Ysgol Reolaeth Gogledd Cymru | North Wales Management School

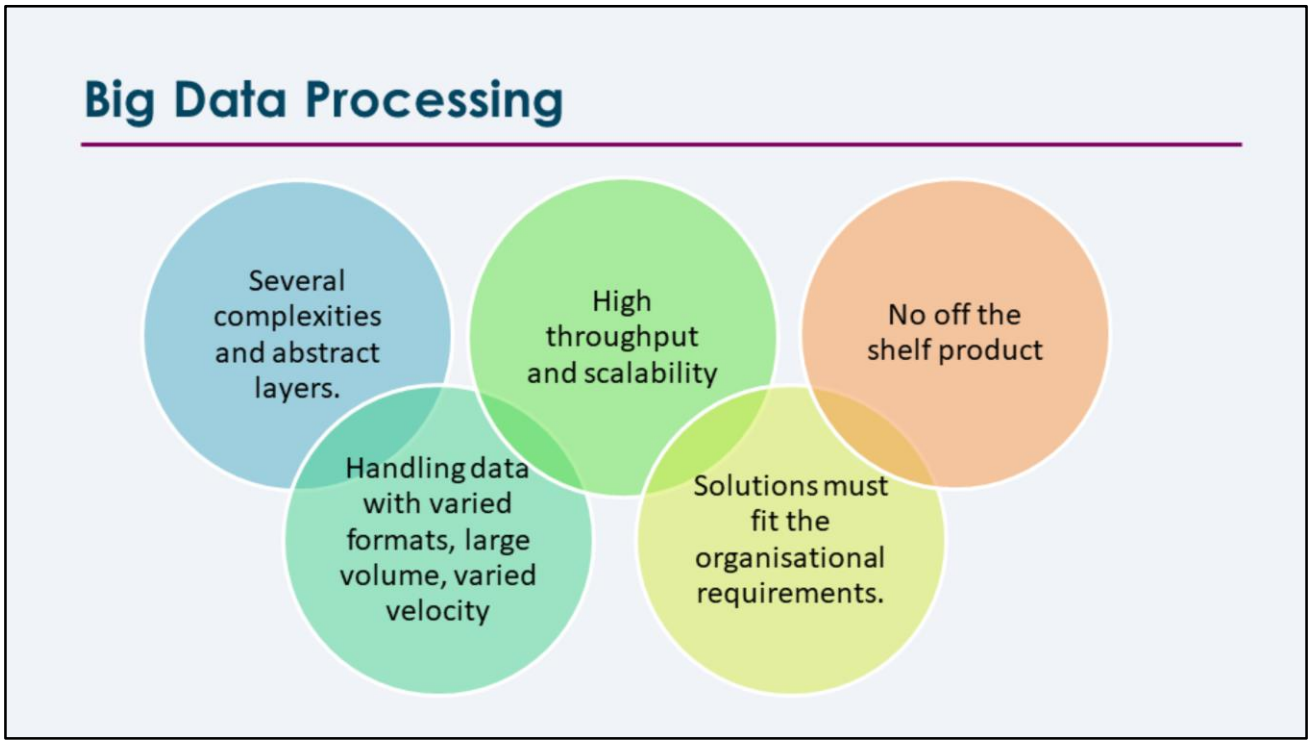
PRIFYSGOL GLYNDŴR WRECSAM | WREXHAM GLYNDŴR UNIVERSITY

CONL 722

Big Data Challenges and Opportunities

1.2.3: Hadoop, HDFS, MapReduce

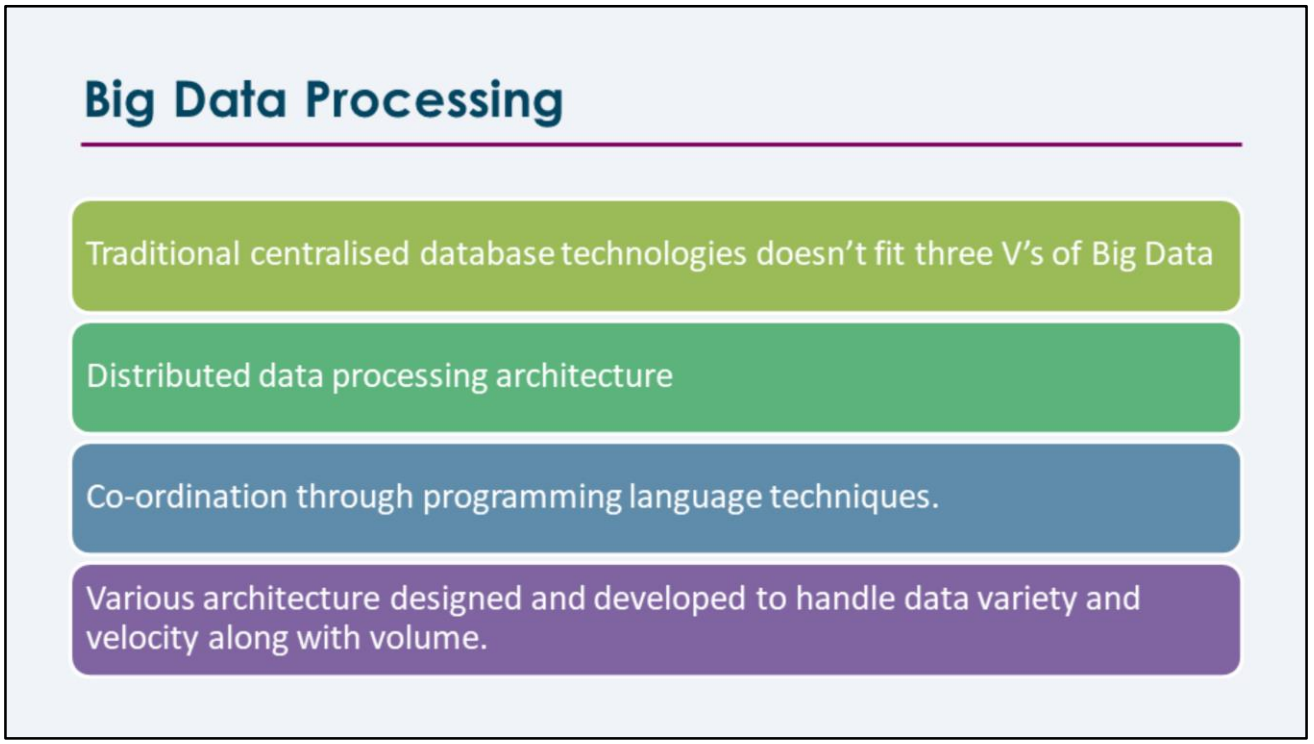
This video will explore the data storage using the distributed file system and provide an overview of Hadoop one of the commonly used big data processing platform.



Big data processing has several complexities and abstract layers.

Solutions must fit the organisational requirements.

Handling data with varied formats, large volume and high velocity providing high throughput and scalability, it is highly unlikely to find an off the shelf product to fit every organisation.



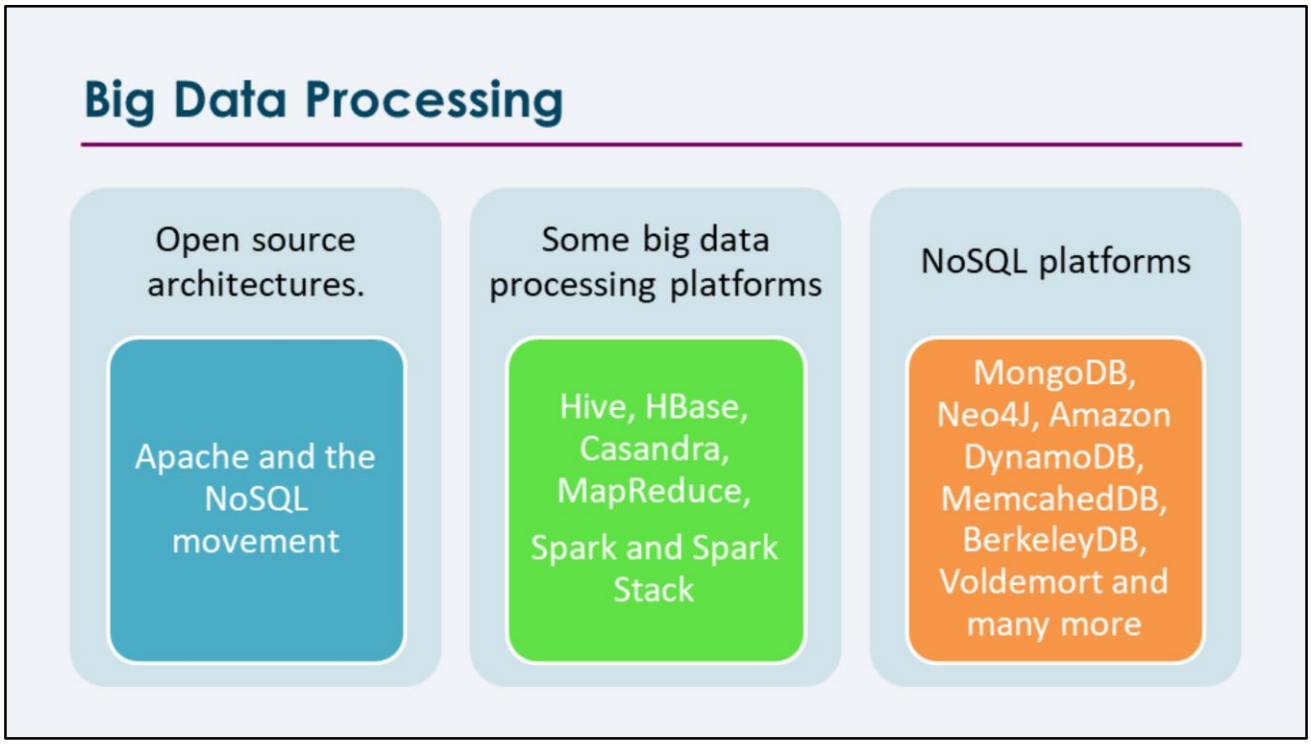
Traditional centralised database technologies don't fit three V's of Big Data.

Big Data processing required to create a distributed data processing architecture and manage the co-ordination through programming language techniques.

This can easily handle the large Volume of data, but not the Variety and Velocity.

Today various architecture designed and developed to handle data variety and velocity along with volume.

They all can be considered while identifying a suitable big data platform for any organisation.



In the past several data processing projects were undertaken, some produced open source architectures. Among those Apache and the NoSQL movement was the most popular.

Some of big data processing platforms are

Hive, HBase, Casandra, MapReduce, Spark, Spark Stack

NoSQL platforms

MongoDB, Neo4J, Amazon DynamoDB, MemcahedDB, BerkeleyDB, Voldemort and many more

Many of these were developed solving data processing needs for Web and Search Engines, but have evolved to support other data processing requirements.

Features

- Extreme Parallel processing
  - Within the system and across multiple systems
- Minimal Database Usage
  - The entire data is not stored in any database, naturally removing the ACID property compliance.
- Distributed file based storage
  - Data stored in files, distributed across systems
- Linearly scalable infrastructure
  - Every component is scalable
- Programmable APIs
  - All modules of data processing will be driven by procedural programming APIs, which allow parallel processing.

Big data processing systems should have the characteristics as listed here.
Expected to have Extreme Parallel processing

Within the system and across multiple systems

It should have Minimal Database Usage,

The entire data is not stored in any database, naturally removing the ACID property compliance.

Data storage rely on Distributed file based storage meaning

Data stored in files, distributed across systems

The system should have a Linearly scalable infrastructure

Every component is scalable

Should use Programmable APIs

All modules of data processing will be driven by procedural programming APIs, which allow parallel processing.

Features

- High speed replication
  - Data is able to replicate at high speed over the network
- High Availability
  - Data and infrastructure are always available and accessible by users.
- Localised processing of data and storage of results
  - Replicated copies of data are required to accomplish localised processing.
- Fault tolerance
  - With extreme replication and distributed processing reduce system failure.

Also,

The system should support high speed replication

Data is able to replicate at high speed over the network

Should ensure high Availability

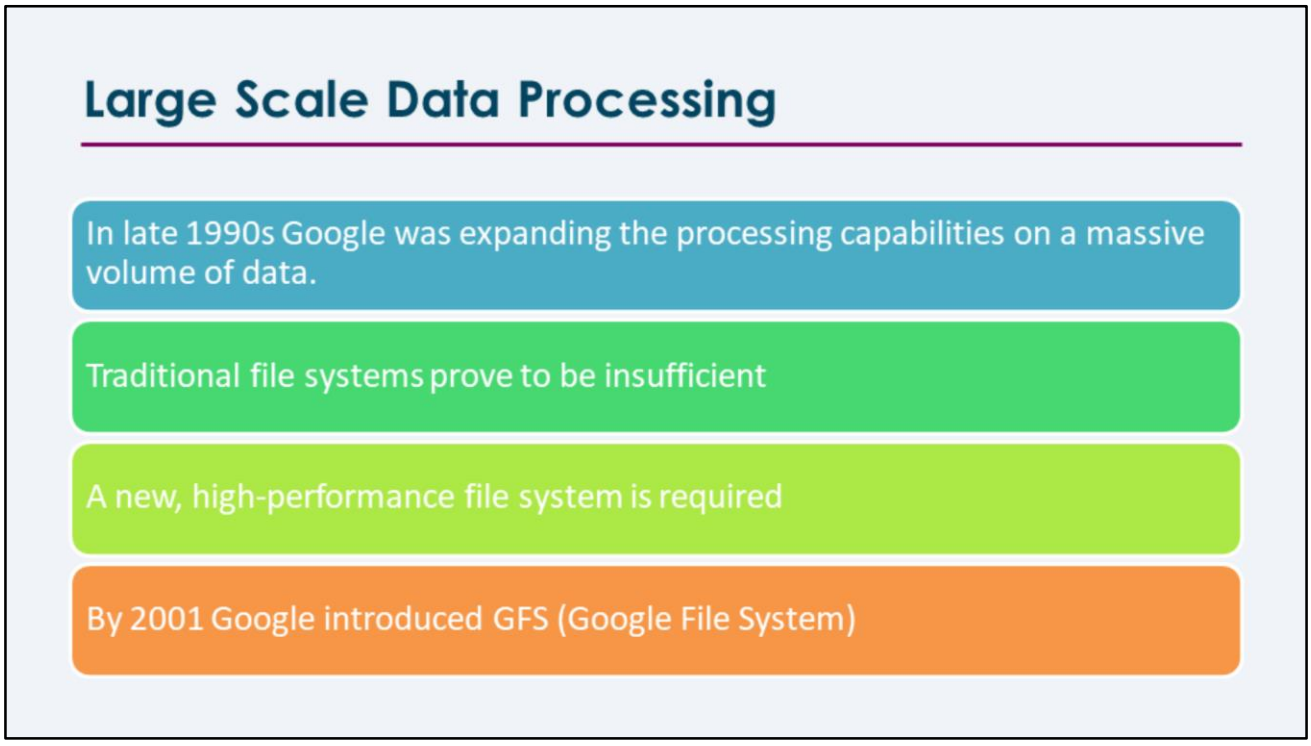
Data and infrastructure are always available and accessible by users.

Should be able to support Localised processing of data and storage of results

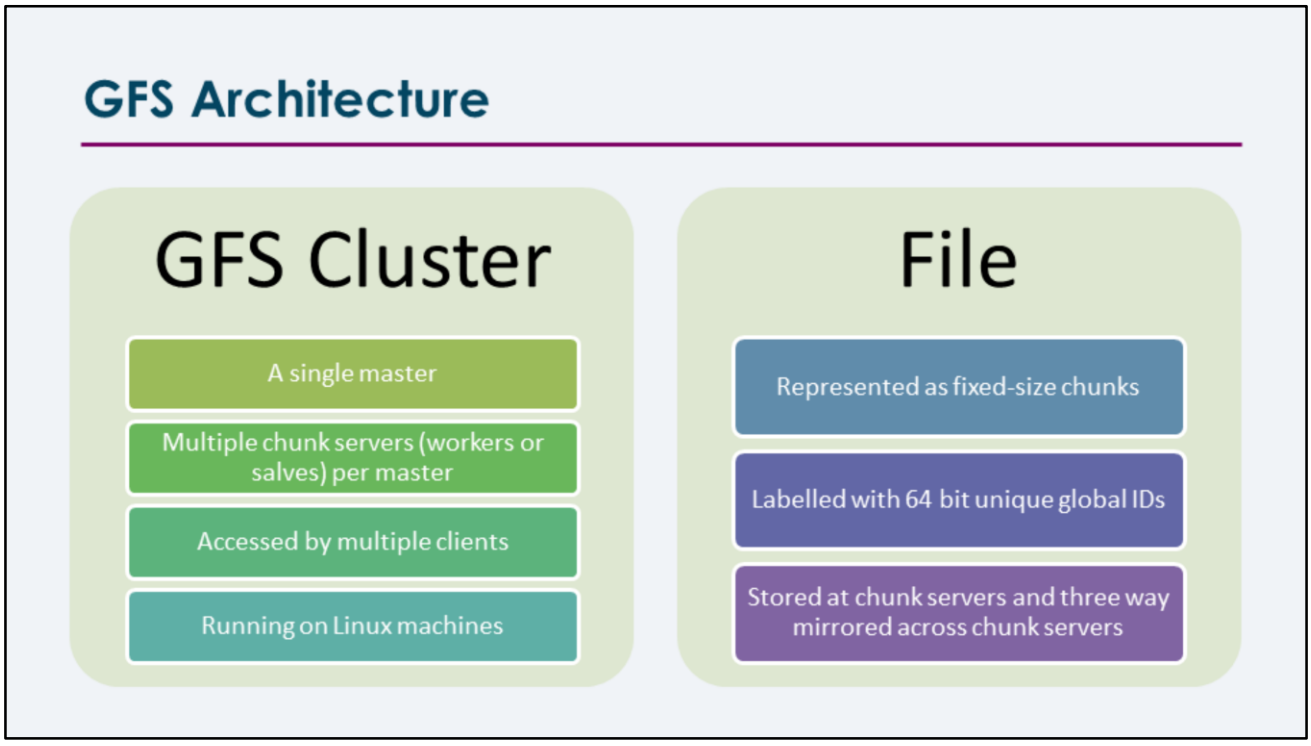
Processing and storing can occur at the same location, but replicated copies of data are required to accomplish localised processing.

Must have extreme Fault tolerance

With extreme replication and distributed processing, system failure could be rebalanced with relative ease and mandated by web users and applications.



In late 1990s Google was expanding the processing capabilities to scale up effectively on a massive volume of data.
Traditional file systems prove to be insufficient
A new, high-performance file system is required
By 2001 Google introduced GFS (Google File System)



Google developed its own data storage using File Storage Cluster

- Each cluster Has a single master
- And will have Multiple chunk servers (workers or salves) per master
- This can be Accessed by multiple clients
- And is Running on Linux machines

Each file is

- Represented as fixed-size chunks,
- Labelled with 64 bit unique global IDs
- Stored at chunk servers and three way mirrored across chunk servers.

GFS Architecture

- There is only one master
- To avoid bottleneck
  - There is minimum communication with master
  - Master communicate the metadata information to the client with the details of the chunk server where the data is currently stored.
  - Client directly contact the chunk server
- Fault Tolerance
  - To avoid the failure due to master node being unavailable the metadata information are replicated across remote nodes

There is only one master in a cluster, bottlenecks must be avoided and the system should ensure fault tolerance.

To avoid any bottleneck:

There will only be minimum communication with master.

Master communicate the metadata information to the client with the details of the chunk server where the data is currently stored.

The client then directly contact the chunk server

Fault Tolerance:

To avoid the failure due to master node being unavailable the metadata information are replicated across remote nodes.

GFS Architecture

- Metadata contains three types of information
  - File and Chunk names or namespaces
  - Mapping from files to chunks (chunks that makes up each file)
  - Location of each chunk replicas.
- Master keeps track of the health of the entire cluster.
  - Handshaking with chunk servers
  - Periodic checksums
- GFS appends the data rather than update

Metadata that is kept in the master and replicated at nodes, contains three types of information

File and Chunk names or namespaces
Mapping from files to chunks (chunks that makes up each file)
Location of each chunk replicas.

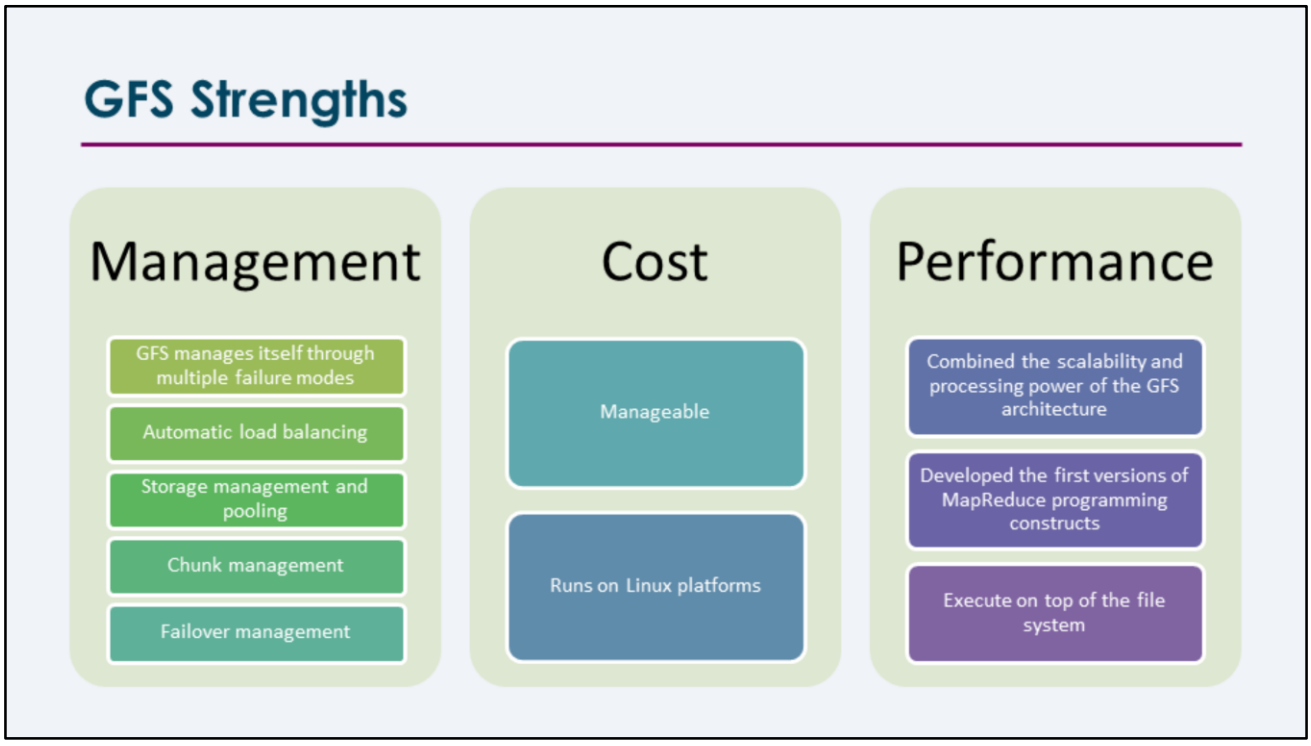
Master keeps track of the health of the entire cluster.

Handshaking with chunk servers
And by Periodic checksums.

GFS appends the data rather than update.

GFS Strengths

- Availability
  - Triple replication-based redundancy
  - Chunk replication
  - Automatic replication management
  - Rapid failovers for any master failure
- Performance
  - Efficient data reads



Google File System has its strength in ensuring availability and fast response.

Availability is guaranteed by

- Triple replication-based redundancy
- Chunk replication
- Automatic replication and load management
- And by Rapid failovers for any master failure

Cost

- Manageable
- Runs on Linux platforms

Performance is managed by efficient data reads.
Google combined the scalability and processing power of the GFS architecture and developed the first versions of MapReduce programming constructs to execute on top of the file system

Hadoop

- Hadoop
  - Cost effective, highly scalable architecture solutions to store and process a large amount of data over a cluster of commodity hardware.
  - Provides new and improved analysis techniques that enable sophisticated analytical processing over multi-structured data.
- Started as an Open source search engine project in 2002 called Nutch
- Developed NDFS (Nutch Distributed File Systems)
  - Based on the architecture concepts of GFS to solve the storage and associated scalability issues
- First generation Hadoop
  - HDFS (modelled after NDFS) distributed file system and MapReduce framework along with a coordinator interface and an interface to write and read from HDFS.
- Today, along with Yahoo other leading companies like IBM, Teradata, Oracle, Microsoft, HP, SAP, DELL etc. also provide solutions in partnership with Hadoop

Hadoop provides cost effective, highly scalable architecture solutions to store and process a large amount of data over a cluster of commodity hardware.

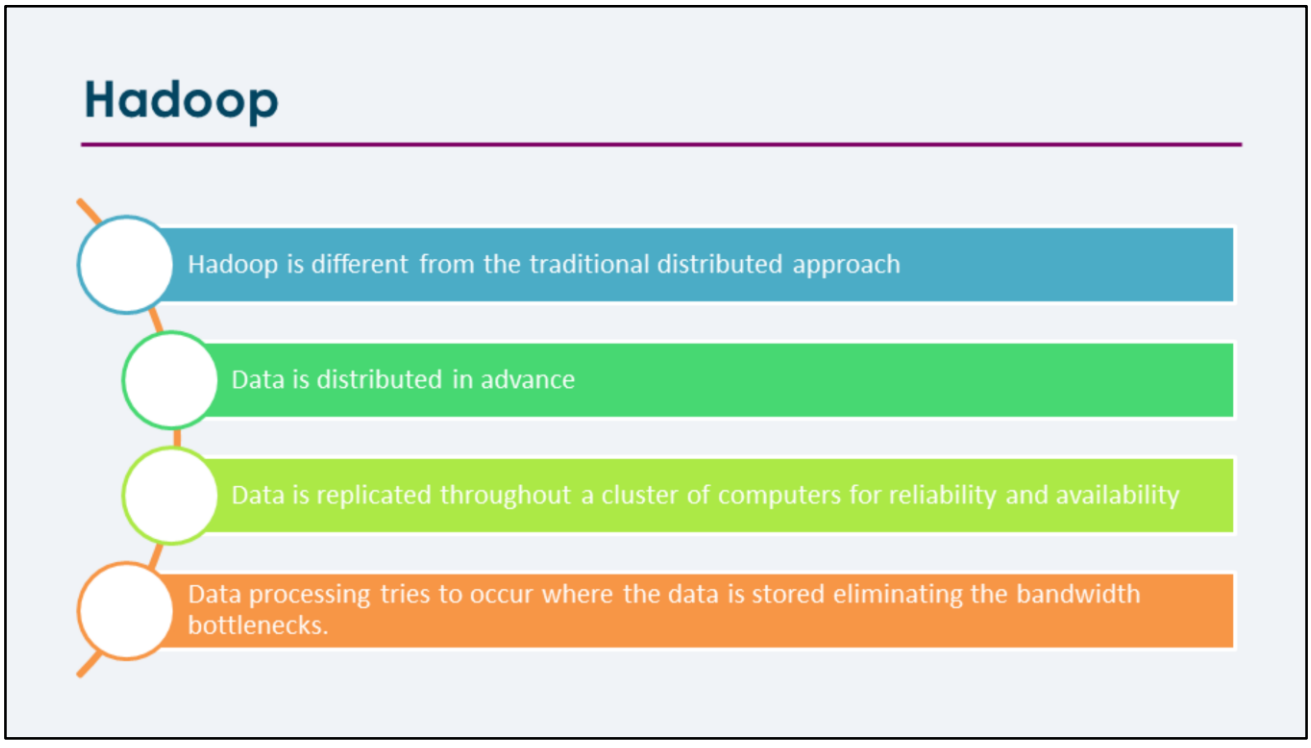
Provides new and improved analysis techniques that enable sophisticated analytical processing over multi-structured data.

Started as an Open source search engine project in 2002 called Nutch.

Developed NDFS (Nutch Distributed File Systems), based on the architecture concepts of GFS to solve the storage and associated scalability issues

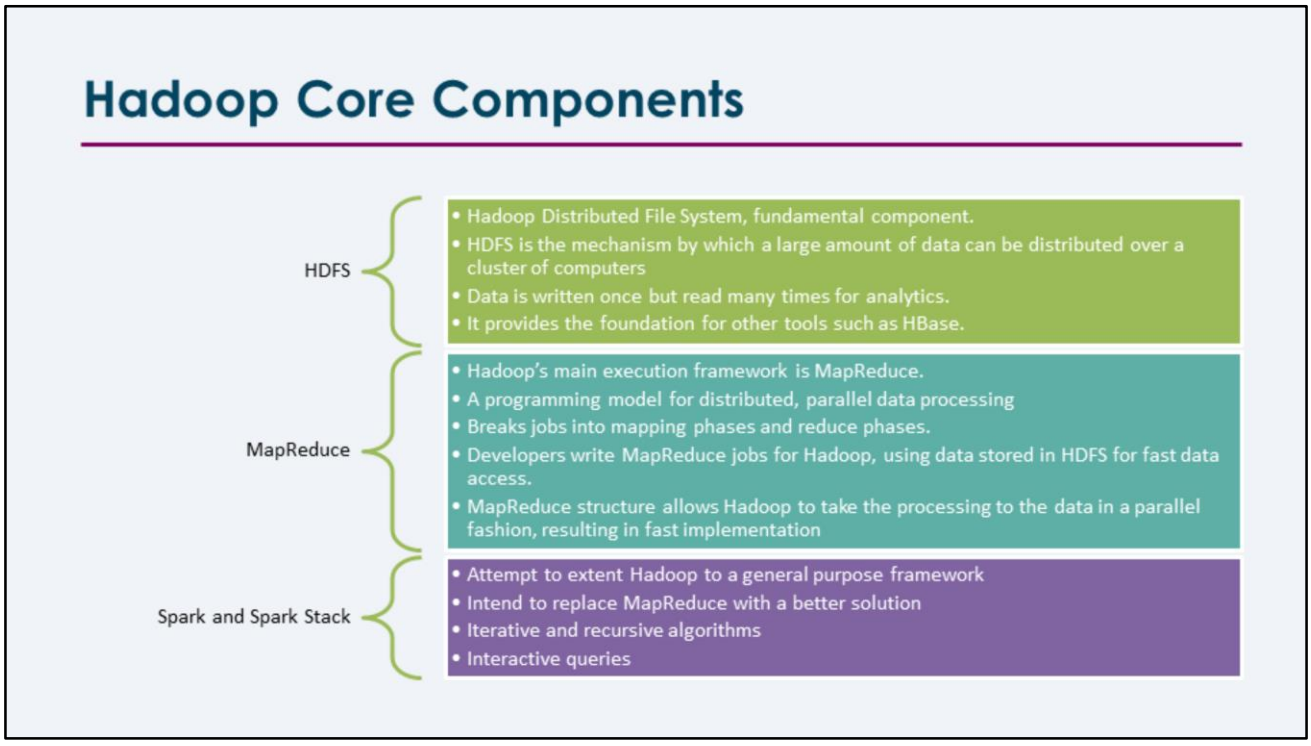
First generation Hadoop consisted of an HDFS (modelled after NDFS) distributed file system and MapReduce framework along with a coordinator interface and an interface to write and read from HDFS.

Today, along with Yahoo other leading companies like IBM, Teradata, Oracle, Microsoft, HP, SAP, DELL etc. also provide solutions in partnership with Hadoop.



Hadoop is different from the traditional distributed approach

- Data is distributed in advance
- Data is replicated throughout a cluster of computers for reliability and availability
- Data processing tries to occur where the data is stored, thus eliminating the bandwidth bottlenecks.



Some of the core components of Hadoop are listed below:

HDFS

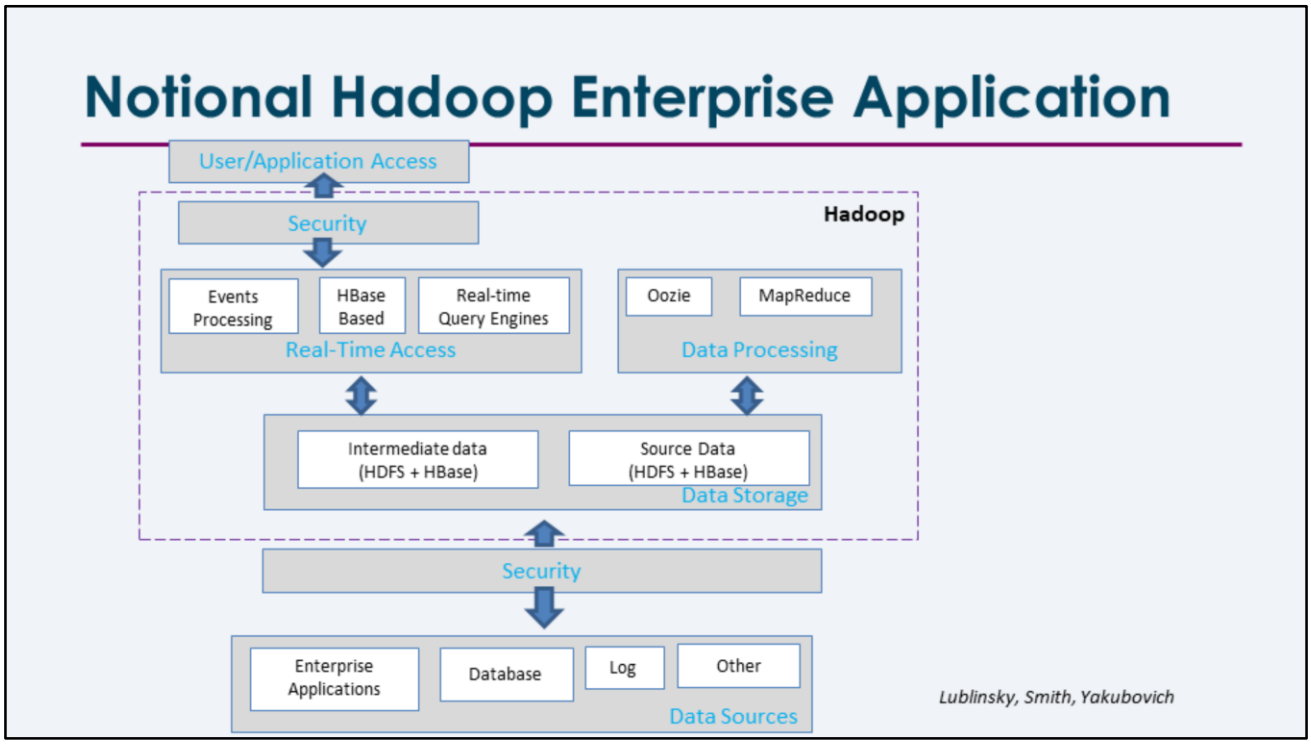
Hadoop Distributed File System, fundamental component.
HDFS is the mechanism by which a large amount of data can be distributed over a cluster of computers
Data is written once but read many times for analytics.
It provides the foundation for other tools such as HBase.

MapReduce

Hadoop's main execution framework is MapReduce.
A programming model for distributed, parallel data processing
Breaks jobs into mapping phases and reduce phases.
Developers write MapReduce jobs for Hadoop, using data stored in HDFS for fast data access.
MapReduce structure allows Hadoop to take the processing to the data in a parallel fashion, resulting in fast implementation

Spark and Spark Stack

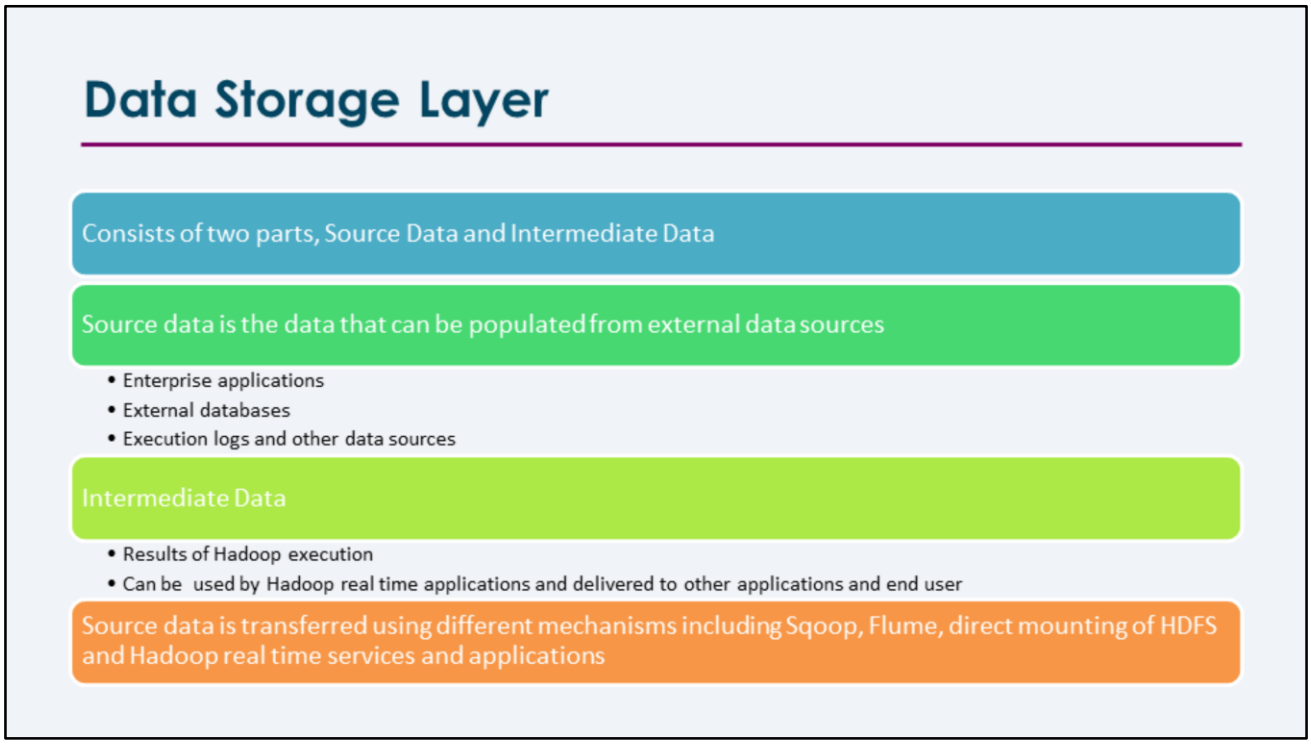
Attempt to extent Hadoop to a general purpose framework
Intend to replace MapReduce with a better solution
Using Iterative and recursive algorithms
And Interactive queries



A typical Hadoop based enterprise application will have the following layers

- Data Storage Layer
- Data Processing layer
- Real Time Access Layer
- Security Layer

Implementation of such architecture requires the understanding of the APIs, their capabilities and limitations, the role of each component in the overall architecture



Data storage layer consists of two parts, Source Data and Intermediate Data
Source data is the data that can be populated from external data sources

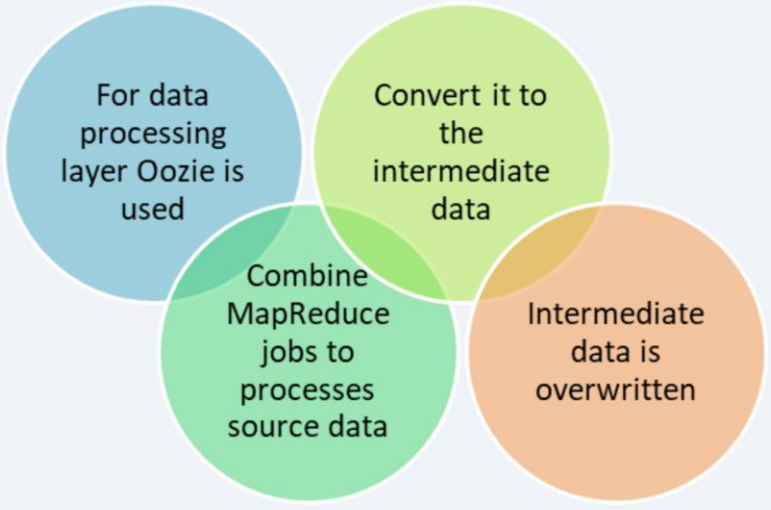
- Enterprise applications
- External databases
- Execution logs and other data sources

Intermediate Data

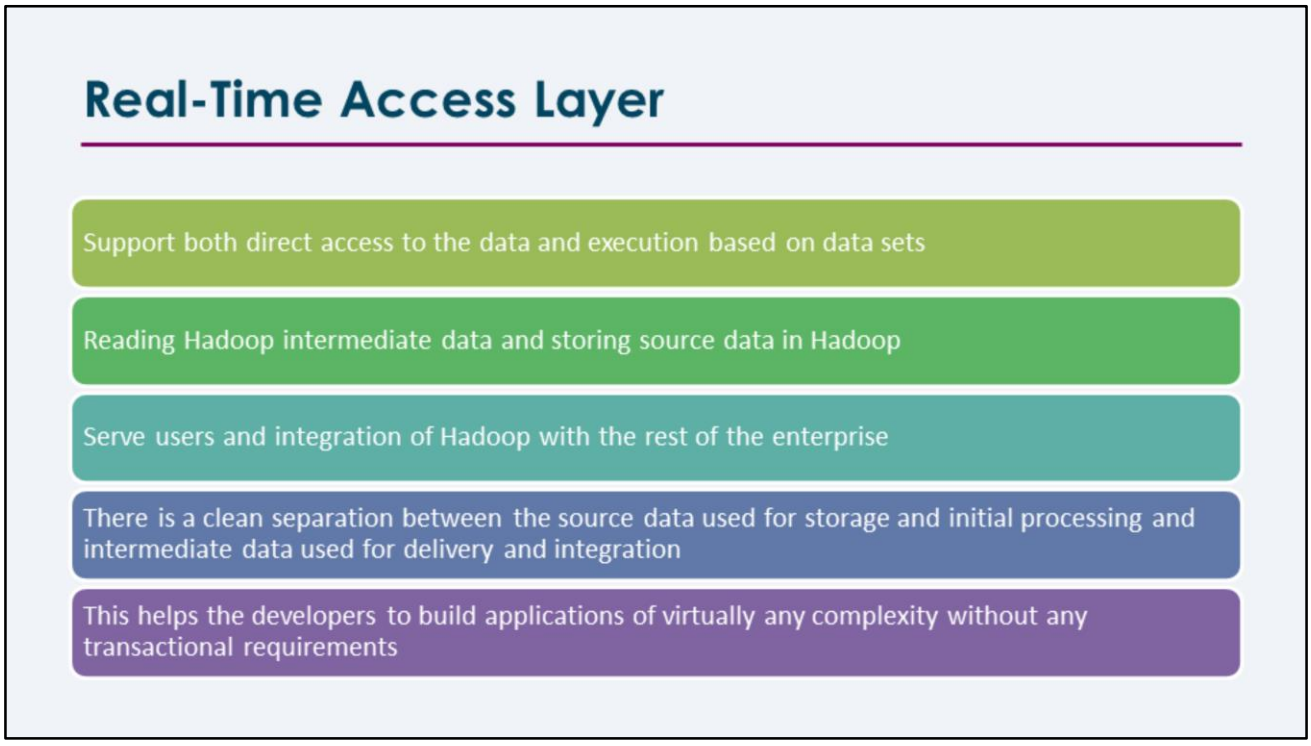
- Results of Hadoop execution
- Can be used by Hadoop real time applications and delivered to other applications and end user

Source data is transferred using different mechanisms including Sqoop, Flume, direct mounting of HDFS and Hadoop real time services and applications.

Data Processing Layer

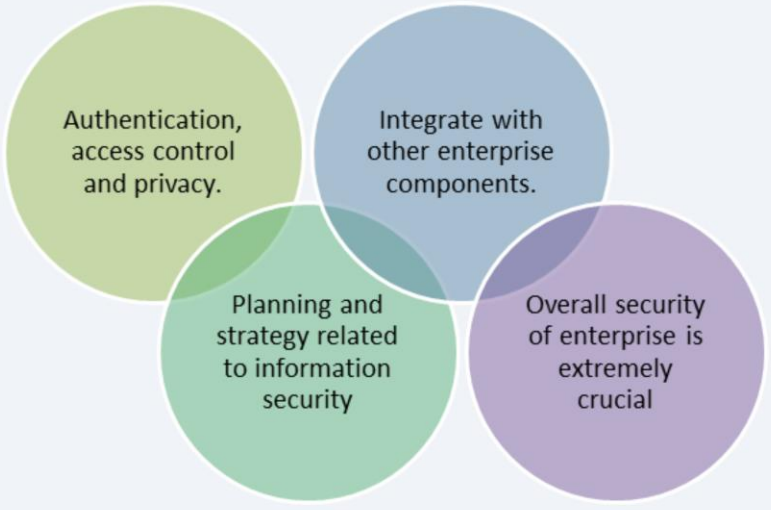


For data processing layer Oozie is used to combine MapReduce jobs to processes source data and convert it to the intermediate data.
Intermediate data is overwritten.



For real time access layer Hadoop real time applications support both direct access to the data and execution based on data sets
Applications can be used for reading Hadoop intermediate data and storing source data in Hadoop
Applications can also be used for serving users and integration of Hadoop with the rest of the enterprise
There is a clean separation between the source data used for storage and initial processing and intermediate data used for delivery and integration
This helps the developers to build applications of virtually any complexity without any transactional requirements.

Security Layer



Manages the authentication, access control and privacy.
Developing enterprise application requires much planning and strategy related to information security
In addition, to secure Hadoop itself, Hadoop implementations often integrate with other enterprise components.
The overall security of enterprise applications is extremely crucial.

Summary

- Big data technologies and architecture
- Database
- Machine learning
- Hadoop and HDFS

During this week you had a brief overview of Big data technologies and architecture, the role of Database, the importance of machine learning and a quick glance into Hadoop and HDFS. This will lay the foundation for the coming weeks.

Please remember to investigate further into various big data technologies, compare them according to their capabilities. Post your findings on the discussion forum and comment on your peers' view with justification.

Next week will be focusing on data types and objects, looking at structured and unstructured data.