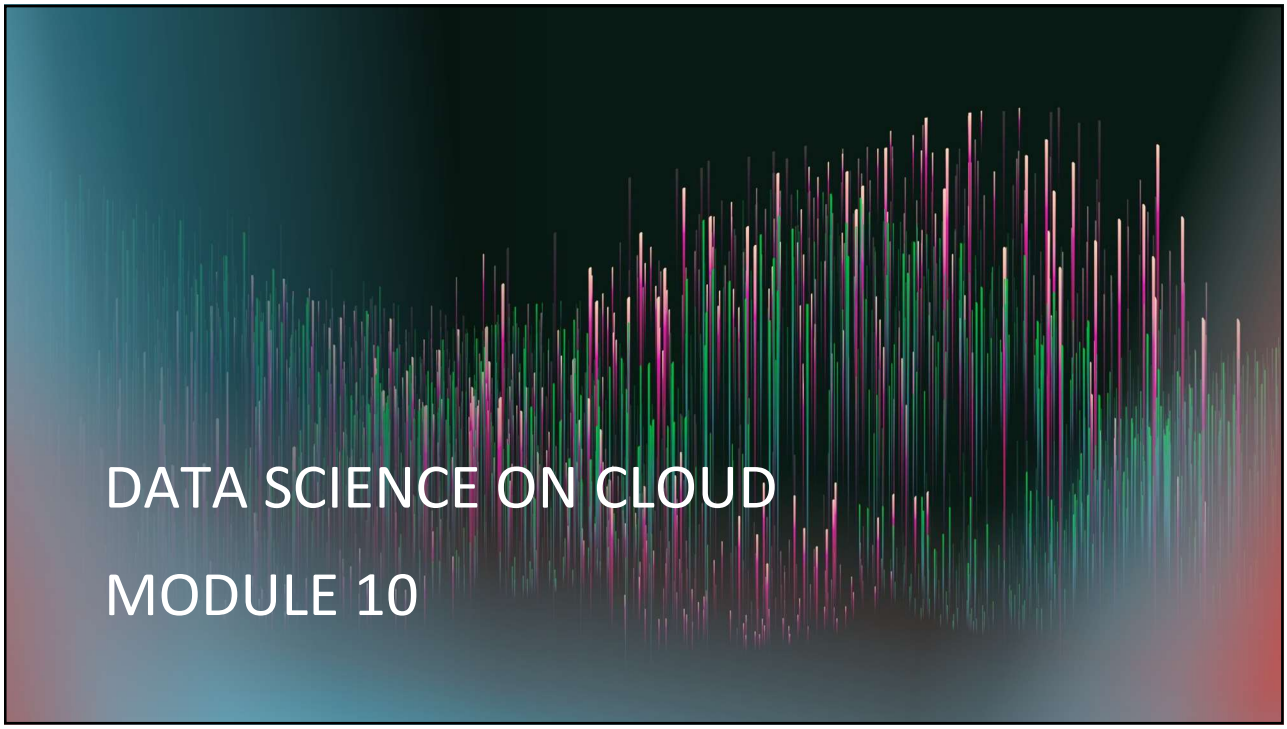


Cloud Architecture

> The ***cloud*** is changing the way applications are ***designed***.

> Instead of monoliths, applications are decomposed into ***smaller***, ***decentralized*** services.

> These services communicate through ***API***s or by using ***asynchronous messaging*** or ***event-driven***.

> Applications ***scale horizontally***, adding ***new instances*** as demand requires.

Challenges in cloud development

> These trends bring new challenges

- Application state is ***distributed***.
- Operations are done in ***parallel*** and ***asynchronously***.
- The system as a whole must be ***resilient*** when ***failures occur***.
- Deployments must be ***automated*** and ***predictable***.
- Monitoring and telemetry are critical for gaining insight into the system.

Challenges in cloud development

Traditional on-premises	Modern Cloud
Monolithic, centralized	Decomposed, de-centralized
Design for predictable scalability	Design for elastic scale
Relational database	Polyglot persistence (mix of storage technologies)
Strong consistency	Eventual consistency
Serial and synchronized processing	Parallel and asynchronous processing
Design to avoid failures (MTBF)	Design for failure (MTTR)
Occasional big updates	Frequent small updates
Manual management	Automated self-management
Snowflake servers	Immutable infrastructure

Challenges in cloud development

- Availability
- Data Management
- Design and Implementation
- Messaging
- Management and Monitoring
- Performance and Scalability
- Resiliency
- Security

Availability

- Availability is the proportion of time that the system is functional and working, usually measured as a percentage of uptime.
- It can be affected by
  - System errors
  - Infrastructure problems
  - Malicious attacks
  - System load
- Cloud applications typically provide users with a service level agreement (SLA), so applications must be designed to maximize availability.

Data Management

- Data management is the key element of cloud applications, and influences most of the quality attributes.
- Data is typically hosted in different locations and across multiple servers
  - Performance
  - Scalability
  - Availability
- This can present a range of challenges:
  - Data consistency must be maintained
  - Data will typically need to be synchronized across different locations.

Design and Implementation

- Good design
  - Consistency and coherence in component design and deployment
  - Maintainability to simplify administration and development
  - Reusability to allow components and subsystems to be used in other applications and in other scenarios.
- Decisions made during the design and implementation phase have a huge impact on the quality and the total cost of ownership of cloud hosted applications and services.

Messaging

- The distributed nature of cloud applications requires a messaging infrastructure
  - Connects the components and services
  - Loosely coupling
    - In order to maximize scalability
- Asynchronous messaging
  - Provides many benefits
  - Brings challenges
    - The ordering of messages
    - Poison message management
    - Idempotency

Management and Monitoring

- Cloud applications run in a remote datacenter
- You do not have full control of the infrastructure or, in some cases, the operating system.
- This can make ***management*** and ***monitoring*** more difficult than an on-premises deployment.
- Applications must ***expose runtime information***
  - To manage and monitor the system
  - To support changing business requirements and customization without requiring the application to be stopped or redeployed

Performance and Scalability

- ***Performance*** is an indication of the ***responsiveness*** of a system to execute any action within a given time interval
- ***Scalability*** is ability of a system either to ***handle increases*** in load ***without impact*** on performance or for the available resources to be readily increased
- Cloud applications typically encounter variable workloads and peaks in activity.
  - Predicting these, especially in a ***multi-tenant*** scenario, is almost impossible.
  - Instead, applications should be able to ***scale out*** within limits to meet peaks in demand, and scale in when demand decreases.

Performance and Scalability

- Scalability concerns ***not just*** ***compute instances***, but other elements such as ***data storage***, ***messaging infrastructure***, and more.

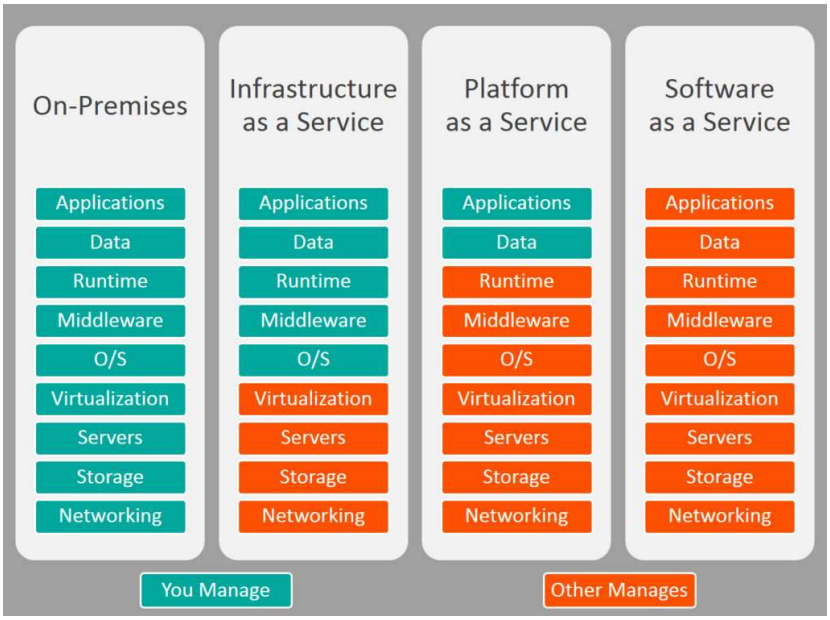
Resiliency

- Resiliency is the ability of a system to gracefully handle and recover from failures.
- The nature of cloud hosting
  - Multi-tenant
  - Use shared platform services
  - Compete for resources and bandwidth
  - Communicate over the Internet
  - Run on commodity hardware
- There is an ***increased likelihood*** that both transient and more permanent faults will arise.
- Detecting failures, and recovering quickly and efficiently, is necessary to maintain resiliency.

Security

- Security is the capability of a system
  - To prevent malicious or accidental actions outside of the designed usage
  - To prevent disclosure or loss of information
- Cloud applications are
  - exposed on the Internet outside trusted on-premises boundaries
  - open to the public, and may serve untrusted users
- Applications must be designed and deployed in a way that
  - **Protects** them from ***malicious attacks***
  - **Restricts** access to ***only approved users***
  - **Protects** ***sensitive data***

Cloud Computing Services



Cloud Computing Services

> Infrastructure as a Service (IaaS)
  – Provides only a base infrastructure (Virtual machine, Software Define Network, Storage attached)
  – End user have to configure and manage platform and environment, deploy applications on it.
> Examples:
  – Amazon Web Services (AWS)
    • Elastic Compute Cloud (EC2)
  – Google Cloud Platform (GCP)
    • Compute Engine (CE)
  – Microsoft Azure
    • Virtual Machine (VM)

Cloud Computing Services

> Platform as a Service (PaaS)
  – Provides a platform allowing end user to develop, run, and manage applications without the complexity of building and maintaining the infrastructure
> Examples:
  – Google App Engine
  – CloudFoundry
  – Heroku
  – **OPEN**SHIFT
  – AWS: Beanstalk
  – Azure: Web Apps

Cloud Computing Services

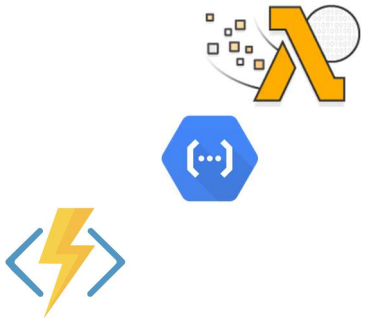
> Software as a Service (SaaS)
  – Called also as “on-demand software”
  – Typically accessed by users using a thin client via a web browser
  – Everything can be managed by vendors: applications, runtime, data, middleware, OSes, virtualization, servers, storage and networking
> Examples:
  – Gmail
  – Microsoft Office 365
  – SAP
  – Salesforce

Cloud Computing Services

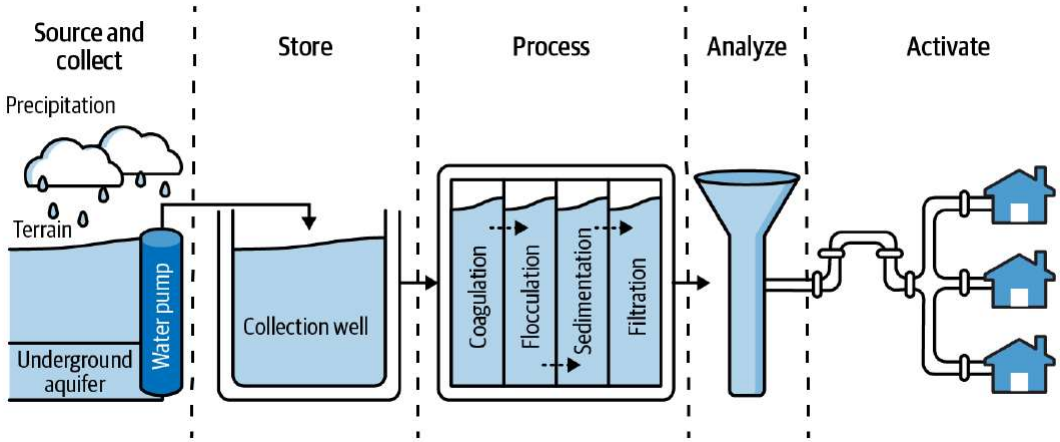
> Container as a Service (CaaS)
  – A form of container-based virtualization in which container engines, orchestration and the underlying compute resources are delivered to users as a service from a cloud provider.
> Examples:
  – Google Container Engine (GKE)
  – AWS ECS
  – Azure Container Service
  – Pivotal Container Service (PKS)

Cloud Computing Services

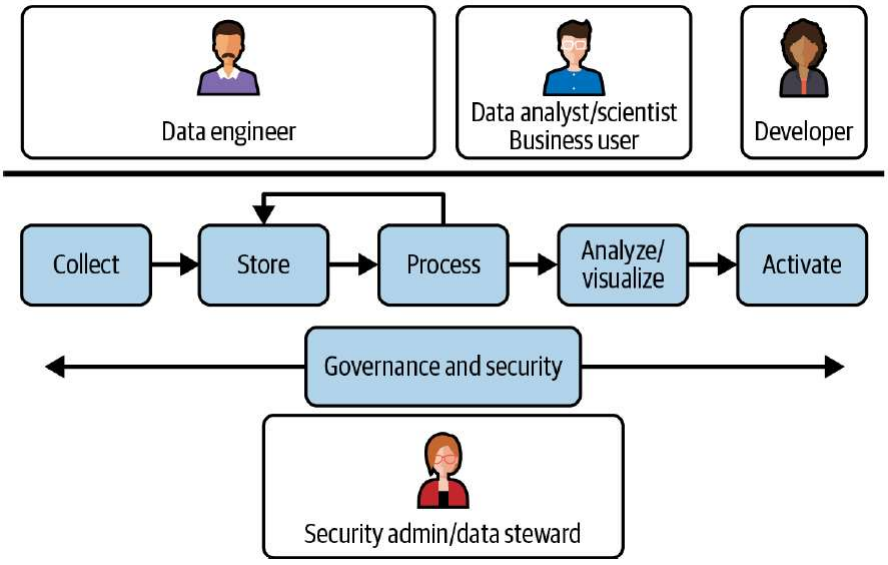
> Function as a Service (FaaS)
  – Provides a platform allowing customers to develop, run, and manage ***application functionalities*** without the complexity of building and maintaining the infrastructure.
> Examples:
  – AWS Lambda
  – Google Cloud Function
  – Azure Function



The data lifecycle



Simplified Data Lifecycle



Collect

- Volume
- Velocity
- Variety

Store

- Store the raw data you collected in the previous step
  - Object storage systems
  - Relational database management systems (RDBMSs)
  - Data warehouses (DWHs)
  - Data lakes

Scalability

- > Vertical scalability
- > Horizontal scalability

UNDERSTANDING BIG DATA

What is big data?

What is big data?

> Big Data is any dataset that cannot be processed or stored using the resources of a single machine to meet the required service level agreements

What is big data?

- Big Data is any dataset that cannot be processed or stored using the resources of a single machine to meet the required service level agreements
- 10 GB high definition video could be a big data for smartphones but not for high-end desktops.

What is big data?

- Big Data is any dataset that cannot be processed or stored using the resources of a single machine to meet the required service level agreements
- 10 GB high definition video could be a big data for smartphones but not for high-end desktops.
- Rendering video from 100 GB 3D graphics data could be a big data for laptop/desktop machines but not for high-end servers.

What is big data?

- A decade back, the size was the first, and at times, the only dimension that indicated big data
- Big (huge) + data (volume + velocity + variety)

  → huge data volume + huge data velocity + huge data variety
- "Big data" not only emerged just from storage capacity (volume) point of view, but also from "processing capability and algorithm ability" of a machine.

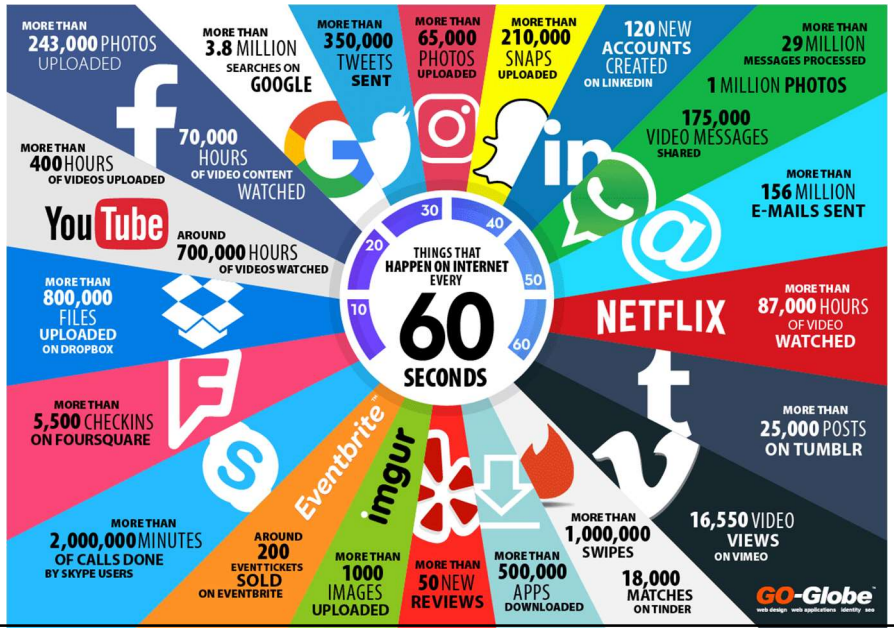
What is big data?

- Every year, data created is almost doubled.
- Over 16 ZB was created in 2016.
- Over 163 ZB was created in 2020.
- In today's digital data world, 90% were created in the last couple of years, in which 95% of data is in semi/unstructured form, and merely less than 5% belongs to structured form of data.

What is big data?

In bytes	Unit	Binary	In bytes	Unit	Binary
1 Bit	0 or 1	-	1024 Kryat byte	1 Amos byte	2^{150} bytes
1 Byte	8 bits	2^{0} bytes	1024 Amos byte	1 Pectrol byte	2^{160} bytes
1024 Bytes	1 Kilo byte	2^{10} bytes	1024 Pectrol byte	1 Bolger byte	2^{170} bytes
1024 Kilo byte	1 Mega byte	2^{20} bytes	1024 Bolger byte	1 Sambo byte	2^{180} bytes
1024 Mega byte	1 Giga byte	2^{30} bytes	1024 Sambo byte	1 Quesa byte	2^{190} bytes
1024 Giga byte	1 Tera byte	2^{40} bytes	1024 Quesa byte	1 Kinsa byte	2^{200} bytes
1024 Tera byte	1 Peta byte	2^{50} bytes	1024 Kinsa byte	1 Ruther byte	2^{210} bytes
1024 Peta byte	1 Exa byte	2^{60} bytes	1024 Ruther byte	1 Dubni byte	2^{220} bytes
1024 Exa byte	1 Zetta byte	2^{70} bytes	1024 Dubni byte	1 Seaborg byte	2^{230} bytes
1024 Zetta byte	1 Yotta byte	2^{80} bytes	1024 Seaborg byte	1 Bohr byte	2^{240} bytes
1024 Yotta byte	1 Bronto byte	2^{90} bytes	1024 Bohr byte	1 Hassiu byte	2^{250} bytes
1024 Bronto byte	1 GeoP byte	2^{100} bytes	1024 Hassiu byte	1 Meitner byte	2^{260} bytes
1024 GeoP byte	1 Sagan byte	2^{110} bytes	1024 Meitner byte	1 Darmstad byte	2^{270} bytes
1024 Sagan byte	1 Pija byte	2^{120} bytes	1024 Darmstad byte	1 Roent byte	2^{280} bytes
1024 Pija byte	1 Alpha byte	2^{130} bytes	1024 Roent byte	1 Coper byte	2^{290} bytes
1024 Alpha byte	1 Kryat byte	2^{140} bytes			

What is big data?



How big is big?

> Size of the job processed by a business unit is **200** GB

How big is big?

> Size of the job processed by a business unit is 200 GB

> Requirement: process the data in **under 5 minutes**!

How big is big?

> Size of the job processed by a business unit is 200 GB
> Requirement: process the data in **under 5 minutes**!
> We read about **50** MB per second

How big is big?

> Size of the job processed by a business unit is 200 GB
> Requirement: process the data in **under 5 minutes**!
> We read about **50** MB per second

200 000 MB / 50 MB/s

How big is big?

> Size of the job processed by a business unit is 200 GB

> Requirement: process the data in **under 5 minutes**!

> We read about **50** MB per second

200 000 MB / 50 MB/s = **4 000 s**

How big is big?

> Size of the job processed by a business unit is 200 GB

> Requirement: process the data in **under 5 minutes**!

> We read about **50** MB per second

200 000 MB / 50 MB/s = **4 000 s**

4 000 s = 4 000 / 60 m = **200 / 3 m**

How big is big?

> Size of the job processed by a business unit is 200 GB
> Requirement: process the data in **under 5 minutes**!
> We read about **50** MB per second

200 000 MB / 50 MB/s = **4 000 s**

4 000 s = 4 000 / 60 m = **200 / 3 m**

66.6 m ≅ **1h**

How big is big?

> Size of the job processed by a business unit is 200 GB
> Requirement: process the data in **under 5 minutes**!
> We read about **50** MB per second

200 000 MB / 50 MB/s = **4 000 s**

4 000 s = 4 000 / 60 m = **200 / 3 m**

66.6 m ≅ **1h**

> ~**1h** to read 200 GB

How big is big?

> Size of the job processed by a business unit is 200 GB

> Requirement: process the data in **under 5 minutes**!

> We read about **50** MB per second

200 000 MB / 50 MB/s = **4 000 s**

4 000 s = 4 000 / 60 m = **200 / 3 m**

66.6 m ≅ **1h**

> **~1h** to read 200 GB

Solution

> Distribute the data (**200 GB**) across **100** nodes

Solution

- Distribute the data (**200 GB**) across **100** nodes
- Each node could process its own data

Solution

- Distribute the data (**200 GB**) across **100** nodes
- Each node could process its own data
- Assemble the results from **100** nodes

Solution

> Distribute the data (**200 GB**) across **100** nodes
> Each node could process its own data
> Assemble the results from **100** nodes
> 2000M / 50 M/s ➔ 40 sec

Solution

> Distribute the data (**200 GB**) across **100** nodes
> Each node could process its own data
> Assemble the results from **100** nodes
> 2000M / 50 M/s ➔ 40 sec
> The total processing can be completed in **under 1 minute**!

Key Idea Behind Big Data Techniques

> Data is distributed across several nodes

Key Idea Behind Big Data Techniques

> Data is distributed across several nodes
  – Network I/O speed << Local Disk I/O Speed

Key Idea Behind Big Data Techniques

- Data is distributed across several nodes
  - Network I/O speed << Local Disk I/O Speed
- Applications are distributed to data nodes in the cluster

Key Idea Behind Big Data Techniques

- Data is distributed across several nodes
  - Network I/O speed << Local Disk I/O Speed
- Applications are distributed to data nodes in the cluster
- As much as possible, **data is processed local to the node**
  - Network I/O speed << Local Disk I/O Speed

Key Idea Behind Big Data Techniques

- Data is distributed across several nodes
  - Network I/O speed << Local Disk I/O Speed
- Applications are distributed to data nodes in the cluster
- As much as possible, **data is processed local to the node**
  - Network I/O speed << Local Disk I/O Speed
- Random disk I/O is replaced by **sequential disk I/O**
  - Transfer Rate << Disk Seek Time

Data Is Distributed Across Several Nodes

- Big Data is data that cannot be processed using the resources of a single machine
- A typical commodity machine would have a 2–4 TB disk
- Not really necessary to have tens of terabytes of data for processing to distribute data across several nodes
- Big Data systems typically process data in place on the node

Data Is Distributed Across Several Nodes

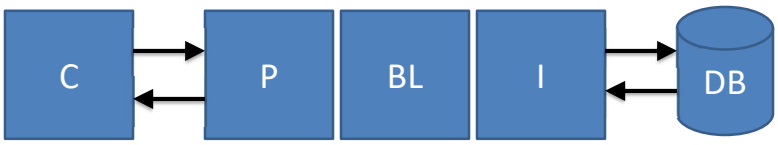
- Big Data is data that cannot be processed using the resources of a single machine
- A typical commodity machine would have a 2–4 TB disk
- Not really necessary to have tens of terabytes of data for processing to distribute data across several nodes
- Big Data systems typically process data in place on the node
- Even a **500 GB** dataset would be **distributed across multiple nodes**, *even if a single machine in the cluster would be capable of storing the data*

Data Distribution

- The purpose of data distribution
  - Each **data block** is replicated across more than one node
    - ***Hadoop replication***
  - **Parallel Processing**
    - ***Local disk is significantly faster than reading from the network***

Applications Are Moved to the Data

> Three-tier architecture



Applications Are Moved to the Data

> Three-tier architecture

> The data is processed in the centralized application tier after being brought into it over the network
> Big Data cannot handle this network overhead
> Moving terabytes of data to the application tier will saturate the networks and introduce considerable inefficiencies, possibly leading to system failure

Data Is Processed Local to a Node

- All Big Data programming models are **distributed-** and **parallel-processing** based
  - Network I/O is orders of magnitude slower than disk I/O
  - Not always possible
  - Schedule tasks on nodes as close to the data as possible

Sequential Reads Preferred Over Random Reads

- Seek operation
  - The disk head needs to be positioned where the data is located on the disk
- Transfer Operation
  - Once the disk head is positioned as needed, the data is read off the disk sequentially

Sequential Reads Preferred Over Random Reads

- Seek time is approximately *10 milliseconds*
- Transfer speeds are on the order of *20 milliseconds*
- Reading **100 MB** from separate **1 MB** sections of the disk
  - 10 (=seek time) * 100 (=seeks) = 1 second
  - 20 (=transfer rate per 1MB) * 100 = 2 seconds.

Big Data Programming Models

- *Massively parallel processing (MPP) database system*
  - EMC's Greenplum
  - IBM's Netezza
  - HP's Vertica
  - Microsoft's DATAllegro

MPP *vs* Hadoop

- MPP uses expensive, specialized hardware tuned for
  - CPU
  - Storage
  - Network performance
- MPP products are bound by the cost of and finite hardware
- MapReduce and Hadoop use clusters of commodity servers that in turn use commodity disks
  - Commodity nature of typical Hadoop hardware means that clusters can grow as data volumes do

MPP *vs* Hadoop

- MapReduce's native control mechanism is Java code (to implement the Map and Reduce logic)
- MPP products are queried with SQL
- In many cases, SQL is easier and more productive than is writing MapReduce jobs
- Database professionals with the SQL skill set are more productive than Hadoop specialists.
- Hive
  - A subproject of the Apache Hadoop project
  - Essentially provides a SQL abstraction over MapReduce

Big Data Programming Models

- *In-memory database systems*
  - Oracle Exalytics
  - SAP HANA
- In-memory databases are blazingly fast

Big Data Programming Models

- *In-memory database systems*
  - Oracle Exalytics
  - SAP HANA
- In-memory databases are blazingly fast
  - *But they are limited in what they can store*

Big Data Programming Models

- *In-memory database systems*
  - Oracle Exalytics
  - SAP HANA
- In-memory databases are blazingly fast
  - *But they are limited in what they can store*
- Hadoop can handle petabytes of information.
  - We aren't getting petabytes into solid state memory in near future

Big Data Programming Models

- *MapReduce systems*
  - These systems include Hadoop

Big Data Programming Models

> Many technical and scientific problems are related to data with the networked nature

Big Data Programming Models

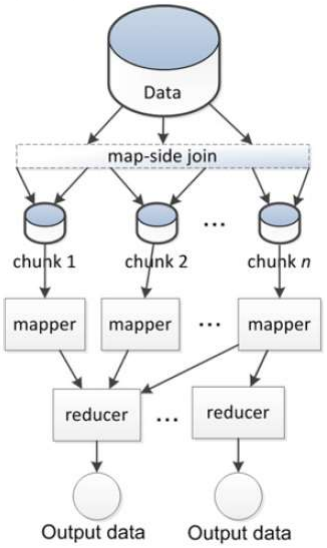
> Many technical and scientific problems are related to data with the networked nature
  – Simply represented by means of graph structures.

Big Data Programming Models

- Many technical and scientific problems are related to data with the networked nature
  - Simply represented by means of graph structures.
  - Graphs provide a very flexible abstraction for describing relationships between discrete objects

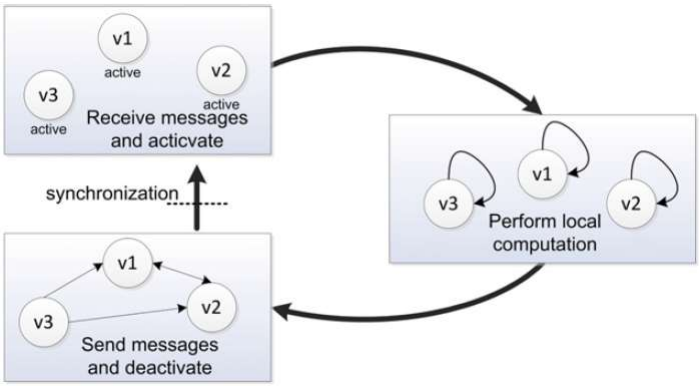
Big Data Programming Models

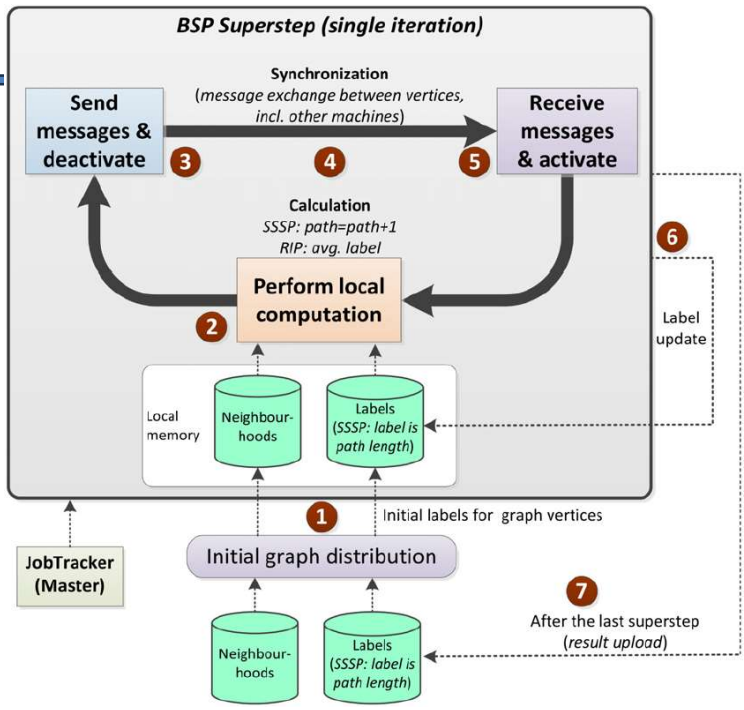
- *Data flow in the MapReduce programming model:*



Big Data Programming Models

> *Data flow in the Bulk Synchronous Parallel programming model:*





Big Data and Transactional Systems

> Big Data uses NoSQL Databases
  – HBase
  – Cassandra
  – Amazon Dynamo

How Much Can We Scale?

> Usually, computational complexity does not dominate
> Computationally expensive problems
  – Data Mining
    • Bayesian statistical techniques
    • Machine/Deep Learning
> Network I/O
  – communication costs for assembling data
> Amdhal's Law

$$S(N) = \frac{1}{(1-P)+\frac{P}{N}}$$

The Data Deluge

- **We're generating more data than ever**
  - Financial transactions
  - Sensor networks
  - Server logs
  - Analytics
  - E-mail and text messages
  - Social media

The Data Deluge (cont'd)

- **And we're generating data faster than ever**
  - Automation
  - Ubiquitous internet connectivity
  - User-generated content
- **For example, every day**
  - X processes 340 million messages
  - Amazon S3 storage adds more than one billion objects
  - Facebook users generate 2.7 billion comments and "Likes"

Data is Value

- **This data has many valuable applications**
  - Marketing analysis
  - Product recommendations
  - Demand forecasting
  - Fraud detection
  - And many, many more…
- **We must process it to extract that value**

Data Processing Scalability

- **How can we process all that information?**
- **There are actually two problems**
  - Large-scale data storage
  - Large-scale data analysis

Disk Capacity and Price

- **We're generating more data than ever before**
- **Fortunately, the size and cost of storage has kept pace**
  - Capacity has increased while price has decreased

Year	Capacity	Cost per GB (USD)
1997	2.1	$157
2004	200	$1.05
2012	3,000	$0.05

Disk Capacity and Performance

- **Disk performance has also increased in the last 15 years**
- **Unfortunately, transfer rates haven't kept pace with capacity**

Year	Capacity (GB)	Transfer Rate (MB/s)	Disk Read Time
1997	2.1	16.6	126 seconds
2004	200	56.5	59 minutes
2012	3,000	210	3 hours, 58 minutes

Disk Capacity and Performance

Year	Cores in a processor	Memory capacity	HDD capacity	Data transfer rate per second in HDD	Time to read entire drive (in minutes)
1990	1	32–128 MB	1 GB	4.5 MB	4.5 ~ 4
2010	2/4/8	16–64 GB	> 1 TB	100 MB	100 ~ 166

Performance

	max read speed	max write speed
HDD	122 MB/s	119 MB/s
SSD	456 MB/s	241 MB/s

Data Access is the Bottleneck

- **Although we can process data more quickly, *accessing* it is slow**
  - This is true for both reads and writes
- **For example, reading a single 3TB disk takes almost four hours**
  - We cannot process the data till we've read the data
  - We're limited by the speed of a single disk
- **We'll see Hadoop's solution in a few moments**
  - But first we'll examine how we *process* large amounts of data

Challenges for Computation and Algorithm Ability

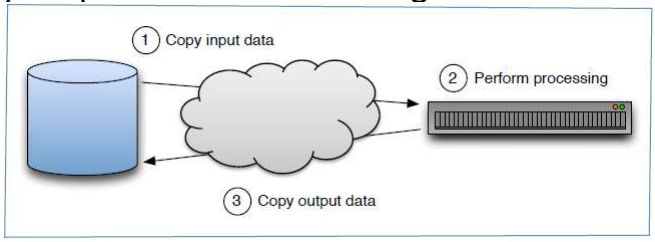
Characteristics	
Volume	CPU heavy IO poor
Velocity	Demands more memory, CPU
Variety	The tool must support to process any type of data
Value	Need potential algorithm to extract more insight
Veracity	Uncertainty of data accuracy and authenticity
Variability	Dynamic and evolving behavior of data source
Volatility	Determines the data validity
Complexity	Unstable number of variables and its interconnectedness

Distributed Computing

- **Modern large-scale processing is distributed across machines**
  - One hundreds or thousands of nodes
  - Common frameworks include MPI, PVM and Condor
- **Focuses on distributing the processing workload**
  - Powerful compute nodes
  - Separate systems for data storage
  - Fast network connections to connect them

Distributed Computing Processing Pattern

- **Typical processing pattern**
  - Step 1: Copy input data from storage to compute node
  - Step 2: Perform necessary processing
  - Step 3: Copy output data back to storage



- **This works fine with relatively small amounts of data**
  - That is, where step 2 dominates overall runtime

Complexity of Distributed Computing

- **Distributed systems pay for scalability by adding complexity**
- **Much of this complexity involves**
  - Availability
  - Data consistency
  - Event synchronization
  - Bandwidth limitations
  - Partial failure
  - Cascading failures
- **These are often more difficult than the original problem**
  - Error handling often accounts for the majority of the code

System Requirements: Failure Handling

- **Failure is inevitable, we should strive to handle it well**
- **An ideal solution should have (at least) these properties**

Failure-Handling Properties of an Ideal Distributed System	
Automatic	Job can still complete without manual intervention
Transparent	Tasks assigned to a failed component are picked up by others
Graceful	Failure results only in a proportional loss of load capacity
Recoverable	That capacity is reclaimed when the component is later replaced
Consistent	Failure does not produce corruption or invalid results

More System Requirements

- **Linear horizontal scalability**
  - Adding new nodes should add proportional load capacity
  - Avoid contention by using a "shared nothing" architecture
  - Must be able to expand cluster at a reasonable cost
- **Jobs run in relative isolation**
  - Results must be independent of other jobs running concurrently
  - Although *performance* can be affected by other jobs
- **Simple programming model**
  - Should support a widely/used language
  - The API must be relatively easy to learn
- **We'll soon see how Hadoop addresses these requirements**

Business Use-Cases for Big Data

> The three big attributes of Big Data
  - Volume
    - The size of the data processed
    - Extract, load, and transform 2 TB of data in 2 hours each night
  - Velocity
    - Speed at which large data arrives
    - Facebook and Twitter encounter velocity problem
  - Variety
    - increasing number of formats that need to be processed

Data Handling Framework

Class	Size	Tools	How it fits
Small	<10 GB	Excel, R, MATLAB	Hardly fits in one computer's memory
Medium	10 GB – 1 TB	DWH	Hardly fits in one computer's storage disk
Big	>1 TB	Hadoop, Distributed DB	Stored across many machines

Operational vs Analytical Big Data Systems

Characteristics	Operational	Analytical
Latency	1–100 ms	1 – 100 minutes
Concurrency	1000–100,000 users	1 -10 users
Access pattern	Write, read, update	Initial load, read, no update
Queries	Selective	Batch fashion
End-user	Customer	Data scientist
Technology	NoSQL	DWH, Hadoop, Spark

HADOOP ECOSYSTEM

Purpose

In this module, you will learn:

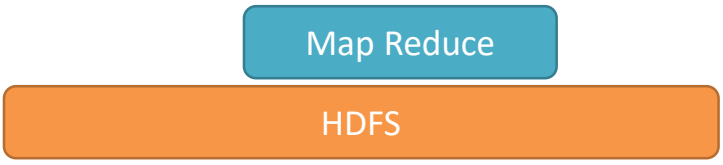
- What the Hadoop Ecosystem is
- What are some popular Hadoop Ecosystem tools
- Which problems these can help you to solve

Apache Hadoop and its Ecosystem

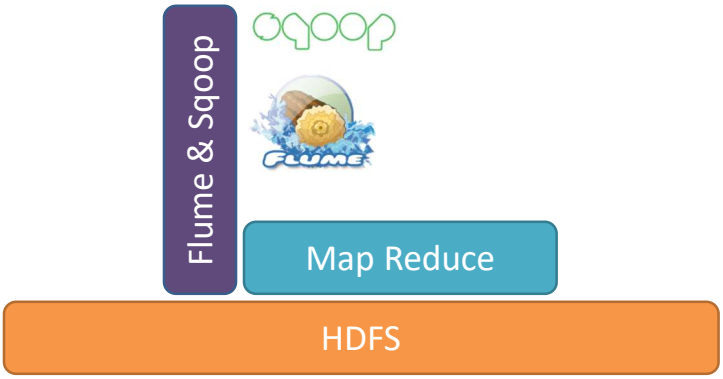
- **Hadoop is a 'top-level' Apache project**
  - Consists of HDFS, MapReduce and basic infrastructure
- **There are many separate, but related, projects**
  - Some are built on top of Hadoop itself
  - Others help you connect Hadoop to other systems
  - They're collectively called the Hadoop ecosystem
- **Ecosystem projects are often also top-level Apache projects**
  - Some are 'Apache incubator' projects
  - Some are not managed by the Apache Software Foundation

Hadoop Technology Stack

Hadoop Technology Stack

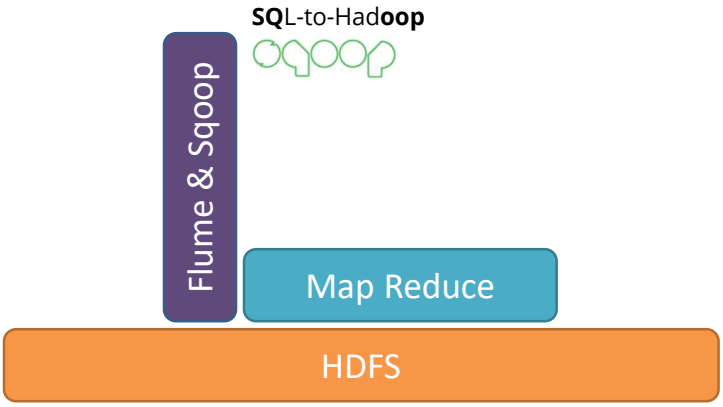


Hadoop Technology Stack



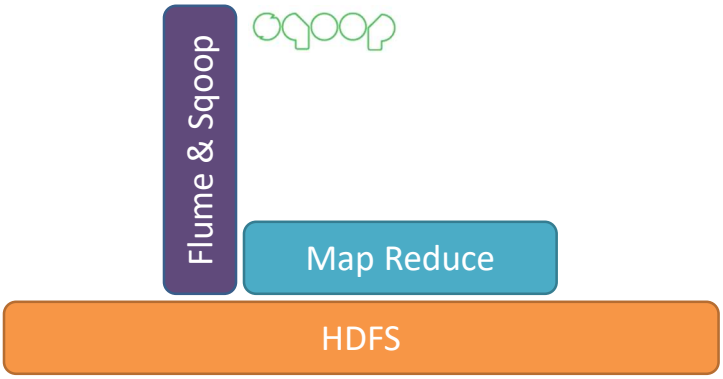
Hadoop Technology Stack

Apache Sqoop is an effective Hadoop tool for ***importing data from RDBMS's.***



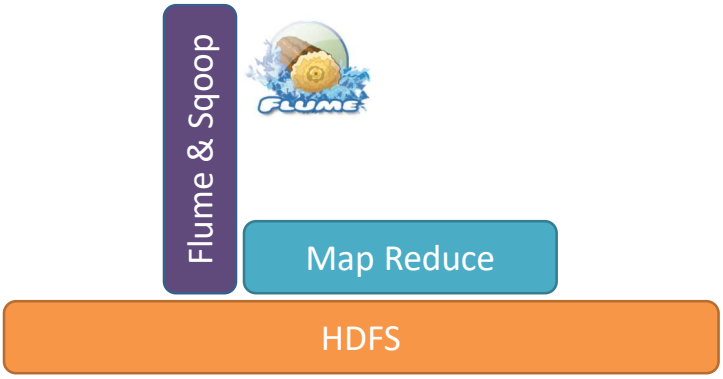
Hadoop Technology Stack

Apache Sqoop is an effective Hadoop tool for ***importing data from RDBMS's.*** **Scoop** works well with any kind of RDBMS that has ***JDBC connectivity***.



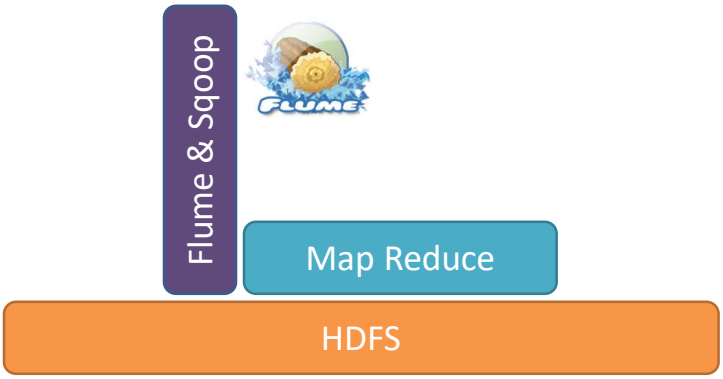
Hadoop Technology Stack

Apache Flume is service designed for streaming logs into Hadoop environment.



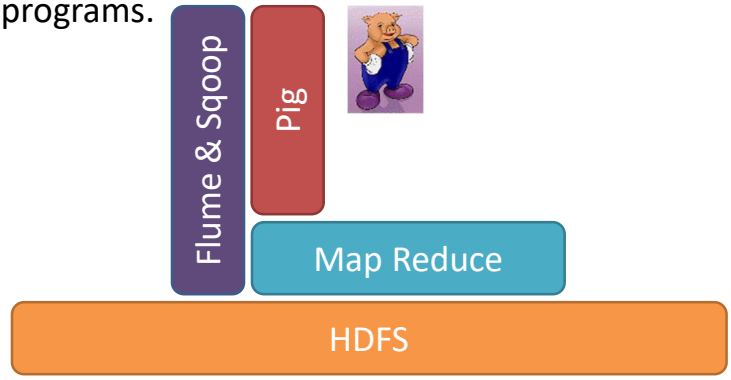
Hadoop Technology Stack

Apache Flume is service designed for streaming logs into Hadoop environment.



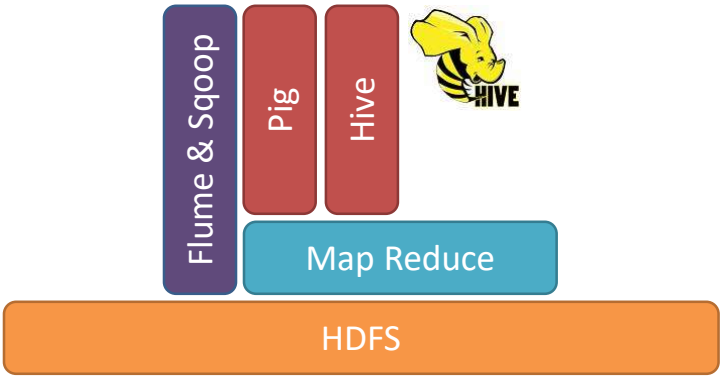
Hadoop Technology Stack

Apache Pig is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs.



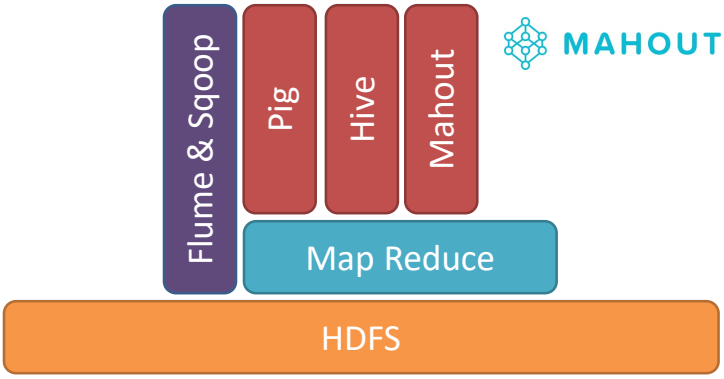
Hadoop Technology Stack

Apache Hive data warehouse software facilitates reading, writing, and managing large datasets residing in distributed storage using SQL.

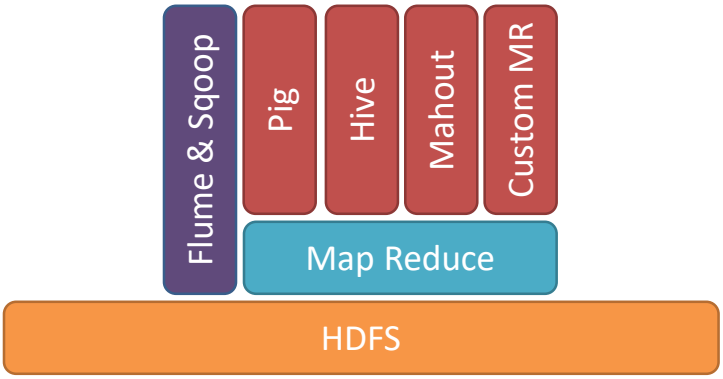


Hadoop Technology Stack

Apache Mahout is a **distributed linear algebra framework** and **mathematically expressive Scala DSL**

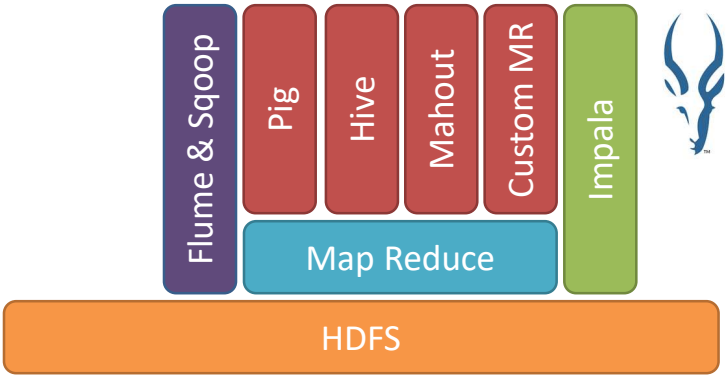


Hadoop Technology Stack



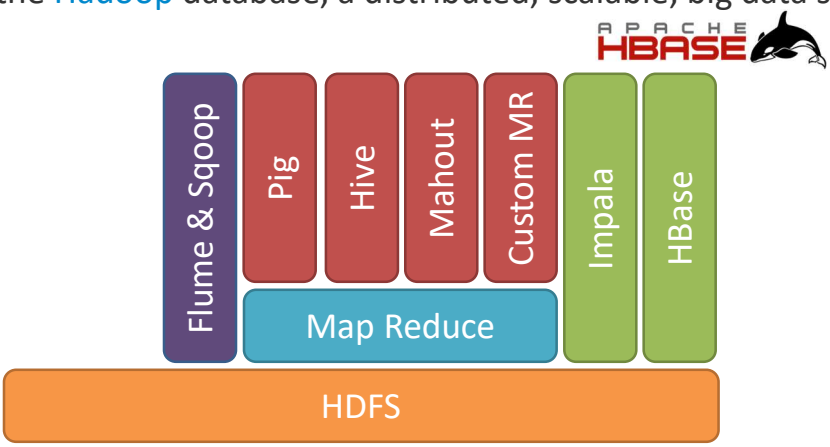
Hadoop Technology Stack

Apache Impala is the open source, native analytic database for Apache Hadoop.



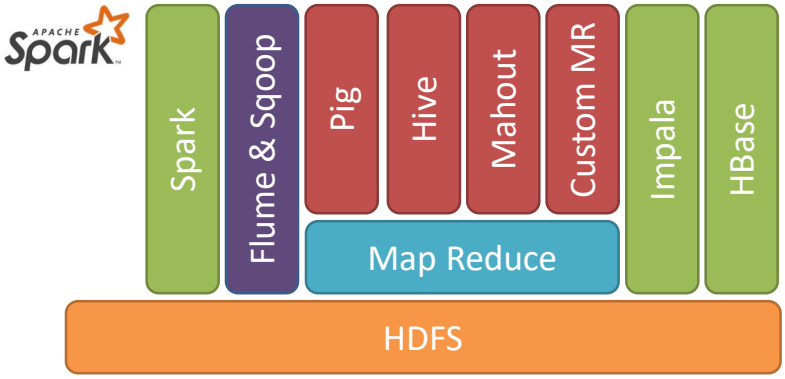
Hadoop Technology Stack

Apache **HBase** is the Hadoop database, a distributed, scalable, big data store.



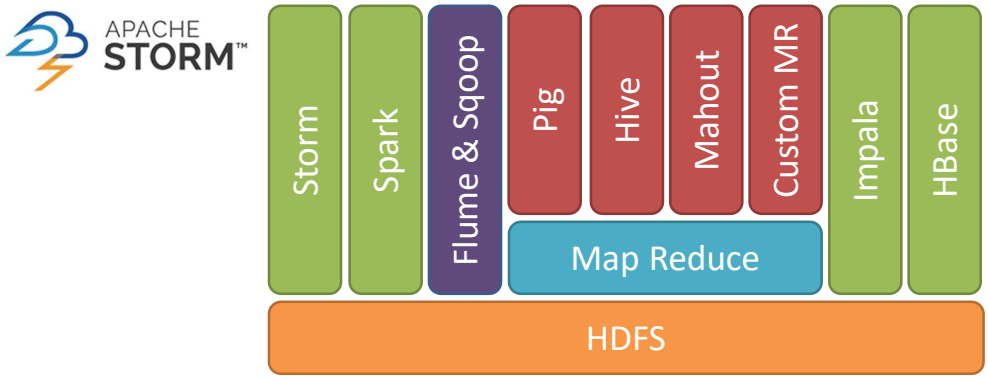
Hadoop Technology Stack

Apache Spark is a unified analytics engine for large-scale data processing.

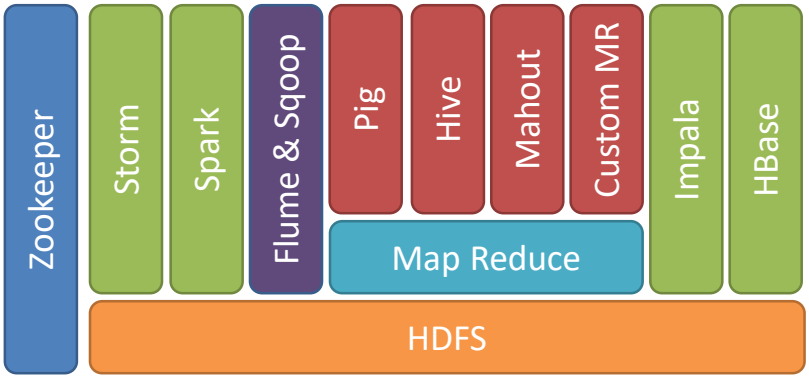


Hadoop Technology Stack

Apache Storm is a free and open source distributed real-time computation system.

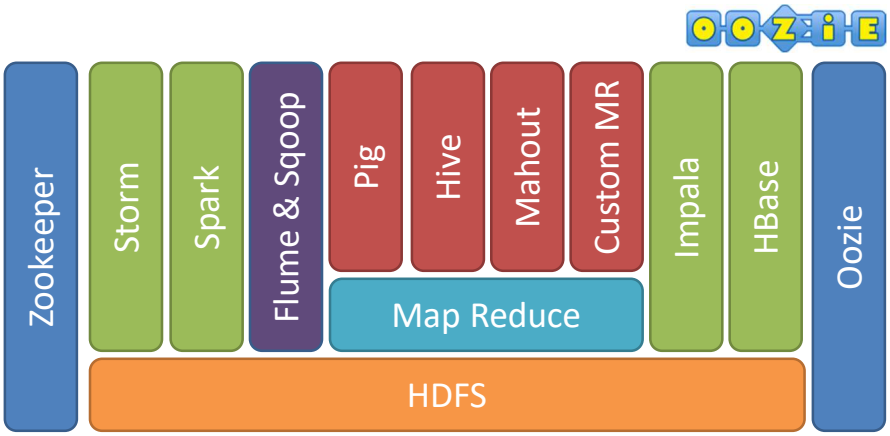


Hadoop Technology Stack



Apache ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services.

Hadoop Technology Stack



Apache Oozie is a workflow scheduler system to manage Apache Hadoop jobs.