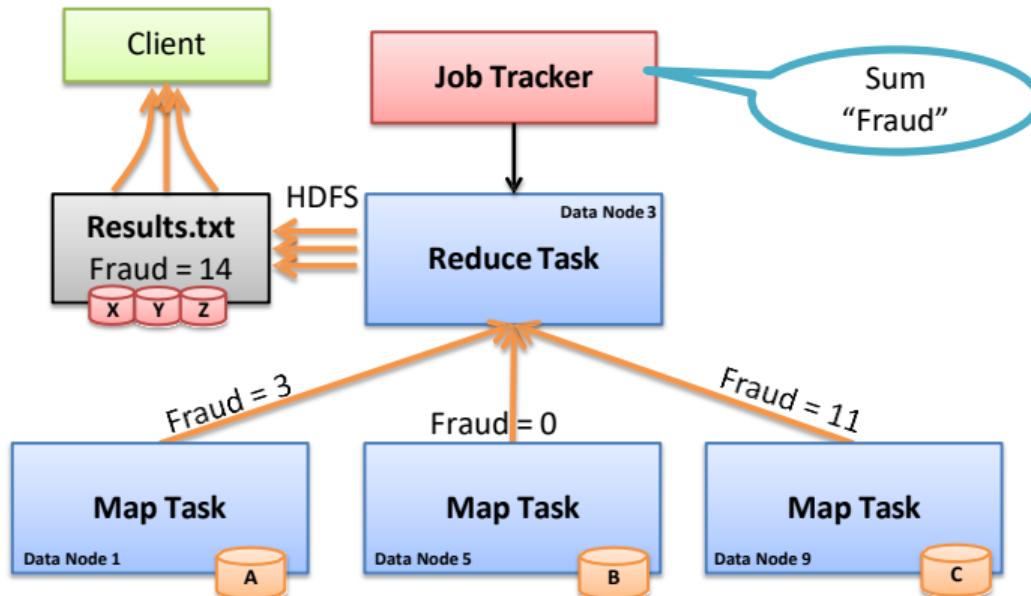
- **Map:** “Run this computation on your local data”
- Job Tracker delivers Java code to Nodes with local data

What if data isn't local?



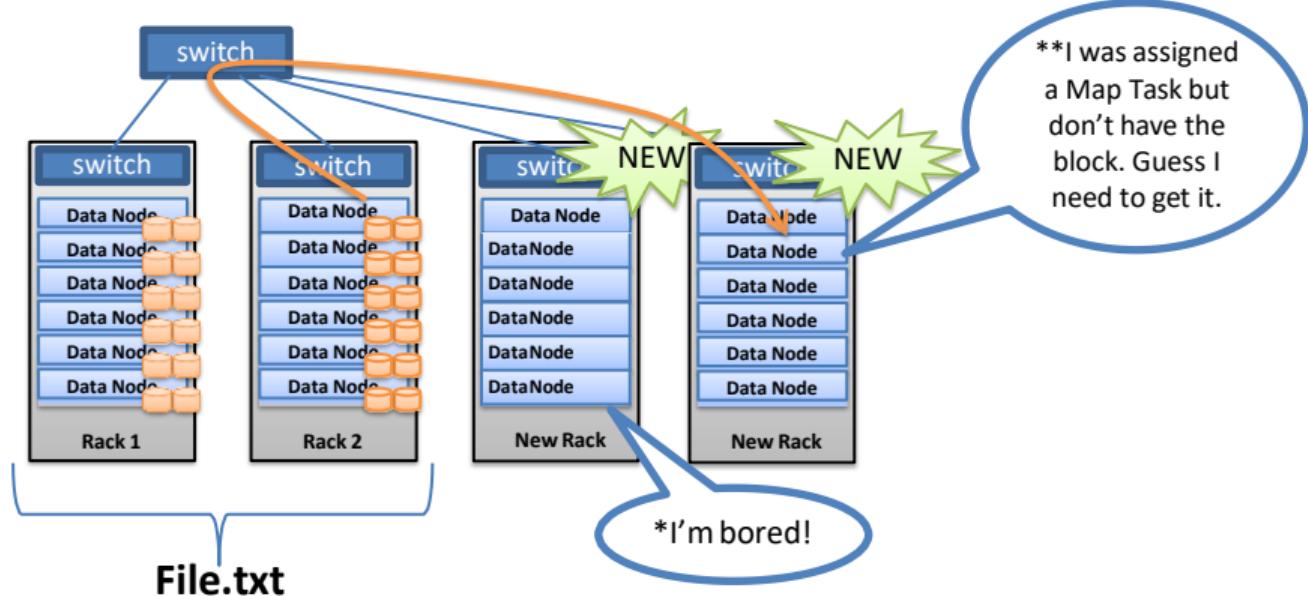
- Job Tracker tries to select Node in same rack as data
- Name Node rack awareness

Data Processing: Reduce



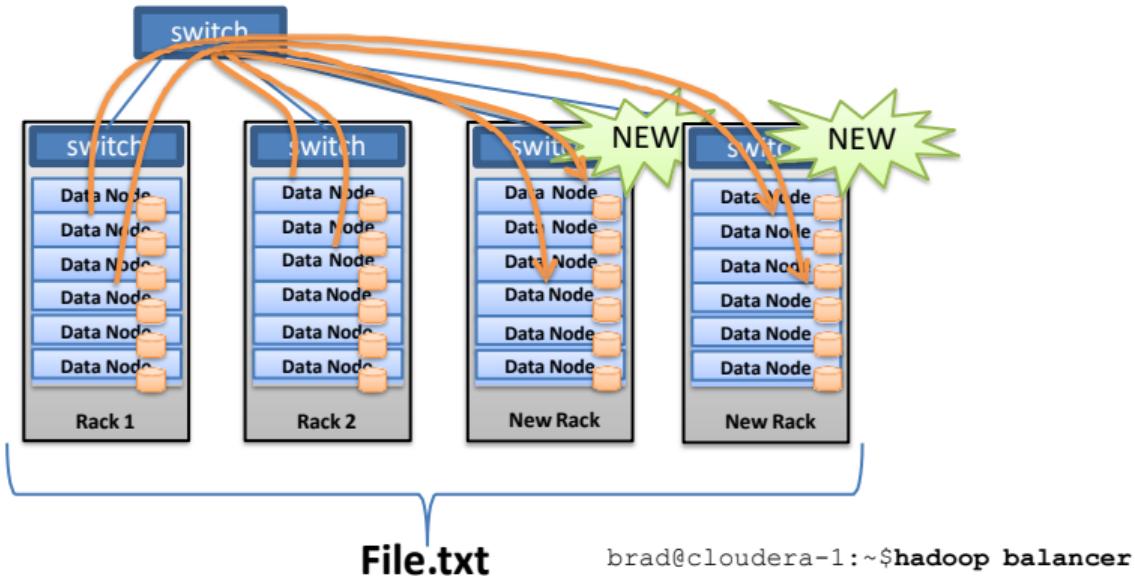
- **Reduce:** “Run this computation across Map results”
- Map Tasks deliver output data over the network
- Reduce Task data output written to and read from HDFS

Unbalanced Cluster



- Hadoop prefers local processing if possible
- New servers underutilized for Map Reduce, HDFS*
- Might see more network bandwidth, slower job times**

Cluster Balancing

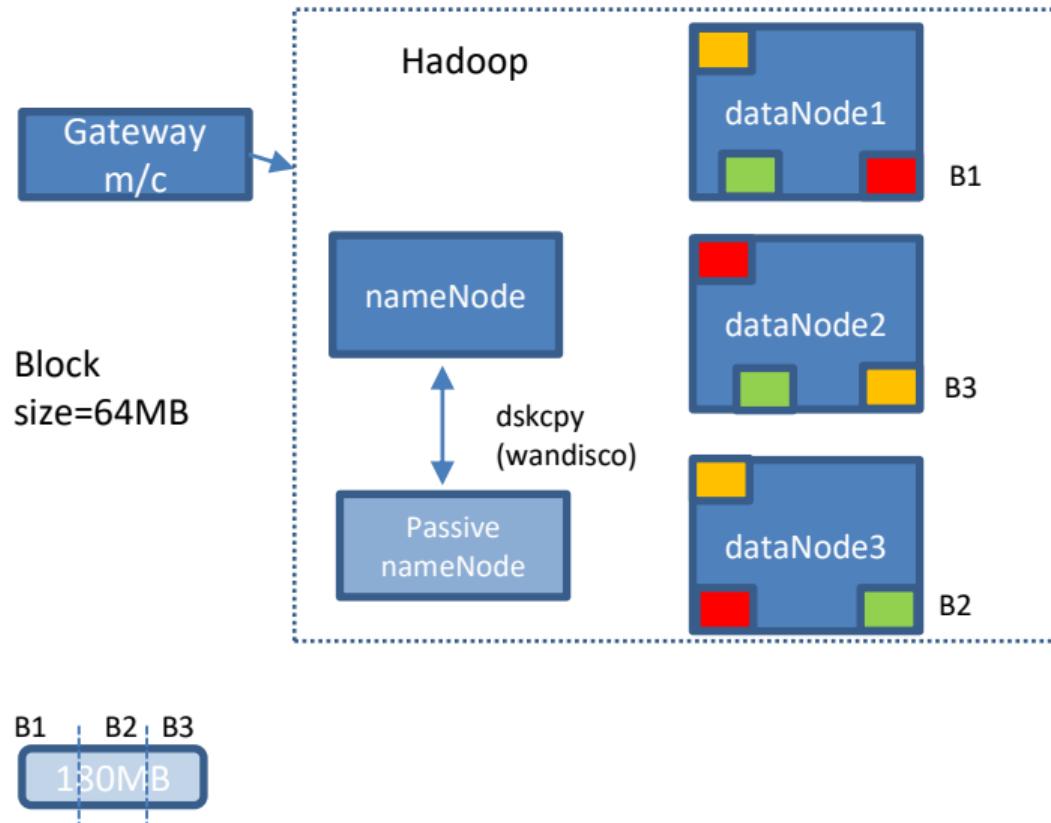


- Balancer utility (if used) runs in the background
- Does not interfere with Map Reduce or HDFS
- Default speed limit 1 MB/s

Hadoop summary

- Distributed, with some centralization
- Main nodes of cluster are where most of the computational power and storage of the system lies
- Main nodes run TaskTracker to accept and reply to MapReduce tasks, and also DataNode to store needed blocks closely as possible
- Central control node runs NameNode to keep track of HDFS directories & files, and JobTracker to dispatch compute tasks to TaskTracker
- Written in Java, also supports Python and Ruby

HDFS Storage



Default replication = 3X
Debate data redundancy
Variable replication possible
So on Hadoop version 3 no replication it uses eraser encoding

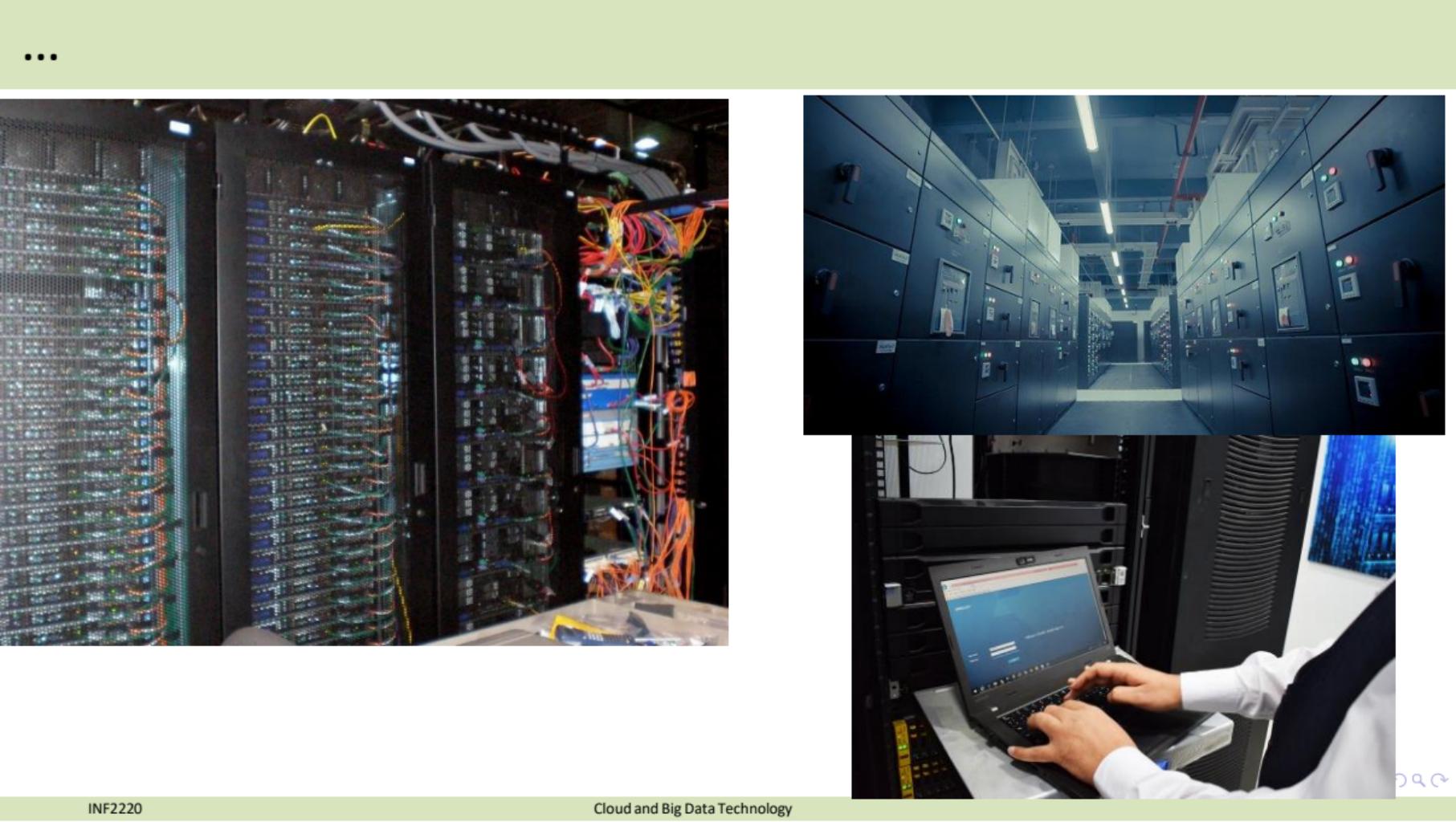
Oozie and HDFS processing

“Apache Oozie is a server-based workflow scheduling system to manage Hadoop jobs.”

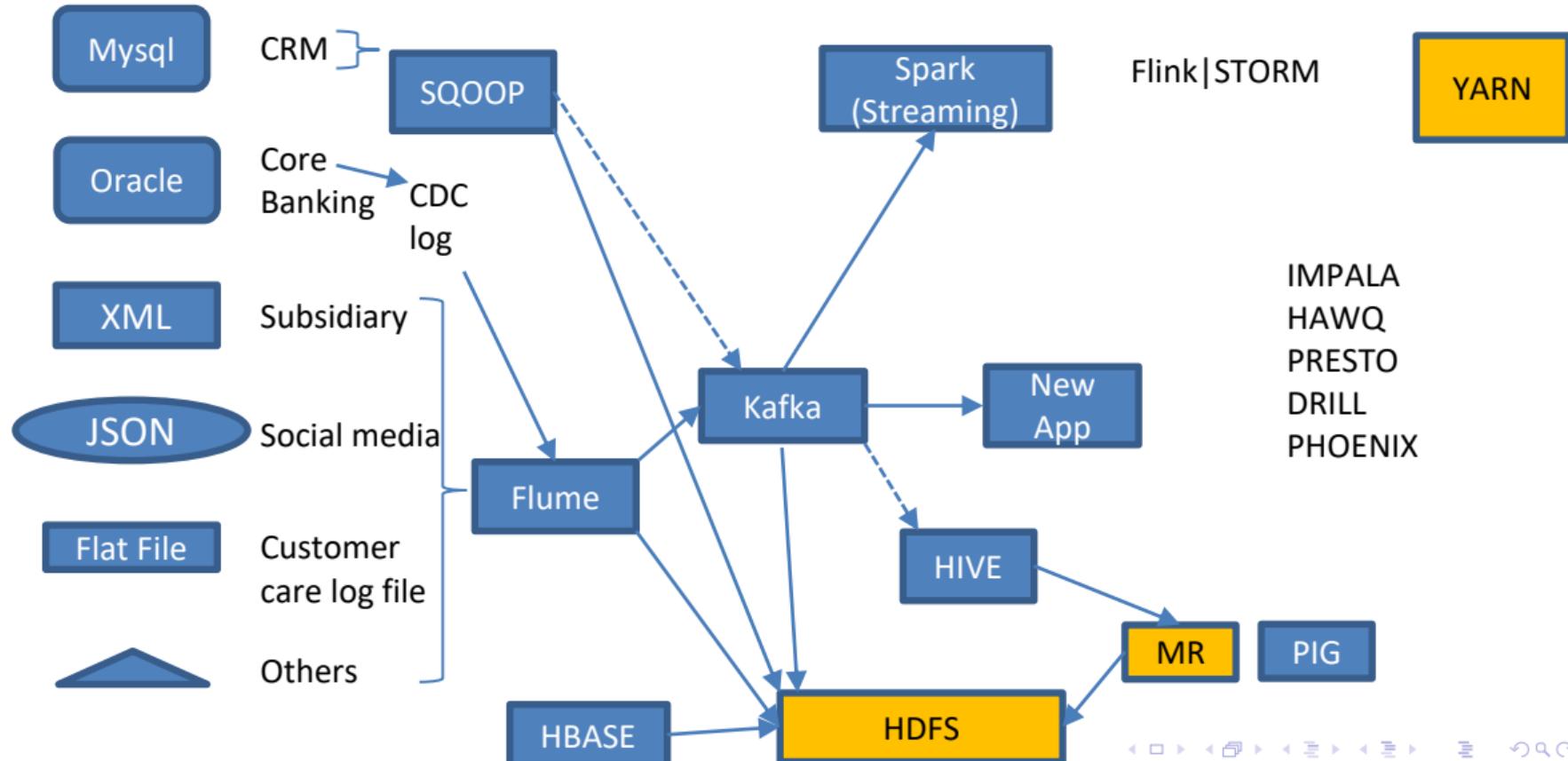
Oozie provides support for different types of actions including Hadoop MapReduce, Hadoop distributed file system operations, Pig, SSH, and email.

Others actions as well can be programmed

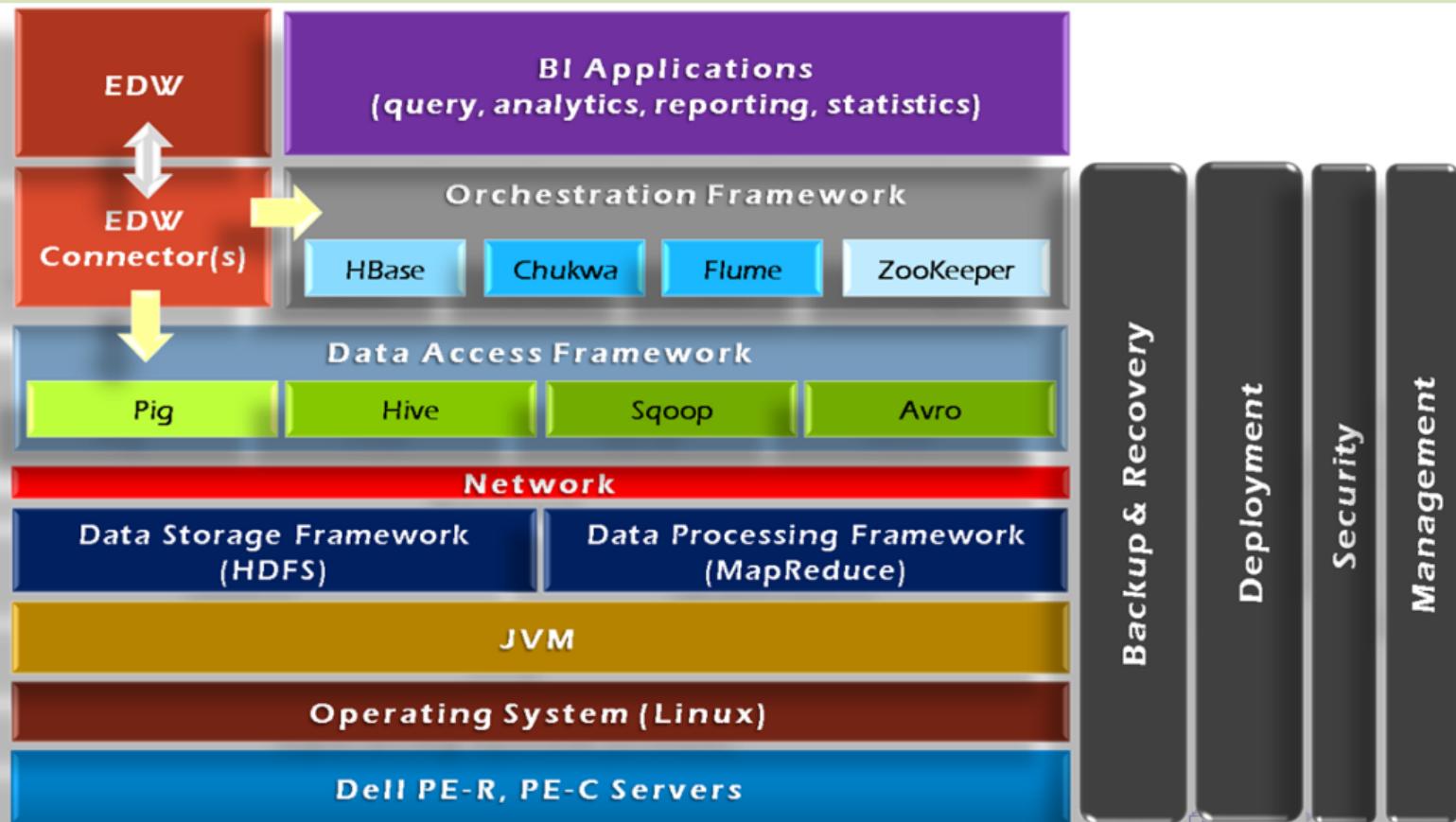
Queue based scheduling (FIFO default)



Hadoop Ecosystem



Hadoop Ecosystem



Map Reduce Fundamentals

Dilip K. Prasad
INF 2220

Outline

MapReduce: Overview

Understanding MapReduce with an example

MapReduce Practical Example

Word Count Program Code – python

YARN (Hadoop) - resource manager

MapReduce

If a problem becomes large what we do?

Divide & Conquer

MapReduce is a framework for processing **parallelizable** problems

What programming language
can be used for MapReduce
framework?

MapReduce

MapReduce is a framework for processing **parallelizable** problems

Data to program → Single node

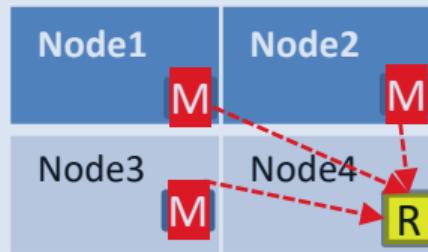
Large datasets ->

Large number of
processing node
locally or across
distributed
network

Program to data --> MapReduce

Oslo
...	Finnmark	
...	...	Troms

MapReduce



Mapper

Reducer

Default:

Mapper – Many/one

Reducer – One (by default)

YARN – Resource manager

Typical problem to be solved by MapReduce

- Read a lot of data
- **Map**: extract something you care about from each record
- Shuffle and Sort
- **Reduce**: aggregate, summarize, filter, or transform
- Write the results

Understanding MapReduce with an example

Word Count example

(key,value)

M1

Arya,Sansa,John
Arya,Sansa,John
Arya,Sansa,John
Arya,Sansa,John
Arya,Sansa,John



(Arya,1)(Sansa,1) (John,1)
(Arya,1)(Sansa,1) (John,1)
(Arya,1)(Sansa,1) (John,1)
(Arya,1)(Sansa,1) (John,1)
(Arya,1)(Sansa,1) (John,1)

Why mapReduce
is slow?

M2

Arya,Sansa,John
Arya,Sansa,John
Arya,Sansa,John



(Arya,1)(Sansa,1) (John,1)
(Arya,1)(Sansa,1) (John,1)
(Arya,1)(Sansa,1) (John,1)

mapReduce is
fault tolerant

M3

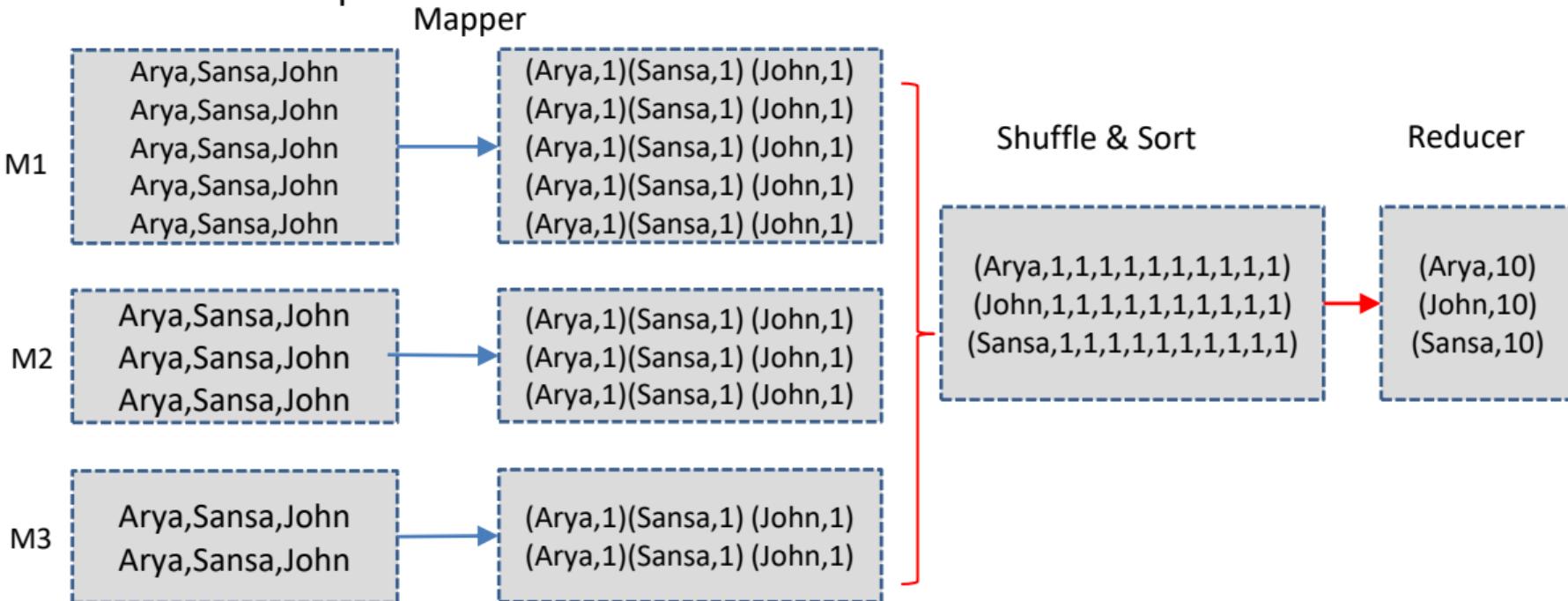
Arya,Sansa,John
Arya,Sansa,John



(Arya,1)(Sansa,1) (John,1)
(Arya,1)(Sansa,1) (John,1)

Understanding MapReduce with an example

Word Count example



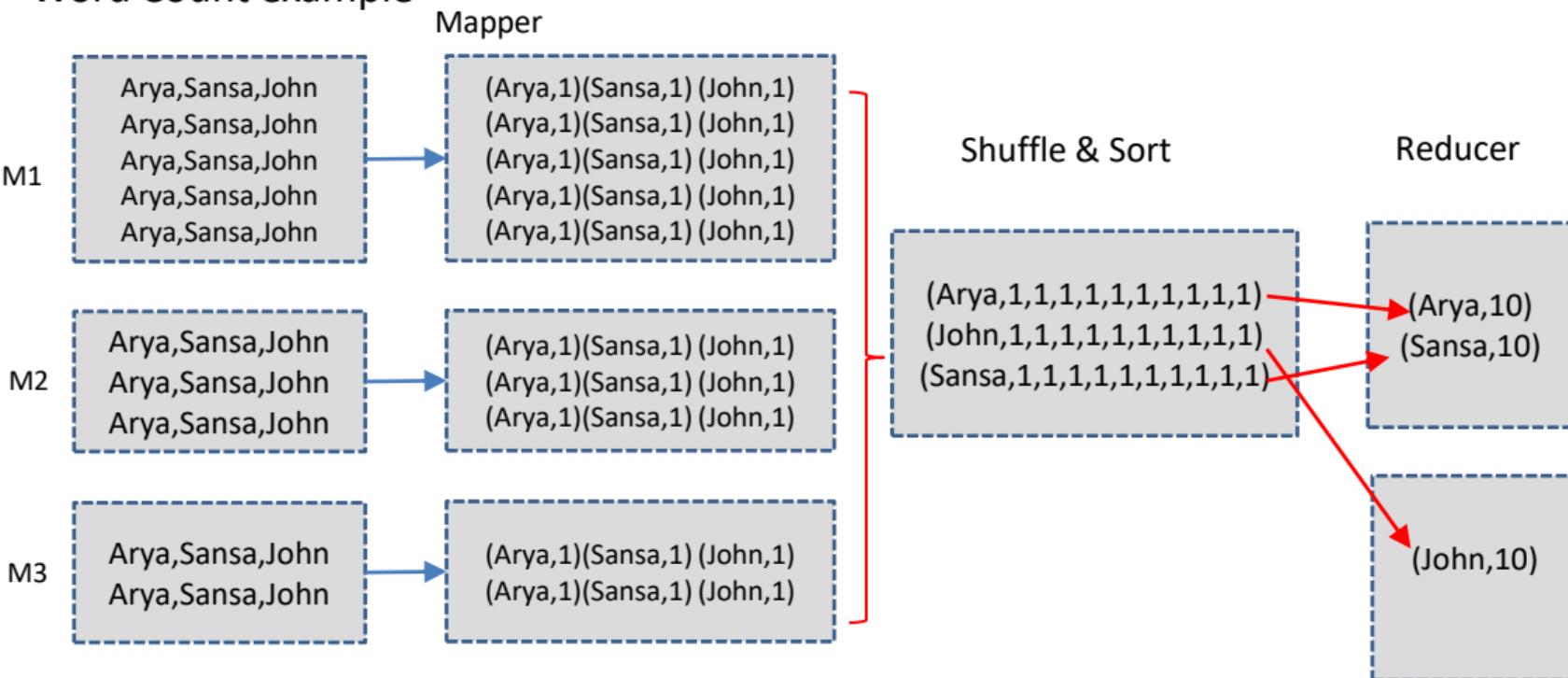
??

Can we have more than one reducer?

How to decide how many reducer?

Understanding MapReduce with an example

Word Count example



Mapper vs reducer balance

Hadoop MapReduce on 1.1GB file multiple times with a different number of mappers and reducers (e.g. 1 mapper and 1 reducer, 1 mapper and 2 reducers, 1 mapper and 4 reducers, ...)
Hadoop is installed on quad-core machine with hyper-threading

time	# of map	# of red
7m 50s	8	2
8m 13s	8	4
8m 16s	8	8
8m 28s	4	8
8m 37s	4	4

MapReduce Practical Example

Oslo - 5 million/month

Narvik – 10,000/month

Bergen – 1 million/month

Tromsø– 20000/month



Word Count mapReduce pseudocode

- Input: a set of key/value pairs
- User supplies two functions:
 - $\text{map}(k, v) \rightarrow \text{list}(k_1, v_1)$
 - $\text{reduce}(k_1, \text{list}(v_1)) \rightarrow v_2$
- (k_1, v_1) is an intermediate key/value pair
- Output is the set of (k_1, v_2) pairs

Word Count mapReduce pseudocode

```
map(key, value):
// key: document name; value: text of document
    for each word w in value:
        return(w, 1)
```

```
reduce(key, values):
// key: a word; value: an iterator over counts
    result = 0
    for each count v in values:
        result += v
    return(result)
```

Mapper in python

```
#!/usr/bin/env python
import sys

#--- get all lines from stdin ---
for line in sys.stdin:
    #--- remove leading and trailing whitespace---
    line = line.strip()

    #--- split the line into words ---
    words = line.split()

    #--- output tuples [word, 1] in tab-delimited format---
    for word in words:
        print '%s\t%s' % (word, "1")
```

Reducer code in python

```
#!/usr/bin/env python
import sys

# maps words to their counts
word2count = {}

# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    word, count = line.split('\t', 1)
    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        continue

    # try:
    #     word2count[word] = word2count[word]+count
    # except:
    #     word2count[word] = count

    # write the tuples to stdout
    # Note: they are unsorted
    for word in word2count.keys():
        print '%s\t%s' % (word, word2count[word])
```

try:

 word2count[word] = word2count[word]+count

except:

 word2count[word] = count

write the tuples to stdout

Note: they are unsorted

for word in word2count.keys():

 print '%s\t%s' % (word, word2count[word])

Data Flow

- Input and final output are stored on a distributed file system (FS):
- Scheduler tries to schedule map tasks “close” to physical storage location of input data
- Intermediate results are stored on local FS of Map and Reduce workers
- Output is often input to another MapReduce task

Coordination Master

- Master node takes care of coordination:
- **Task status:** (idle, in-progress, completed)
- Idle tasks get scheduled as workers become available
- When a map task completes, it sends the master the location and sizes of its R intermediate files, one for each reducer
- Master pushes this info to reducers
- Master pings workers periodically to detect failures

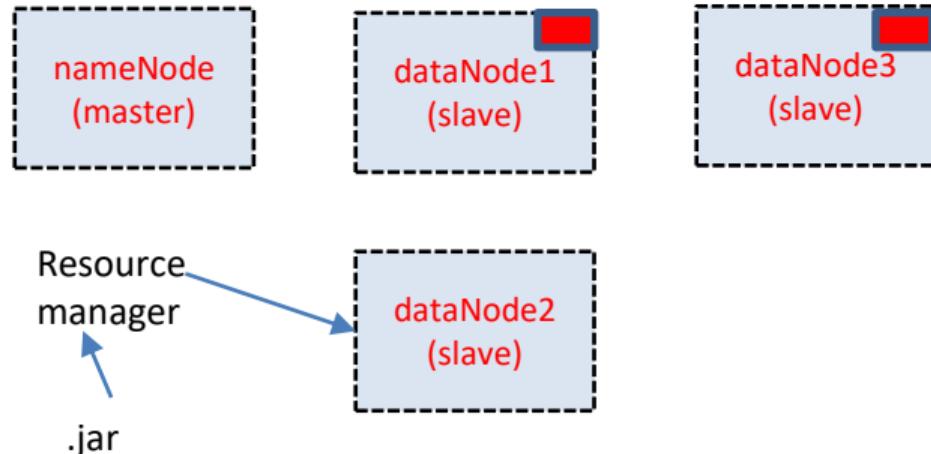
Dealing with the Failure

- Map worker failure
- Map tasks completed or in-progress at worker are reset to idle
- Reduce workers are notified when task is rescheduled on another worker
- Reduce worker failure
- Only in-progress tasks are reset to idle
- Reduce task is restarted
- Master failure
- MapReduce task is aborted and client is notified

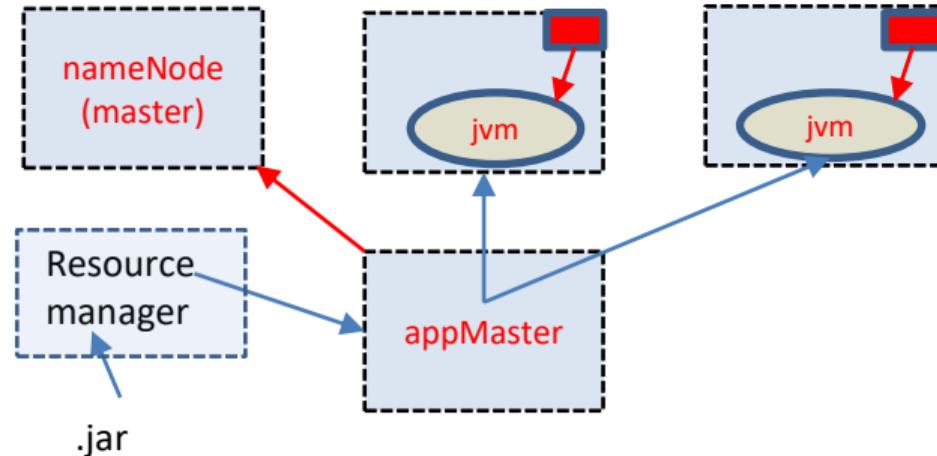
How many Map and Reduce jobs?

- M map tasks, R reduce tasks
- Rule of a thumb:
- *Make M much larger than the number of nodes in the cluster*
- One DFS chunk per map is common
- Improves dynamic load balancing and speeds up recovery from worker failures
- *Usually R is smaller than M*
- Because output is spread across R files

YARN (Hadoop)



YARN (Hadoop)



After completing the mapper and reducer task Jvms and appMaster will be deleted

Security in Hadoop

KerberOS (works on the basis of tickets to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner)

Sentry (fine grained role based authorization to both data and metadata stored on an Apache Hadoop cluster)

Reference books

Michael Noll, Writing an Hadoop MapReduce Program in Python.

Hadoop, the definitive guide, Tim White, O'Reilly Media, 2009.

Apache Spark Fundamentals

Dilip K. Prasad

Outline

Introduction

Spark Fundamentals

Spark Architecture

Spark and it's Ecosystem

Spark vs Hadoop

RDD Fundamentals

Spark Transformations, Actions and Operations

Introduction

2009 AMPLab (Berkley) – resource/cluster manager

2009 No YARN
 MESOS

Created framework to test Mesos – Spark was created (which is in memory execution) – huge RAM requirement

2010/11 SPARK further development started

2012 Apache Spark -> Databricks

2015 It got traction and more support system came up

2021 INF2220 Cloud and Big Data Technology started

Spark version 0

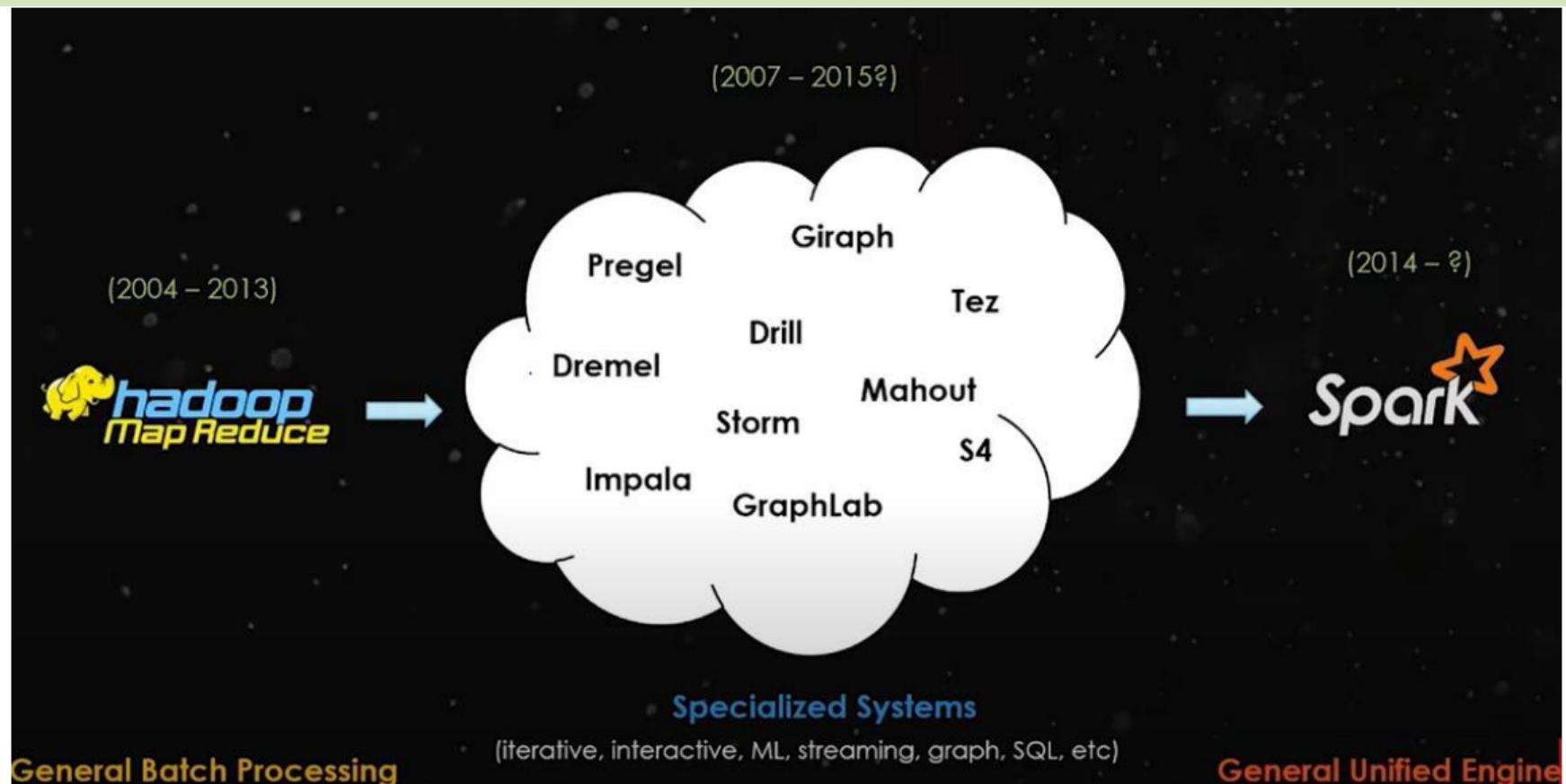
1.6

2.7

3.2.0

Best book – Spark the definitive guide – Mathe Zaharia (**don't buy** it its not text book type)

Changing Big Data World



Introduction to Spark



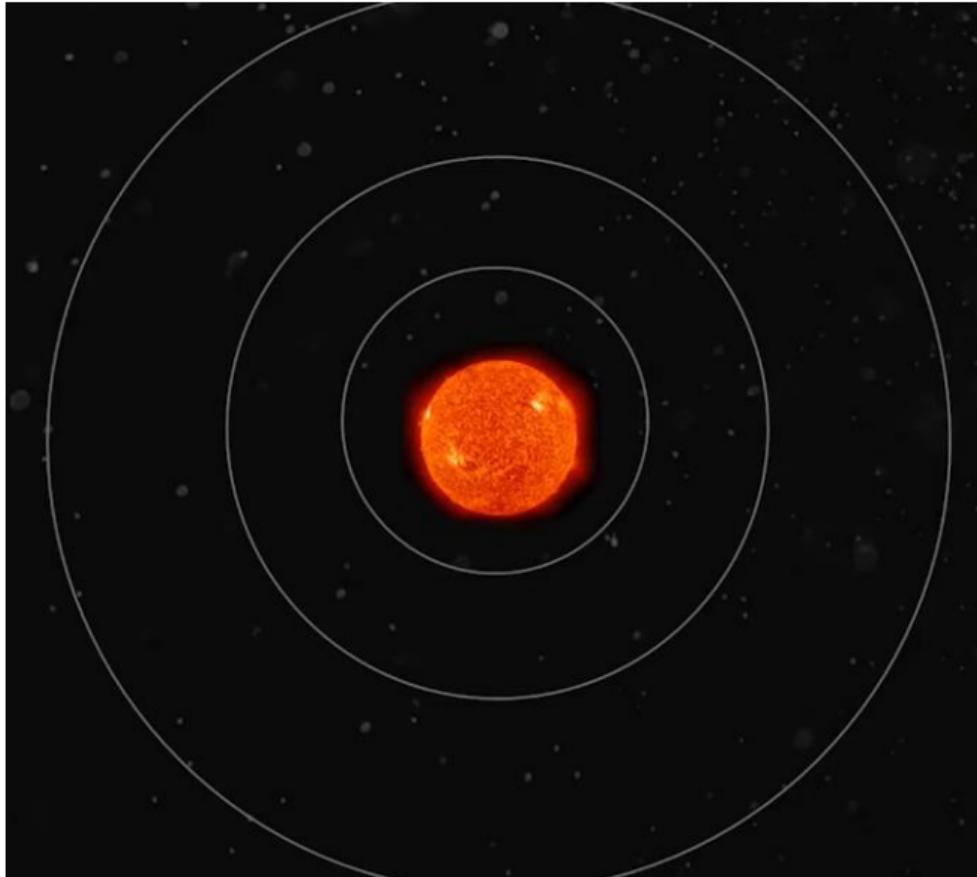
Scheduling

Monitoring

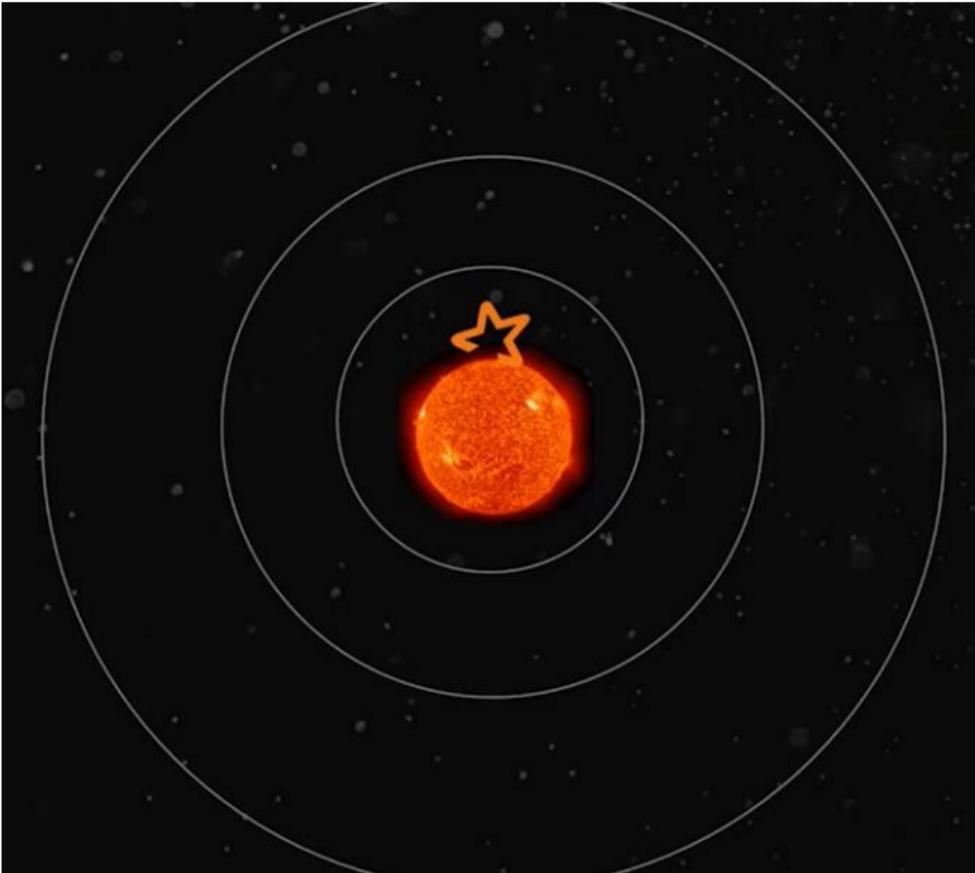


Distributing

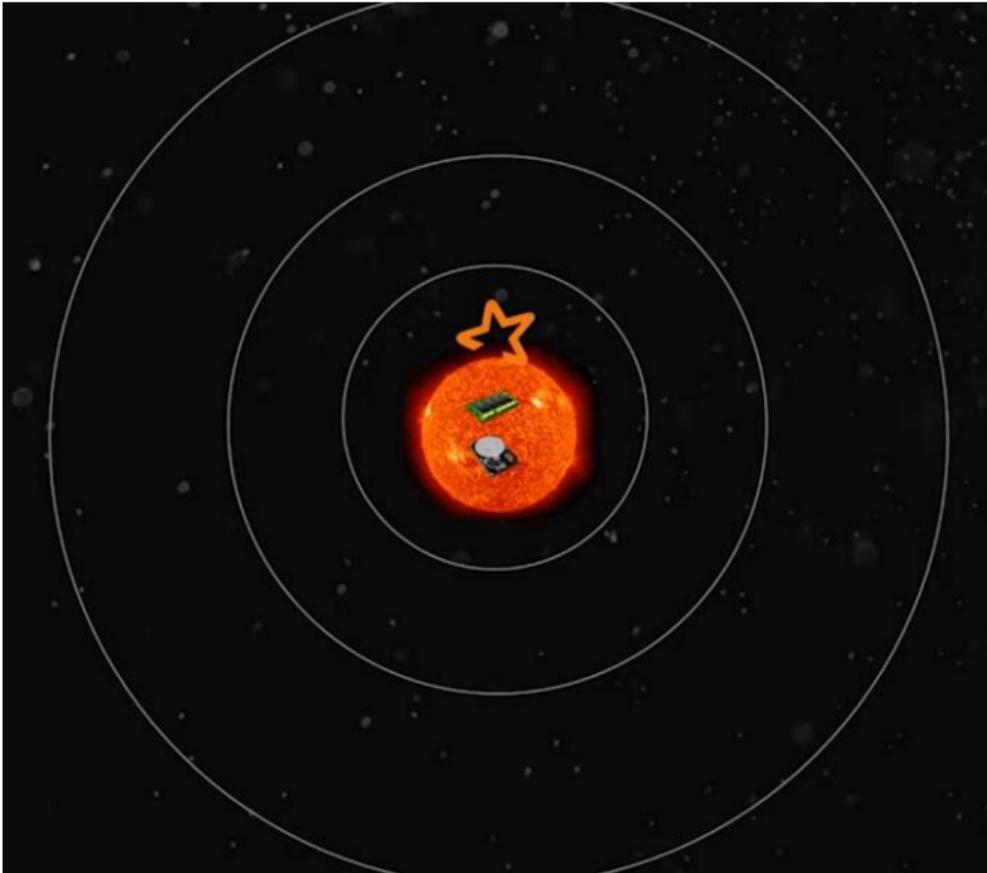
Introduction to Spark



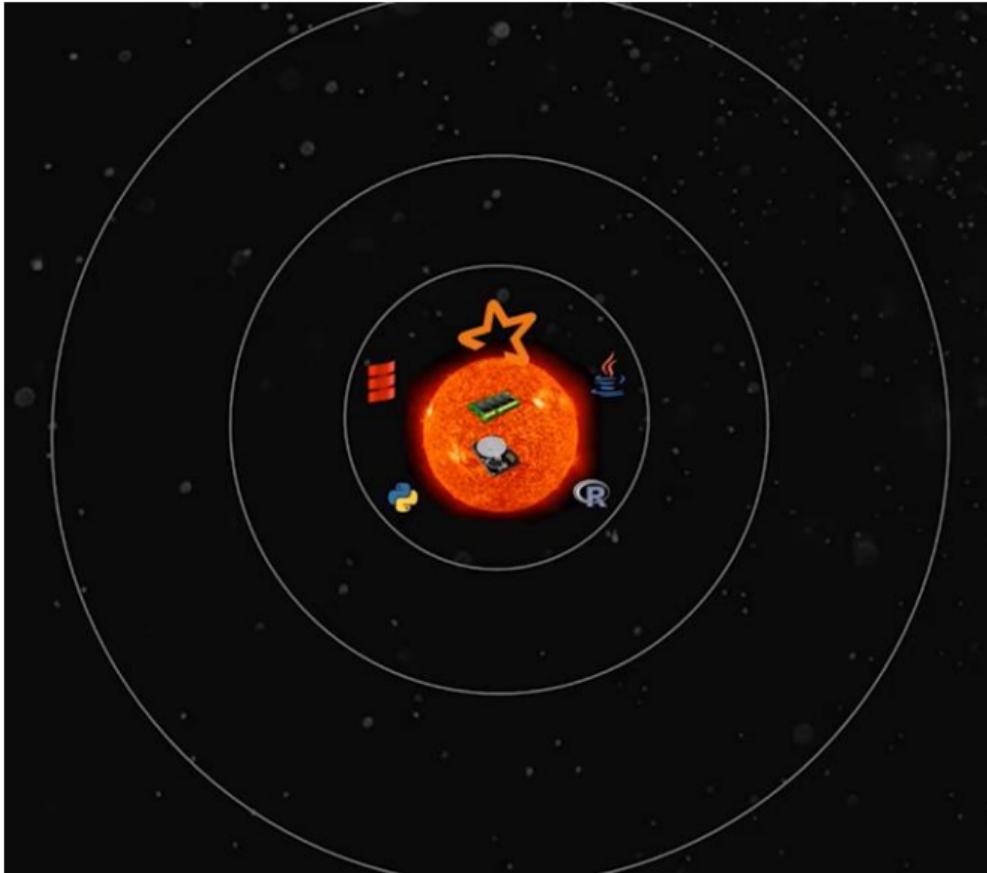
Spark Architecture & its Ecosystem



Spark Architecture & its Ecosystem



Spark Architecture & its Ecosystem



Spark Architecture & its Ecosystem



Spark Architecture & its Ecosystem



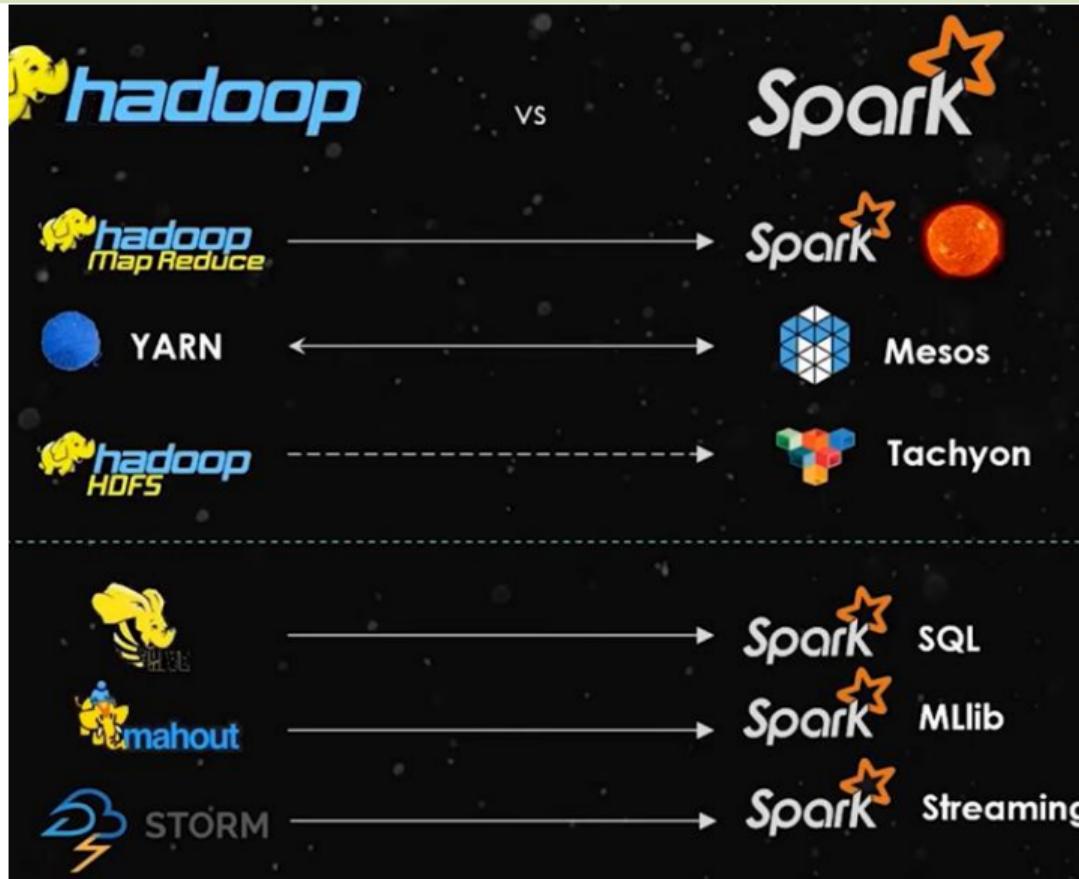
Spark Architecture & its Ecosystem



Spark Architecture & its Ecosystem



Hadoop Vs Spark



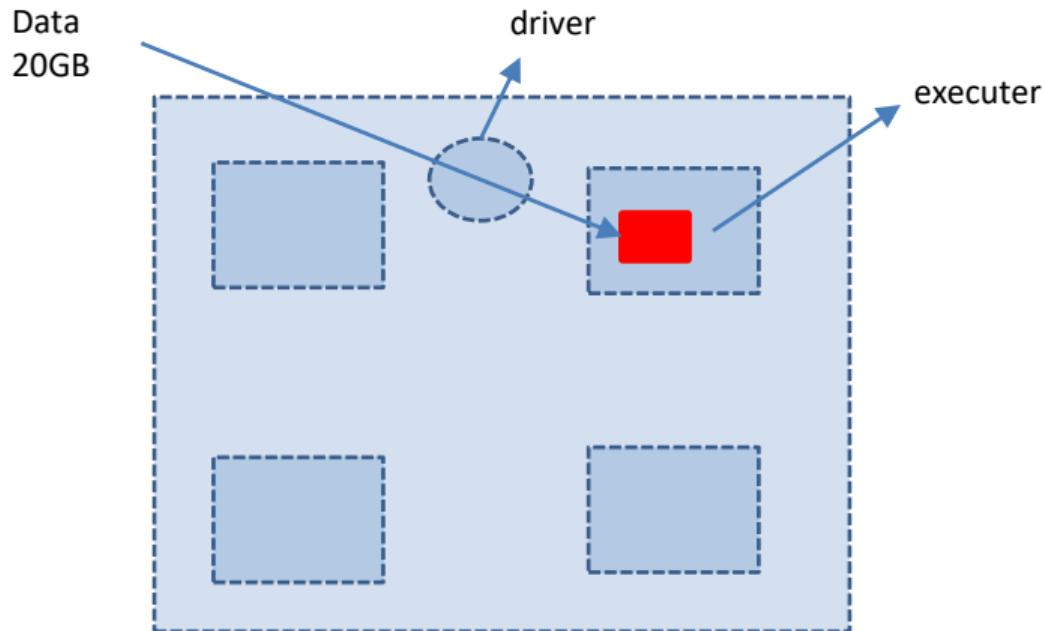
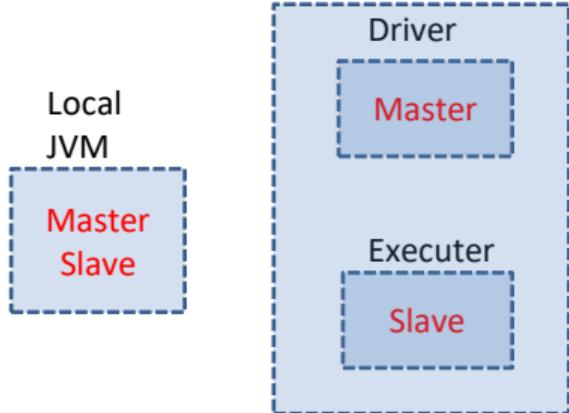
DISTRIBUTORS



APPLICATIONS



How spark program get executed



Sort Competition

	Hadoop MR Record (2013)	Spark Record (2014)	Spark, 3x faster with 1/10 the nodes
Data Size	102.5 TB	100 TB	
Elapsed Time	72 mins	23 mins	
# Nodes	2100	206	
# Cores	50400 physical	6592 virtualized	
Cluster disk throughput	3150 GB/s (est.)	618 GB/s	
Network	dedicated data center, 10Gbps	virtualized (EC2) 10Gbps network	
Sort rate	1.42 TB/min	4.27 TB/min	
Sort rate/node	0.67 GB/min	20.7 GB/min	

Sort benchmark, Daytona Gray: sort of 100 TB of data (1 trillion records)

<http://databricks.com/blog/2014/11/05/spark-officially-sets-a-new-record-in-large-scale-sorting.html>

Resilient Distributed Datasets (RDDs)

- RDDs (Resilient Distributed Datasets) is Data Containers
- All the different processing components in Spark share the same abstraction called RDD
- As applications share the RDD abstraction, you can mix different kind of transformations to create new RDDs
- Created by parallelizing a collection or reading a file
- Fault tolerant

RDD



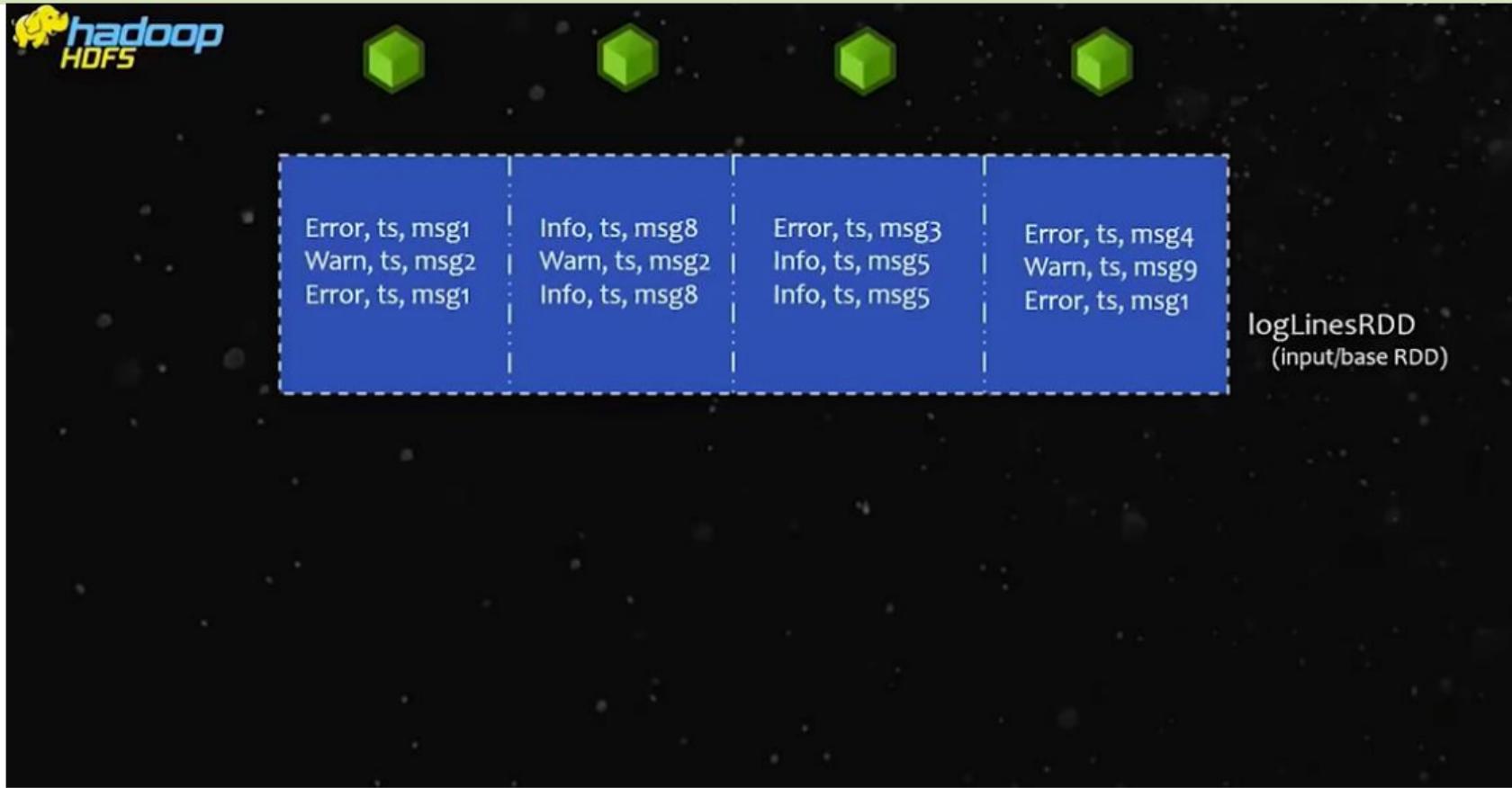
Error, ts, msg1
Warn, ts, msg2
Error, ts, msg1

Info, ts, msg8
Warn, ts, msg2
Info, ts, msg8

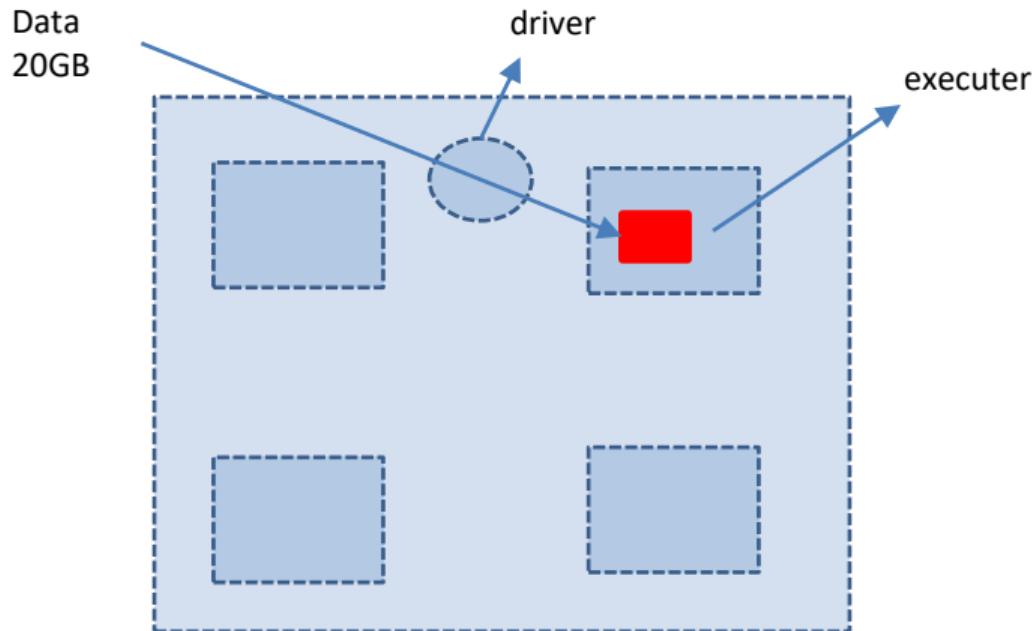
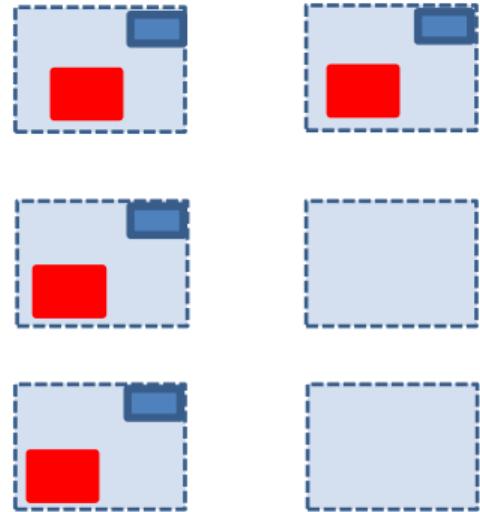
Error, ts, msg3
Info, ts, msg5
Info, ts, msg5

Error, ts, msg4
Warn, ts, msg9
Error, ts, msg1

logLinesRDD
(input/base RDD)



RDD



DataFrames & SparkSQL

- DataFrames (DFs) is one of the other distributed datasets organized in named columns
- Similar to a relational database, Python Pandas Dataframe or R's DataTables
 - Immutable once constructed
 - Track lineage
 - Enable distributed computations
- How to construct Dataframes
 - Read from file(s)
 - Transforming an existing DFs(Spark or Pandas)
 - Parallelizing a python collection list
 - Apply transformations and actions

DataFrame example

```
// Create a new DataFrame that contains “students”
students = users.filter(users.age < 21)
```

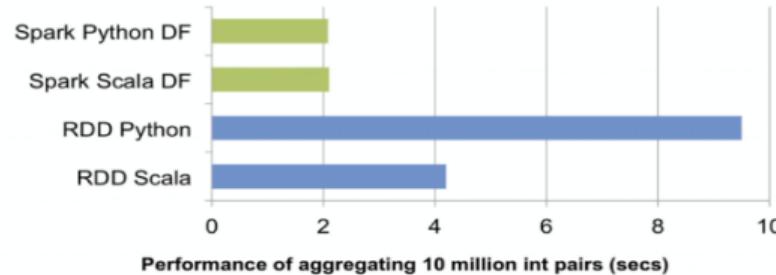
```
//Alternatively, using Pandas-like syntax
students = users[users.age < 21]
```

```
//Count the number of students users by gender
students.groupBy("gender").count()
```

```
// Join young students with another DataFrame called logs
students.join(logs, logs.userId == users.userId,
“left_outer”)
```

RDDs vs. DataFrames

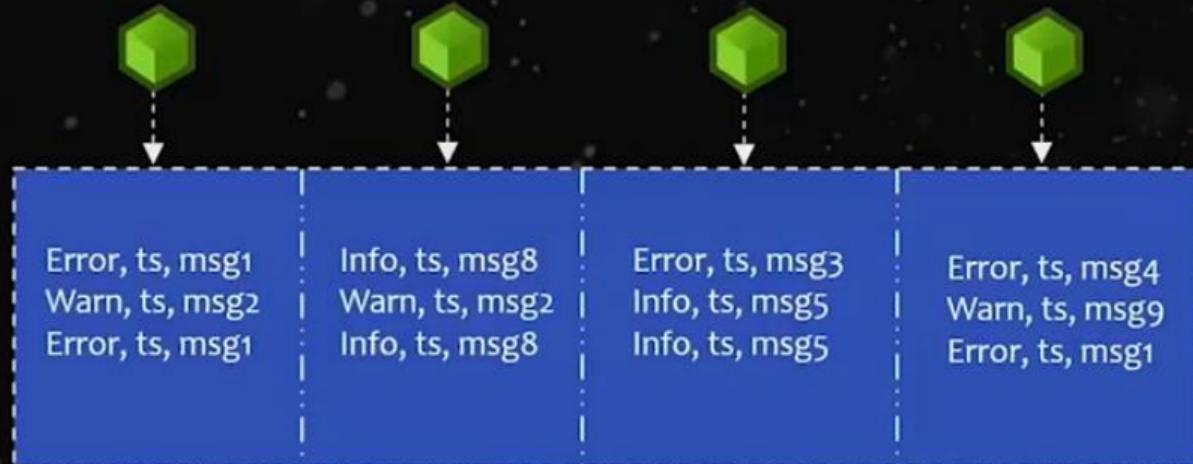
- RDDs provide a low level interface into Spark
- DataFrames have a schema
- DataFrames are cached and optimized by Spark
- DataFrames are built on top of the RDDs and the core Spark API



Example: performance

Spark Operations

Transformations (create a new RDD)	map filter sample groupByKey reduceByKey sortByKey intersection	flatMap union join cogroup cross mapValues reduceByKey
Actions (return results to driver program)	collect Reduce Count takeSample take lookupKey	first take takeOrdered countByKey save foreach



`.filter(f(x))`



logLinesRDD
(input/base RDD)

errorsRDD

Transformation

Error, ts, msg1

Error, ts, msg1

Error, ts, msg3

Error, ts, msg4

Error, ts, msg1

errorsRDD

.coalesce(2)



Error, ts, msg1
Error, ts, msg3
Error, ts, msg1

Error, ts, msg4
Error, ts, msg1

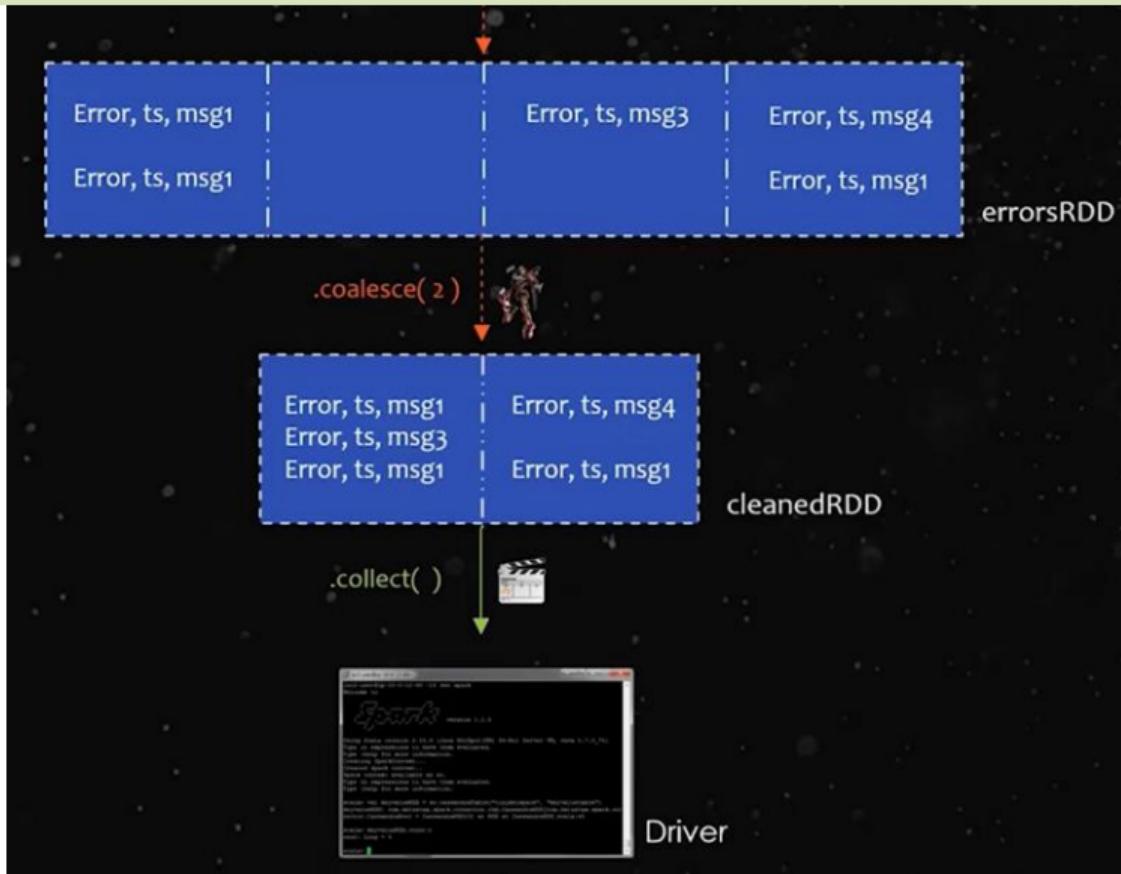
cleanedRDD

Actions

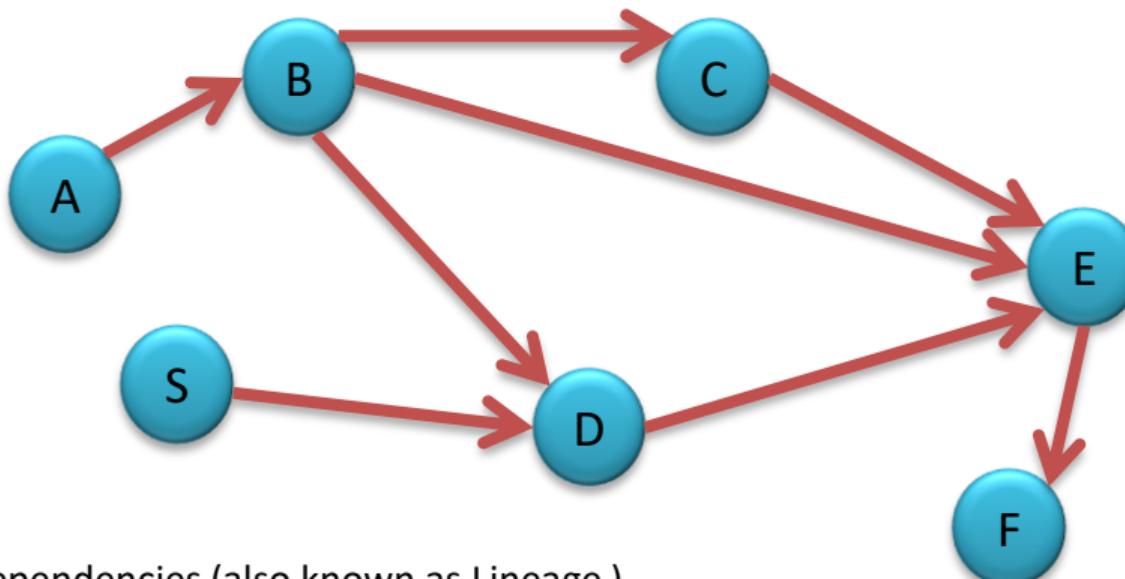
What is an action?

- The final stage of the workflow
- Triggers the execution of the DAG
- Returns the results to the driver
- Or writes the data to HDFS or to a file

Spark Actions



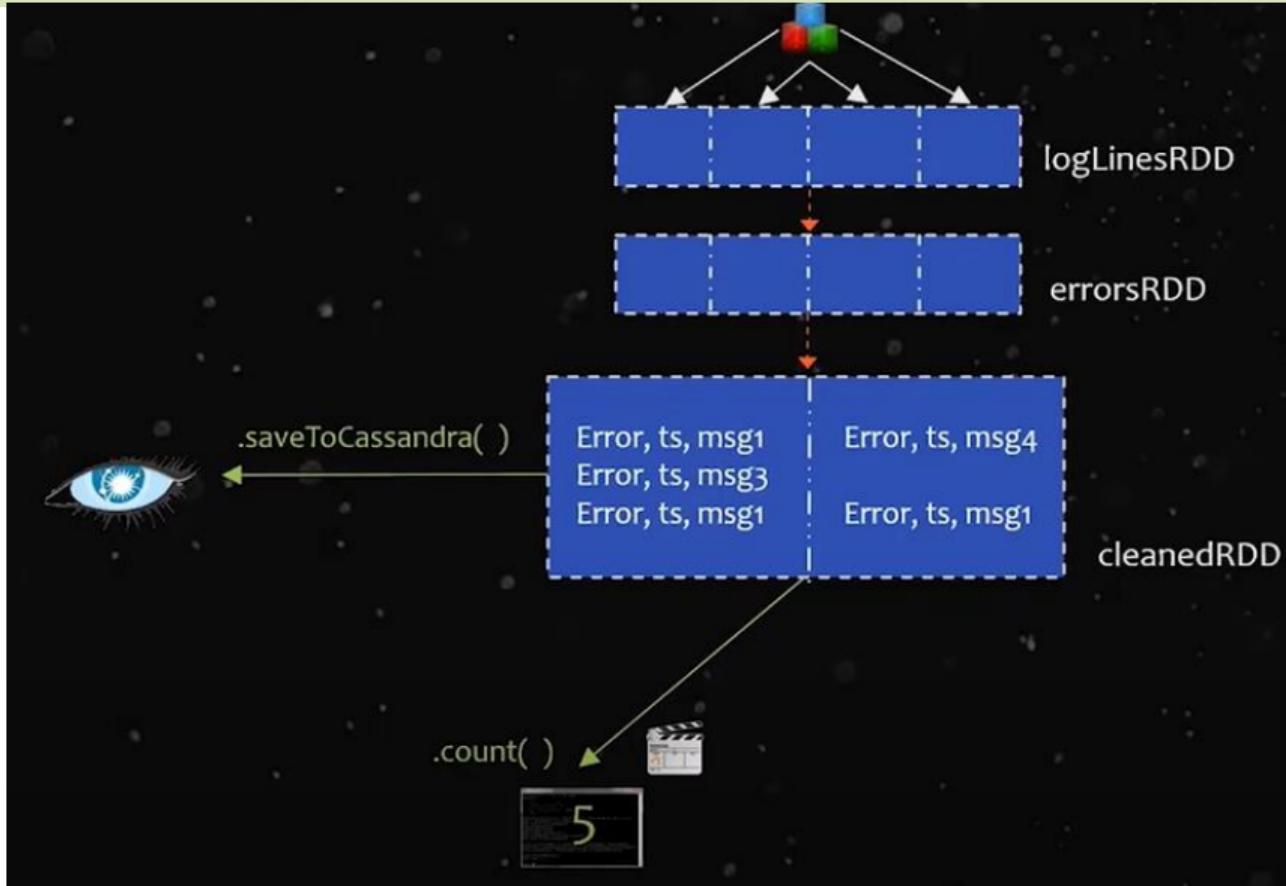
Directed Acyclic Graphs (DAG)



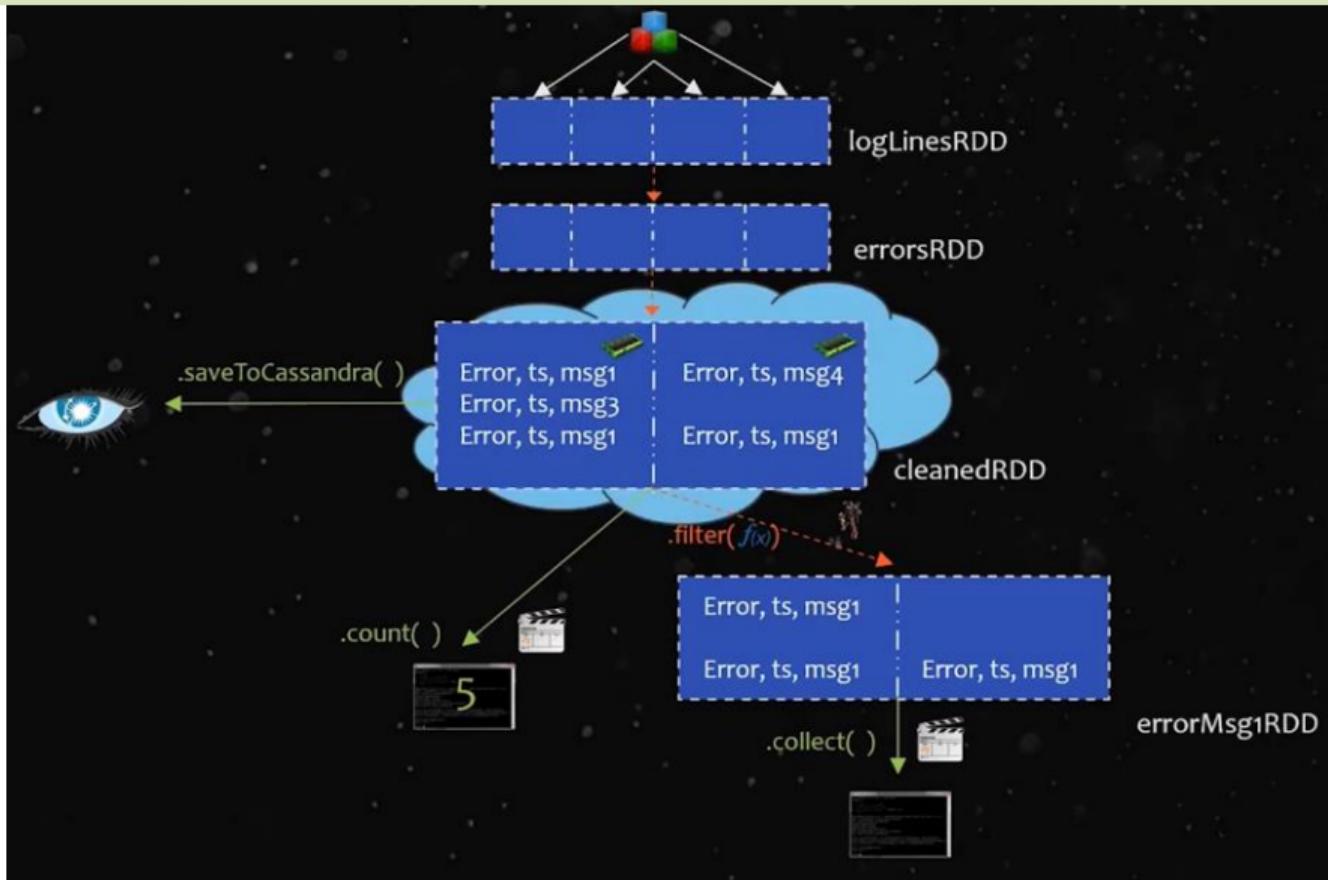
DAGs track dependencies (also known as Lineage)

- nodes are RDDs
- arrows are Transformations

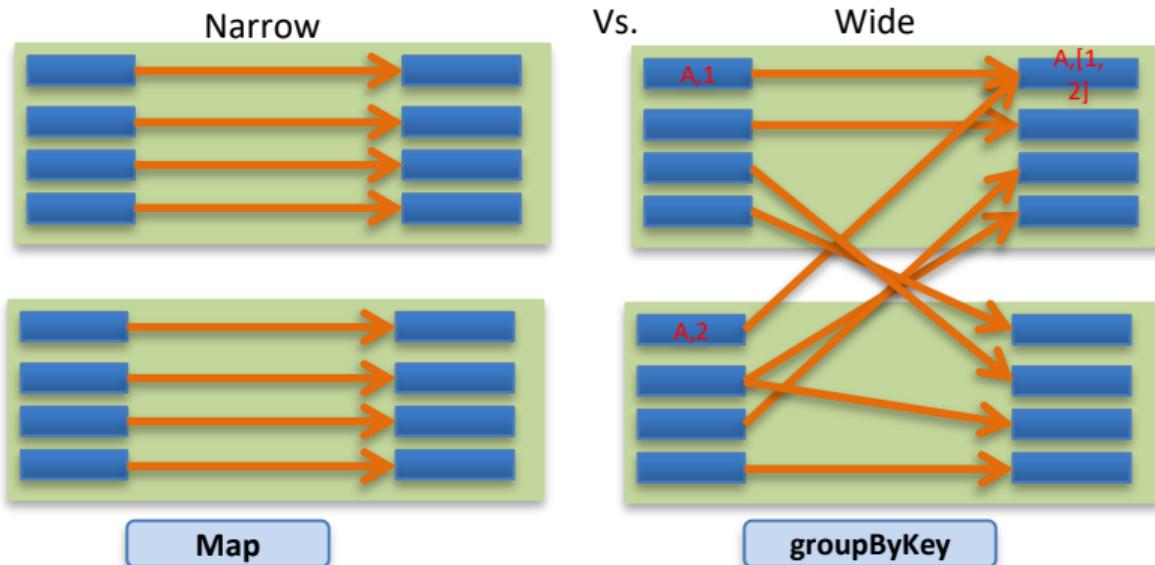
Spark DAG



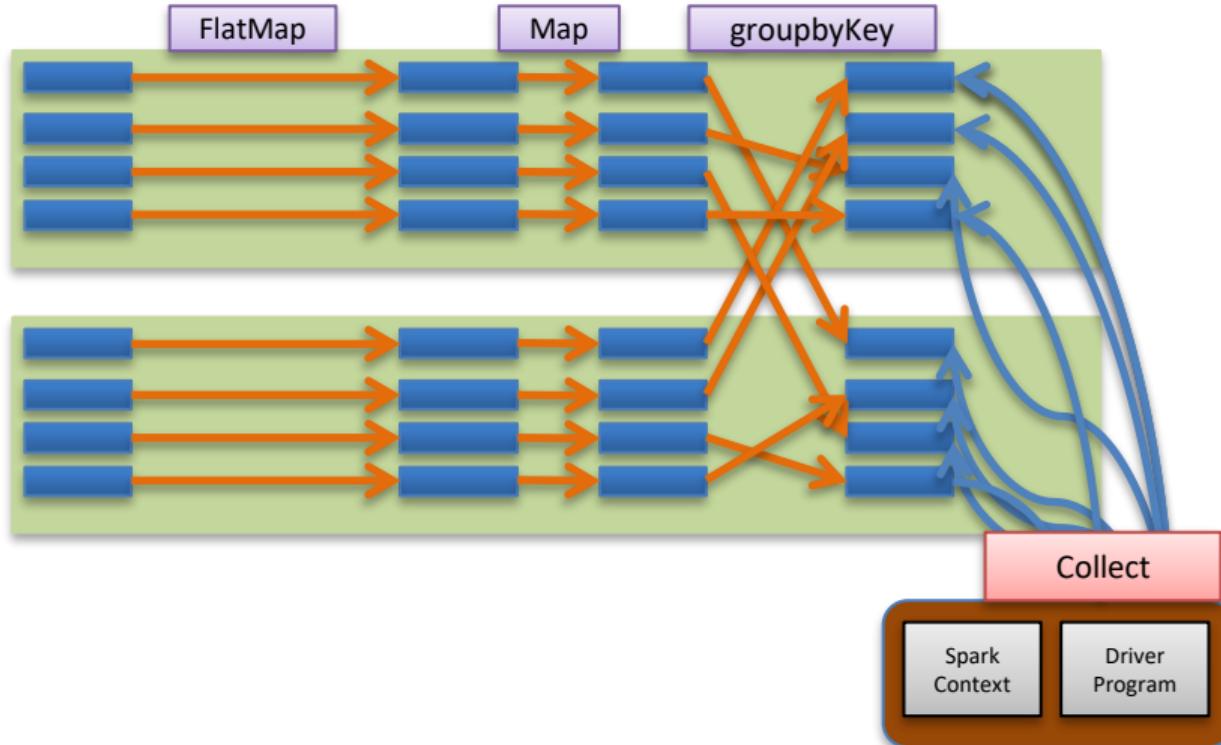
Spark DAG



Narrow Vs. Wide transformation



Spark Workflow



Python RDD API Examples

Word count

```
text_file = sc.textFile("hdfs://usr/godil/text/book.txt")
counts = text_file.flatMap(lambda line: line.split(" ")) \
    .map(lambda word: (word, 1)) \
    .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("hdfs://usr/godil/output/wordCount.txt")
```

Logistic Regression

```
# Every record of this DataFrame contains the label and
# features represented by a vector.
df = sqlContext.createDataFrame(data, ["label", "features"])
# Set parameters for the algorithm.
# Here, we limit the number of iterations to 10.
lr = LogisticRegression(maxIter=10)
# Fit the model to the data.
model = lr.fit(df)
# Given a dataset, predict each point's label, and show the results.
model.transform(df).show()
```

Examples from <http://spark.apache.org/>

RDD Persistence and Removal

RDD Persistence

RDD.persist()

Storage level:

MEMORY_ONLY, MEMORY_AND_DISK, MEMORY_ONLY_SER, DISK_ONLY,.....

RDD Removal

RDD.unpersist()

Broadcast Variables and Accumulators (Shared Variables)

- Broadcast variables allow the programmer to keep a read-only variable cached on each node, rather than sending a copy of it with tasks

```
>broadcastV1 = sc.broadcast([1, 2, 3,4,5,6])
>broadcastV1.value
[1,2,3,4,5,6]
```

- Accumulators are variables that are only “added” to through an associative operation and can be efficiently supported in parallel

```
accum = sc.accumulator(0)
accum.add(x)
accum.value
```

Spark's Main Use Cases

- Streaming Data
 - Machine Learning
 - Interactive Analysis
 - Data Warehousing
 - Batch Processing
 - Exploratory Data Analysis
 - Graph Data Analysis
 - Spatial (GIS) Data Analysis
 - And many more

Spark in the Real World (I)

Uber – the online taxi company gathers terabytes of event data from its mobile users every day.

- By using Kafka, Spark Streaming, and HDFS, to build a continuous ETL pipeline
- Convert raw unstructured event data into structured data as it is collected
- Uses it further for more complex analytics and optimization of operations

Pinterest – Uses a Spark ETL pipeline

- Leverages Spark Streaming to gain immediate insight into how users all over the world are engaging with Pins—in real time.

- Can make more relevant recommendations as people navigate the site

- Recommends related Pins

- Determine which products to buy, or destinations to visit

Spark in the Real World (II)

Here are Few other Real World Use Cases:

Conviva – 4 million video feeds per month

This streaming video company is second only to YouTube.

Uses Spark to reduce customer churn by optimizing video streams and managing live video traffic

Maintains a consistently smooth, high quality viewing experience.

Capital One – is using Spark and data science algorithms to understand customers in a better way.

Developing next generation of financial products and services

Find attributes and patterns of increased probability for fraud

Netflix – leveraging Spark for insights of user viewing habits and then recommends movies to them.

User data is also used for content creation

Spark: when **not** to use

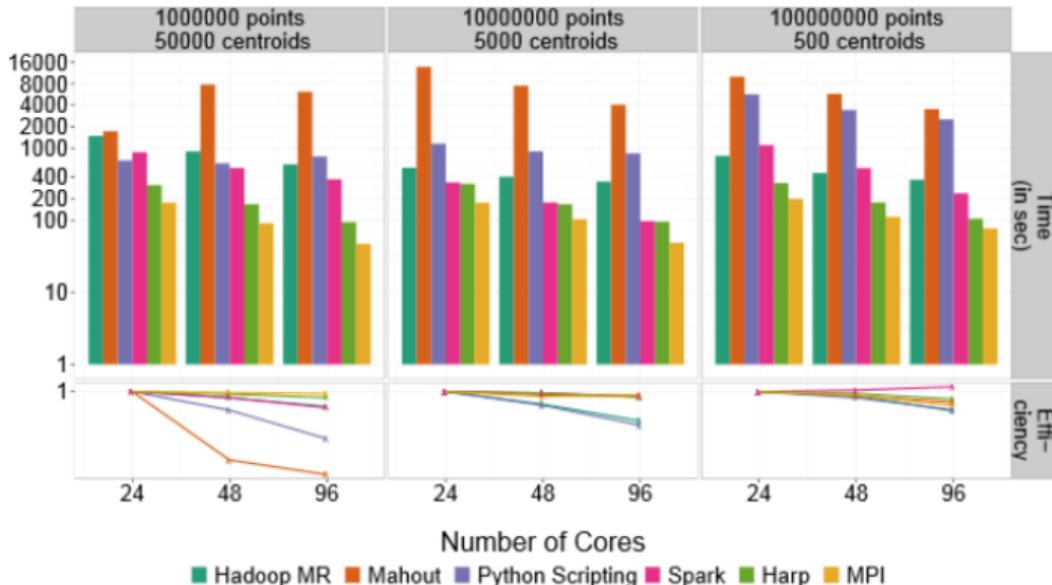
Even though Spark is versatile, that doesn't mean Spark's in-memory capabilities are the best fit for all use cases:

- For many simple use cases Apache MapReduce and Hive might be a more appropriate choice
- Spark was not designed as a multi-user environment
- Spark users are required to know that memory they have is sufficient for a dataset
- Adding more users adds complications, since the users will have to coordinate memory usage to run code

HPC and Big Data Convergence

- Clouds and supercomputers are collections of computers networked together in a datacenter
- Clouds have different networking, I/O, CPU and cost trade-offs than supercomputers
- Cloud workloads are data oriented vs. computation oriented and are less closely coupled than supercomputers
- Principles of parallel computing same on both
- Apache Hadoop and Spark vs. Open MPI

HPC and Big Data K-Means example



MPI definitely outpaces Hadoop, but can be boosted using a hybrid approach of other technologies that blend HPC and big data, including Spark and HARP. Dr. Geoffrey Fox, Indiana University. (<http://arxiv.org/pdf/1403.1528.pdf>)

PGAS Vs MPI vs openMP

	Thread Count	Memory Count	Nonlocal Access
Traditional	1	1	N/A
OpenMP	Either 1 or p	1	N/A
MPI	p	p	No. Message required.
C+CUDA	1+p	2 (Host/device)	No. DMA required.
UPC, CAF, pMatlab	p	p	Supported.
X10, Asynchronous PGAS	p	q	Supported.

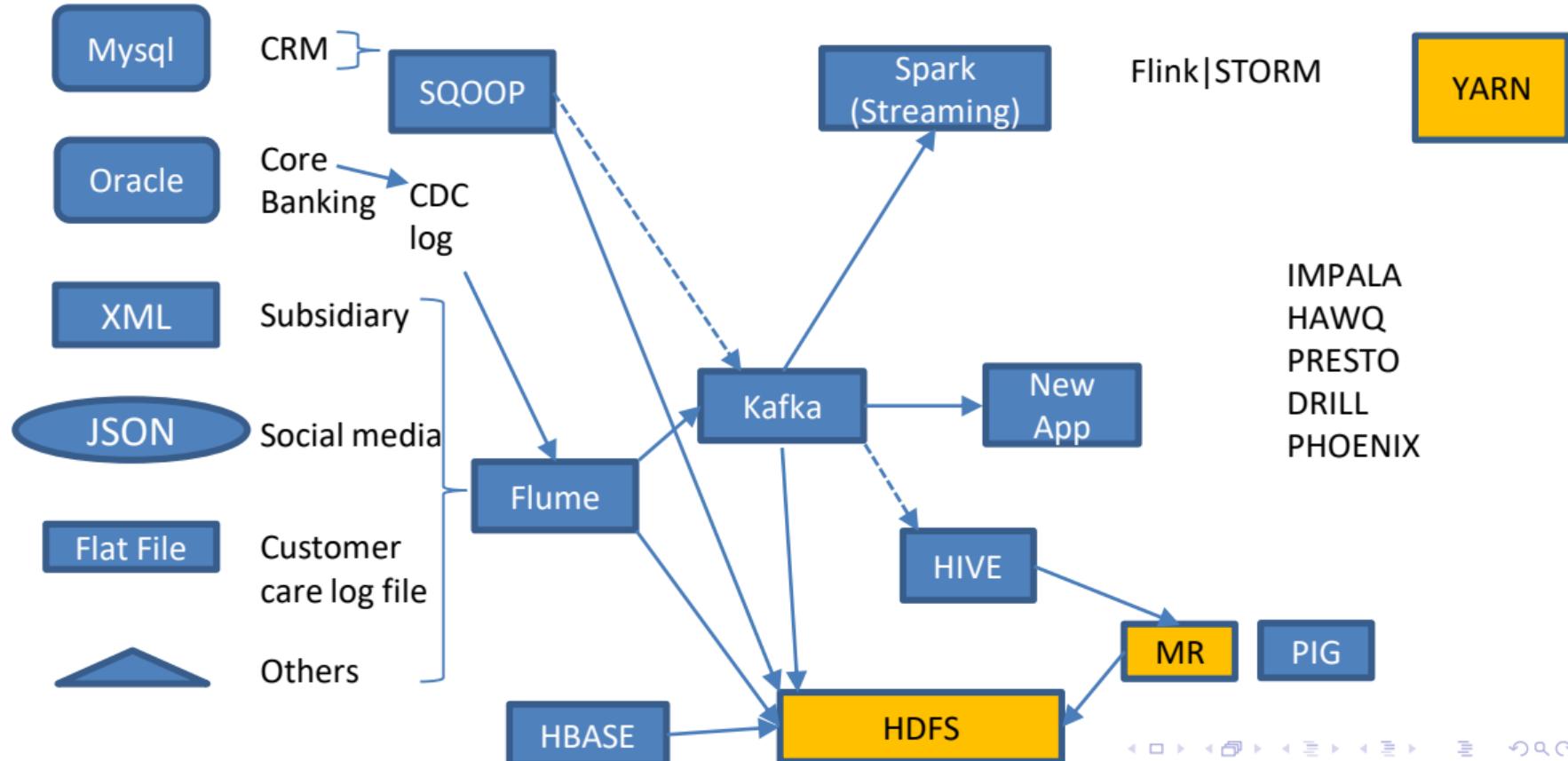
NoSQL Database S

Dilip K. Prasad

Outline

- NoSQL Database
- Vector database

Hadoop Ecosystem-Recap



NoSQL Database

Next Generation Databases mostly addressing some of the points : being non-relational, distributed, open-source and horizontally scalable

NoSQL



NoSQL databases

Key-value stores are the simplest NoSQL databases. Every single item in the database is stored as an attribute name (or 'key'), together with its value. Examples of key-value stores are Riak and Berkeley DB.

Columnar stores such as Cassandra and HBase are optimized for queries over large datasets, and store columns of data together, instead of rows.

Graph stores are used to store information about networks of data, such as social connections. Graph stores include Neo4J and triple stores like Fuseki.

Document databases pair each key with a complex data structure known as a document.

Key-Value databases

- **Key-value** stores are the simplest NoSQL databases. Every single item in the database is stored as an attribute name (or 'key'), together with its value. Examples of key-value stores are Riak and Berkeley DB.
- Key-value pair storage databases store data as a hash table where each key is unique, and the value can be a JSON, BLOB(Binary Large Objects), string, etc.

Key	Value
Name	Joe Bloggs
Age	42
Occupation	Stunt Double
Height	175cm
Weight	77kg

Examples: Redis, Dynamo, Riak

Column databases

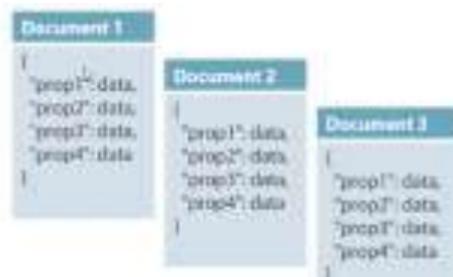
Columnar stores such as Cassandra and HBase are optimized for queries over large datasets, and store columns of data together, instead of rows.

ColumnFamily		
Row Key	Column Name	
	Key	Key
Value	Value	Value
	Column Name	
Key	Key	Key
	Value	Value

Examples: HBase, Cassandra, Hypertable

Document databases

- **Document** databases pair each key with a complex data structure known as a document.
- Document-Oriented NoSQL DB stores and retrieves data as a key value pair but the value part is stored as a document.
- The document is stored in JSON or XML formats.
- The value is understood by the DB and can be queried.



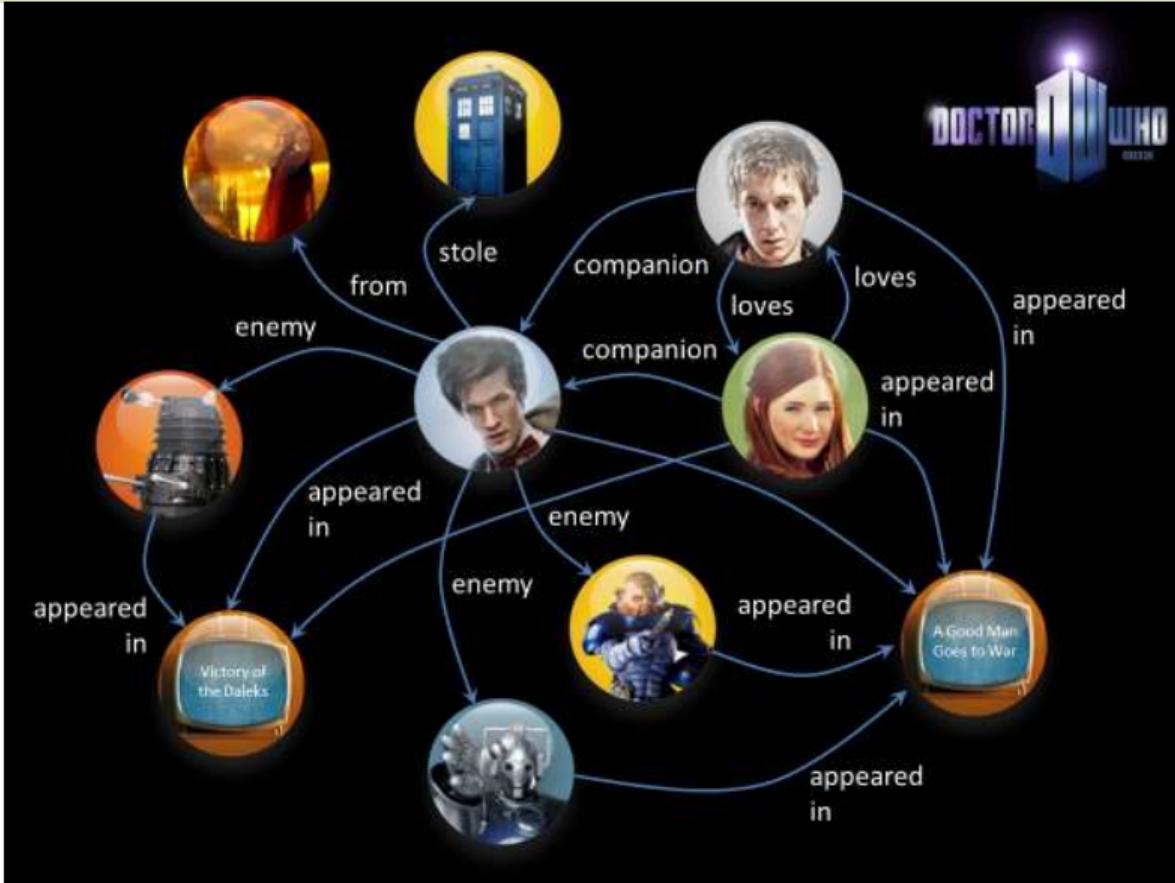
Examples: Amazon SimpleDB,
CouchDB, MongoDB

MongoDB to documents (JSON):

```
{  
    "_id": ObjectId("51156a1e056d6f966f268f81"),  
    "type": "Article",  
    "author": "Derick Rethans",  
    "title": "Introduction to Document Databases with MongoDB",  
    "date": ISODate("2013-04-24T16:26:31.911Z"),  
    "body": "This arti..."},  
{  
    "_id": ObjectId("51156a1e056d6f966f268f82"),  
    "type": "Book",  
    "author": "Derick Rethans",  
    "title": "php|architect's Guide to Date and Time Programming with PHP",  
    "isbn": "978-0-9738621-5-7"}  
}
```



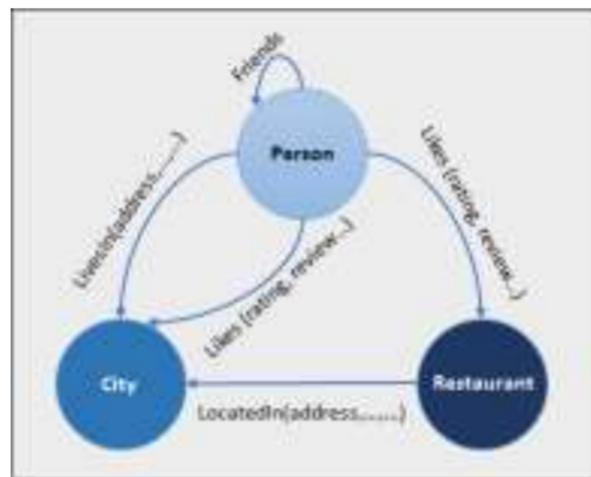
Graph DB



Courtesy: Jim Webber

Graph DB

- A graph type database stores entities as well the relations amongst those entities. The entity is stored as a node with the relationship as edges. An edge gives a relationship between nodes. Every node and edge has a unique identifier.
- Graph base database mostly used for social networks, logistics, spatial data.
- Examples: Neo4J, Infinite Graph, OrientDB, FlockDB



Courtesy: Guru99

Some of the most used No SQL databases

Redis

CouchBase

MongoDB

Amazon DynamoDB

IBM Cloudant

Cassandra

CouchDB

Hbase

Azure CosmosDB

Google Cloud Datastore

Etc.....

NoSQL

NoSQL Features & Capabilities

- Performance
- Availability
- Multi-Model
- Concurrency
- Security
- Scalability
- Data Model Flexibility
- Deployment Model Flexibility

How many NoSQL databases are there?

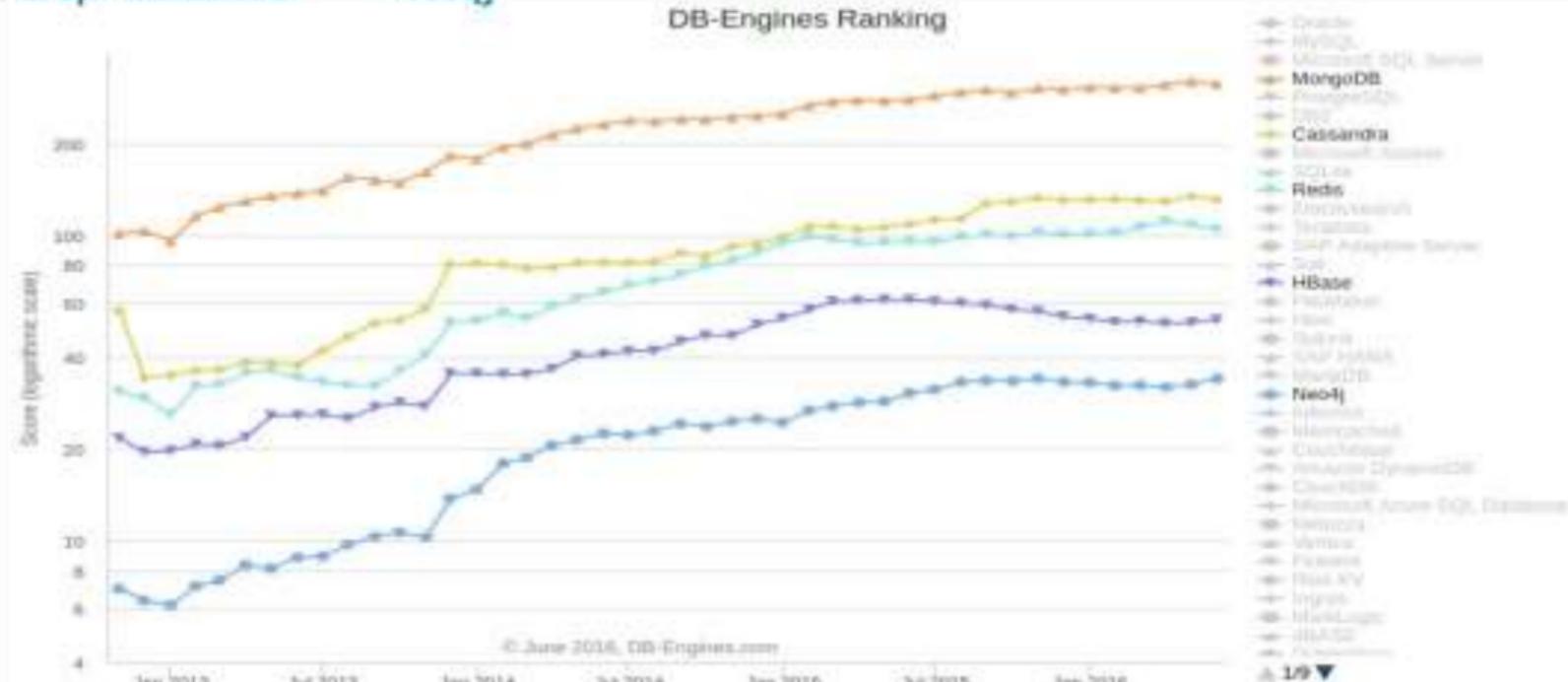


Document databases: MongoDB

Wide-column stores: Cassandra and Hbase

key-value: Redis

Graph database: Neo4j



NoSQL Advantages

- Elastic scalability, because these databases are designed to be used with low-cost commodity hardware
- Support for big data applications, with NoSQL databases able to handle massive volumes of data
- Dynamic schemas, because NoSQL databases require no schemas to start working with data
- Compatibility with cheap commodity hardware clusters as transaction and data volumes increase, allowing you to process and store more data at a lower cost
- Support for auto-sharding, allowing NoSQL databases to natively and automatically spread data across an arbitrary number of servers, without needing the application to be aware of the server pool composition

NoSQL Disadvantages

Does not Guarantee ACID property like RDBMS

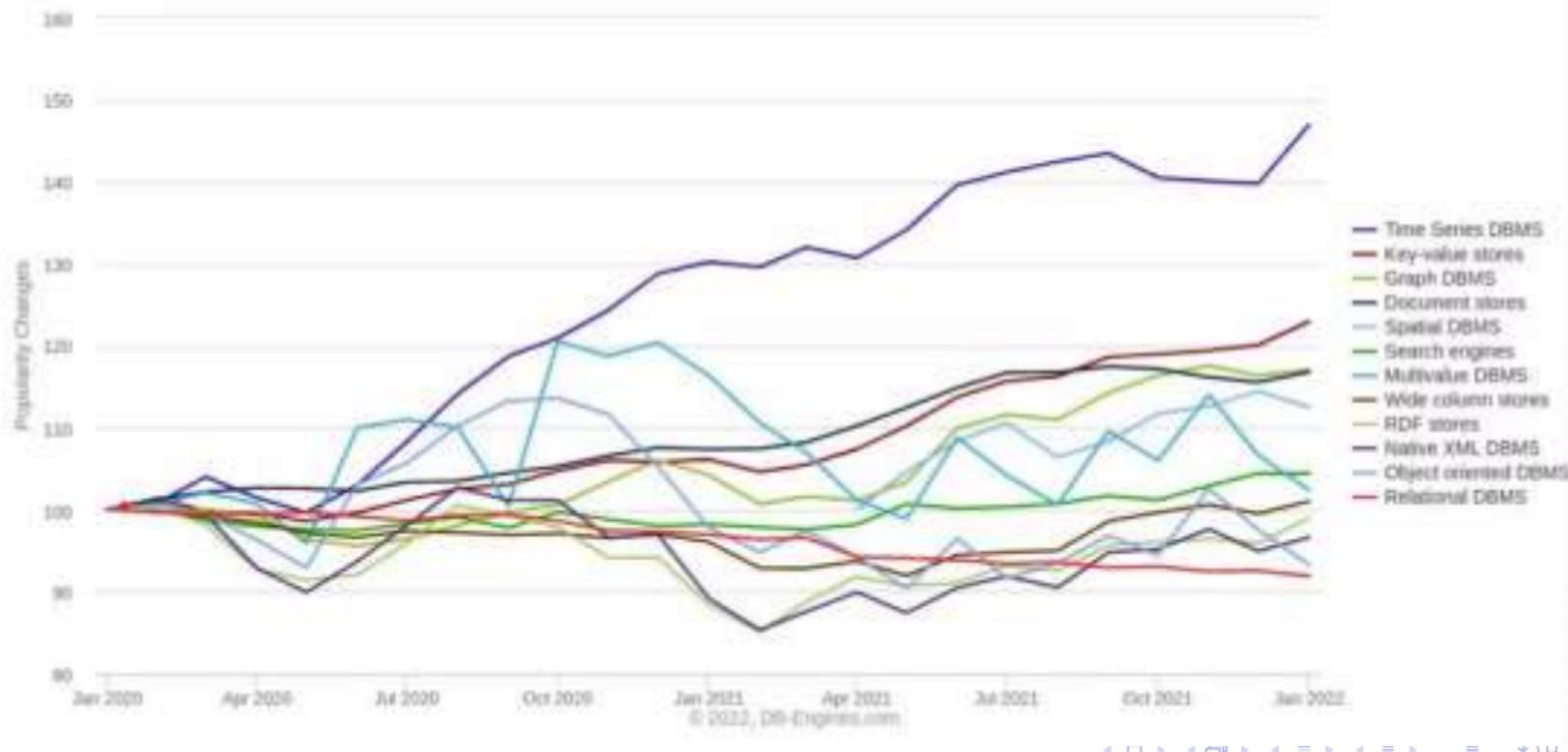
- 1.Atomic: The transaction must perform in an all-or-nothing manner.
- 2.Consistent: Transactions must be processed in a uniform and consistent way.
- 3.Isolated: Transactions must be appropriately isolated until they're complete.
- 4.Durable: The DBMS must maintain a record of any incomplete transactions to facilitate recovery if a failure occurs.

Alternative to ACID in NoSQL support “BASE”, or
“Basically Available, Soft State, Eventual Consistency”

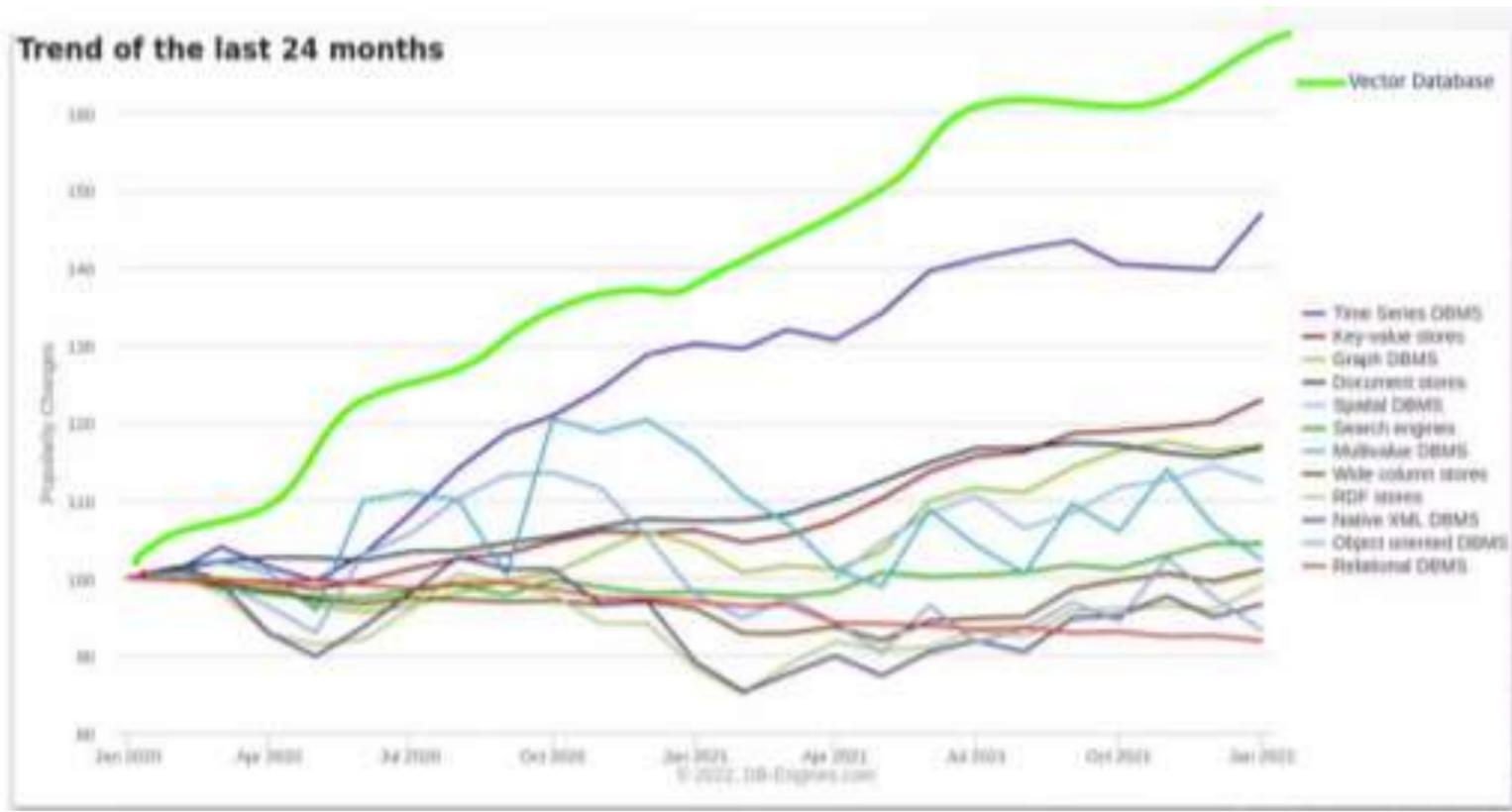
What?
Why?
How?

NoSQL database trend

Trend of the last 24 months



Vector database is a new NoSQL database vertical



Why do we need it?

Visual Search



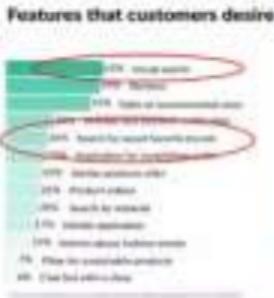
Image Recognition:
face recognition,
similar products

Natural Language Search



Customer Service:
semantic search,
Q&A, content
recommendation,
Jobs

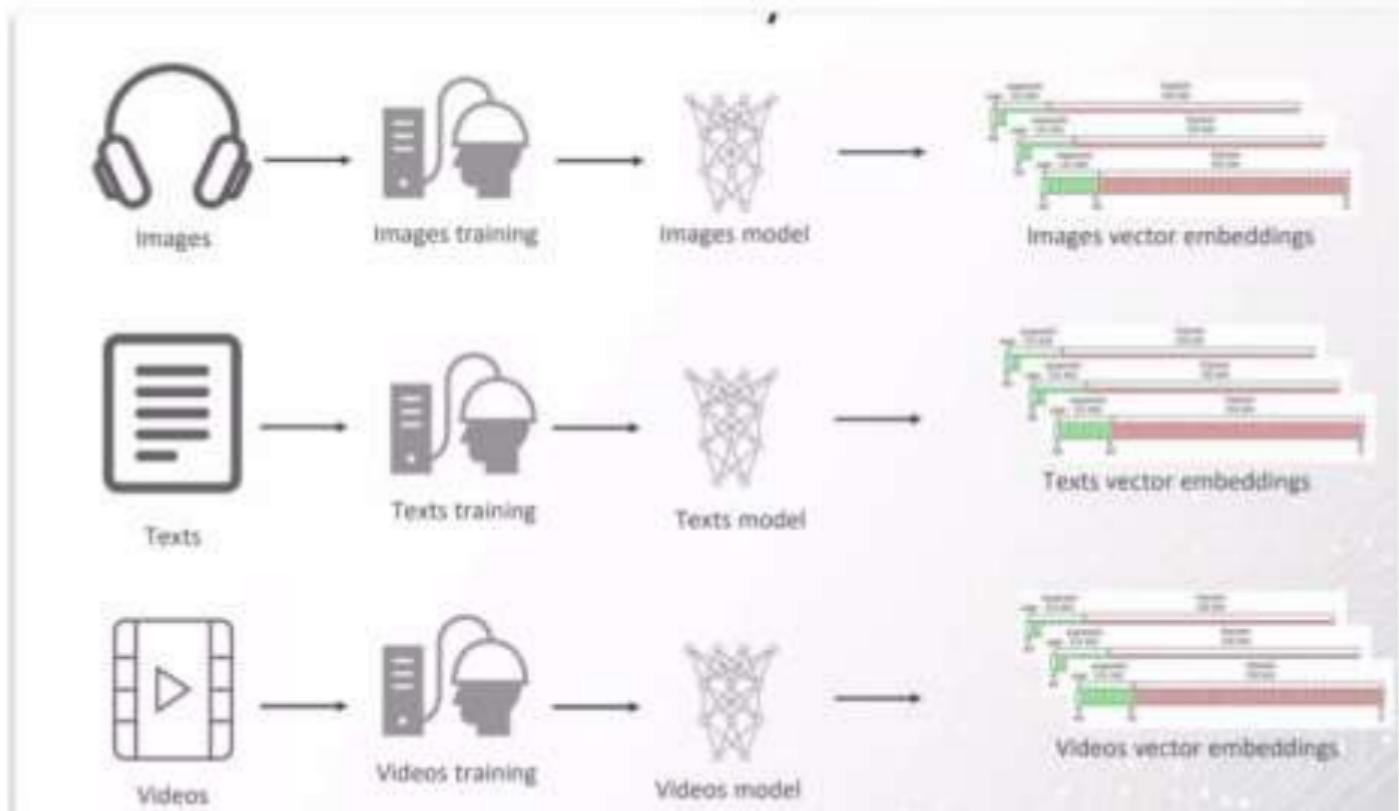
E-Commerce: Recommenders



Recommend:
Similar products,
brands, properties

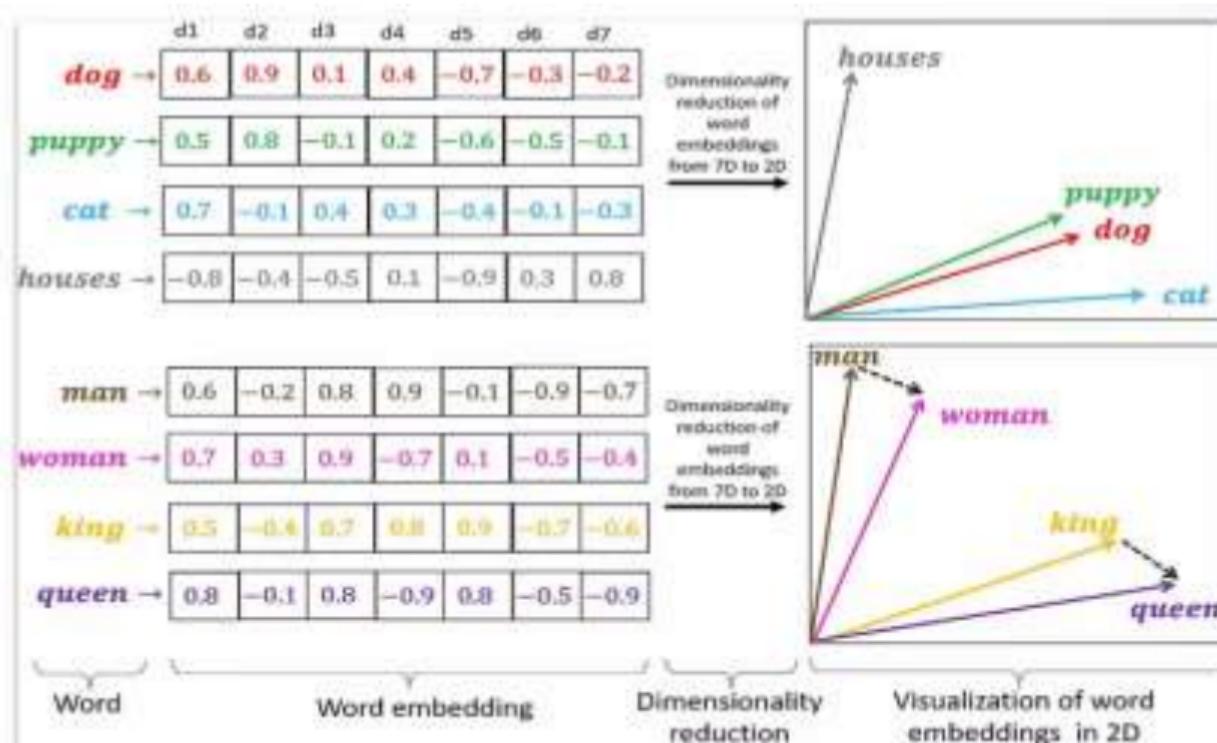
How?

Vector Embeddings

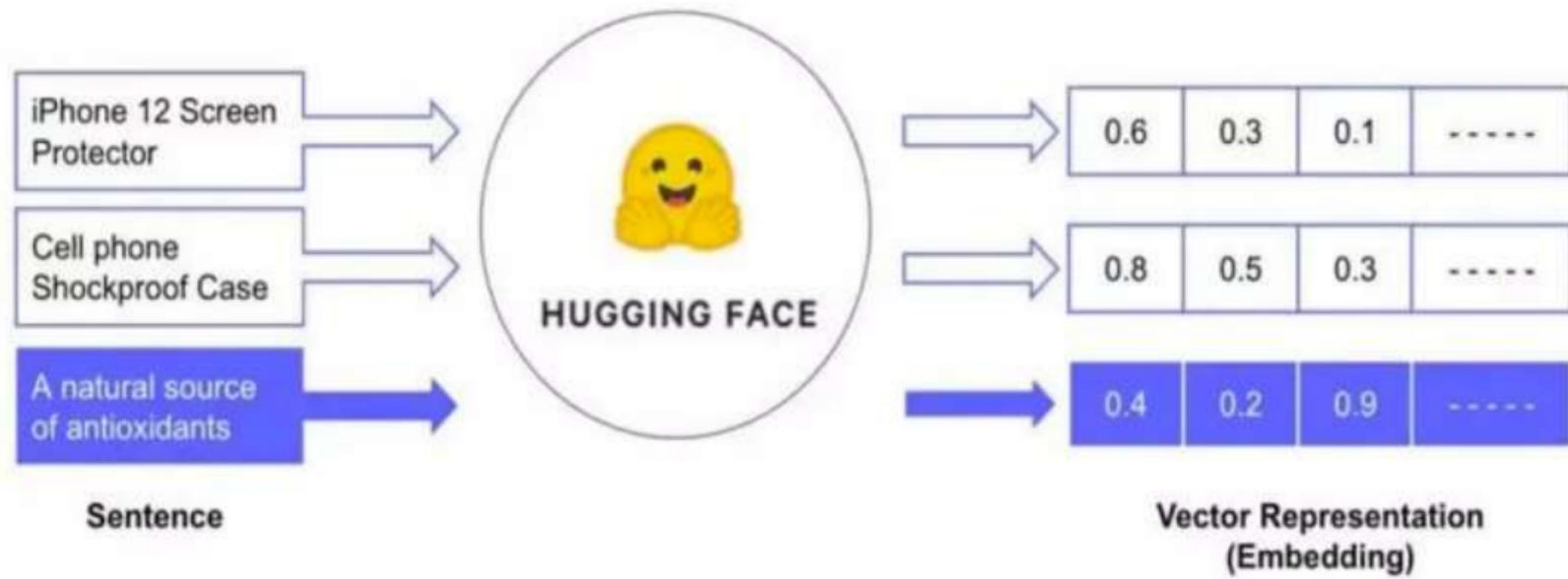


Vector representation of words + similarity (word2vec)

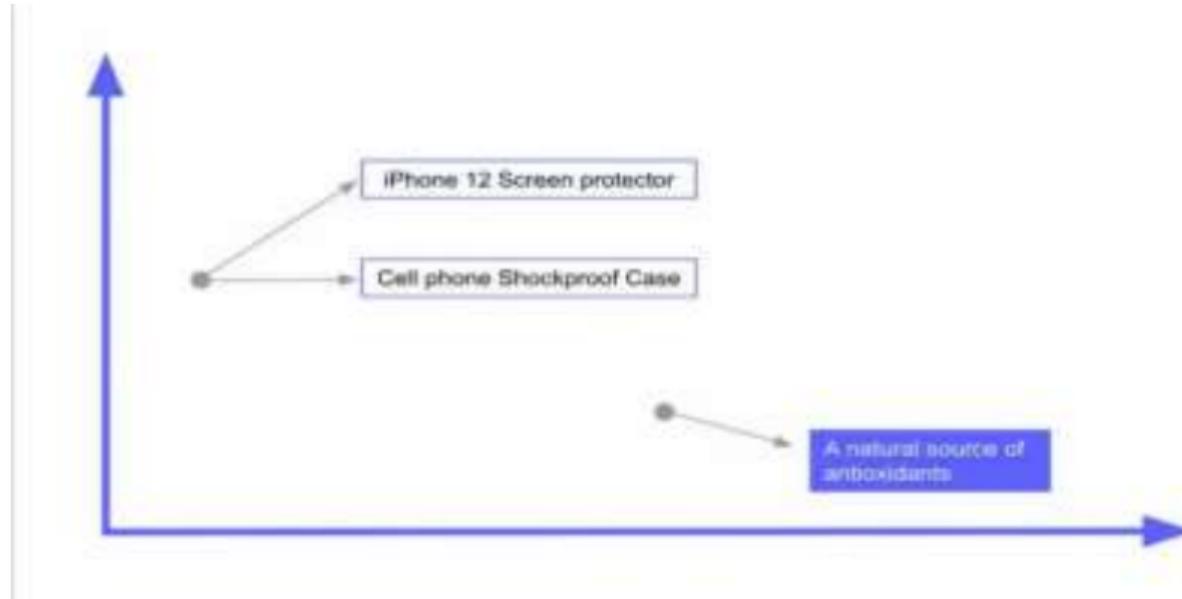
<https://medium.com/@hari4om/word-embedding-d816f643140>



Text embeddings

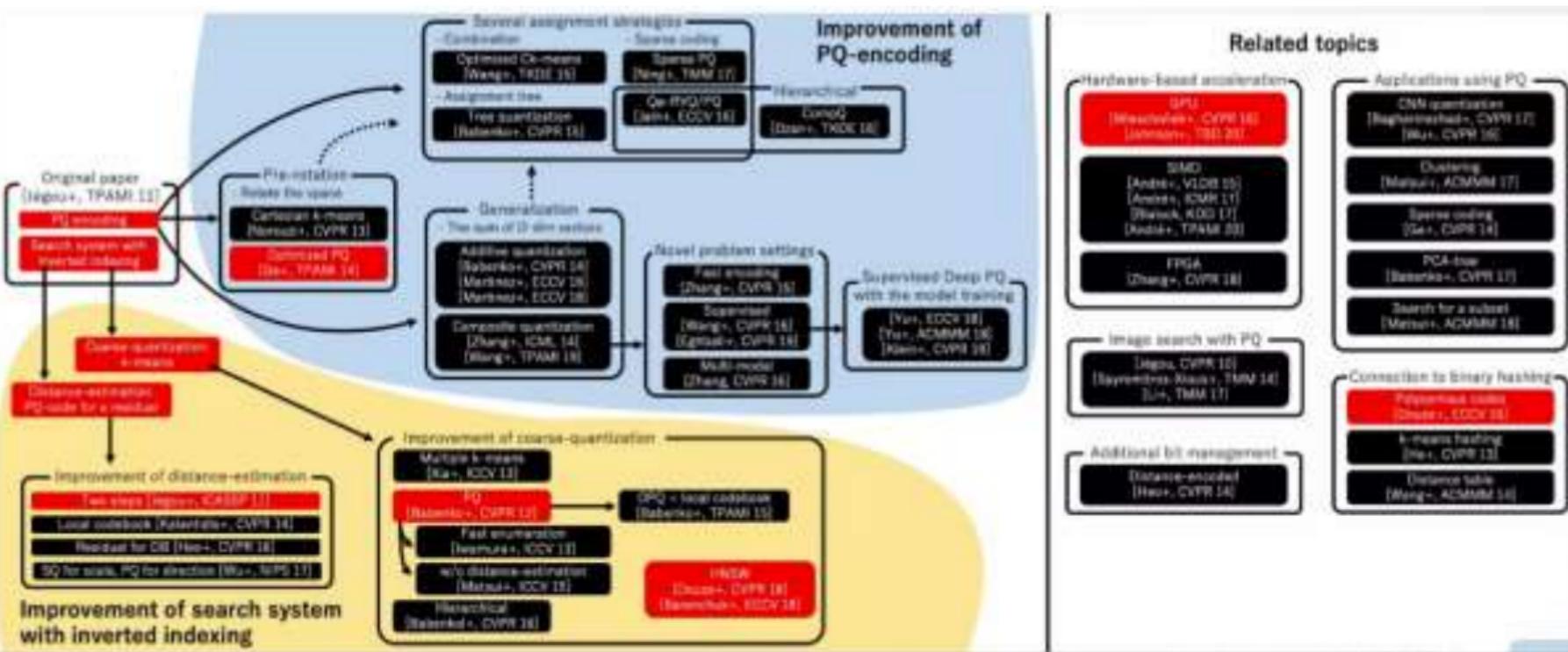


Semantic similarity



Works in the space

<https://github.com/facebookresearch/faiss/wiki>





redis

Example

Catalog

```
{  
    "name": "MY Blue shirt"  
    "images": [  
        [0.2, 0.4, 0.3, 0.5...],  
        [0.1, 0.5, 0.3, 0.2...],  
    ],  
    "price": 19.90,  
    "location": "32.05, 34.680"  
}
```

Example

Find me the **Top 10** products most similar to given image

FT.SEARCH products

"*=>[TOP_K 10 @images \$BLOB]"



Example

Catalog

```
{  
    "name": "MY Blue shirt"  
    "images": [  
        [0.2, 0.4, 0.3, 0.5...],  
        [0.1, 0.5, 0.3, 0.2...],  
    ],  
    "price": 19.90,  
    "location": "32.05, 34.680"  
}
```

Example cont.

Find me the Top 10 products most similar to given image
that cost \$10-\$20
and the store located **10 KM from TLV**

```
FT.SEARCH my_index  
"@price[10 15] @location:[32.082, 34.780 10 km]  
=> [TOP_K 10 @images $BLOB]"
```



Example

Catalog

```
{  
    "name": "MY Blue shirt"  
    "images": [  
        [0.2, 0.4, 0.3, 0.5...],  
        [0.1, 0.5, 0.3, 0.2...],  
    ],  
    "price": 19.90,  
    "location": "32.05, 34.680"  
}
```

Example cont.

Find me the Top 10 products most similar to given image
that cost \$10-\$20
and the store located 10 KM from TLV sort by Price

```
FT.SEARCH my_index  
"@price[10 15] @location:[32.082, 34.780 10 km]  
=> [TOP_K 10 @images $BLOB]"  
SORTBY @price ASC
```

Where to read more? ·

<https://rediseach.io> <https://github.com/RediSearch/rediseach>