Master en

Big Data e Inteligencia Artificial

Module 6

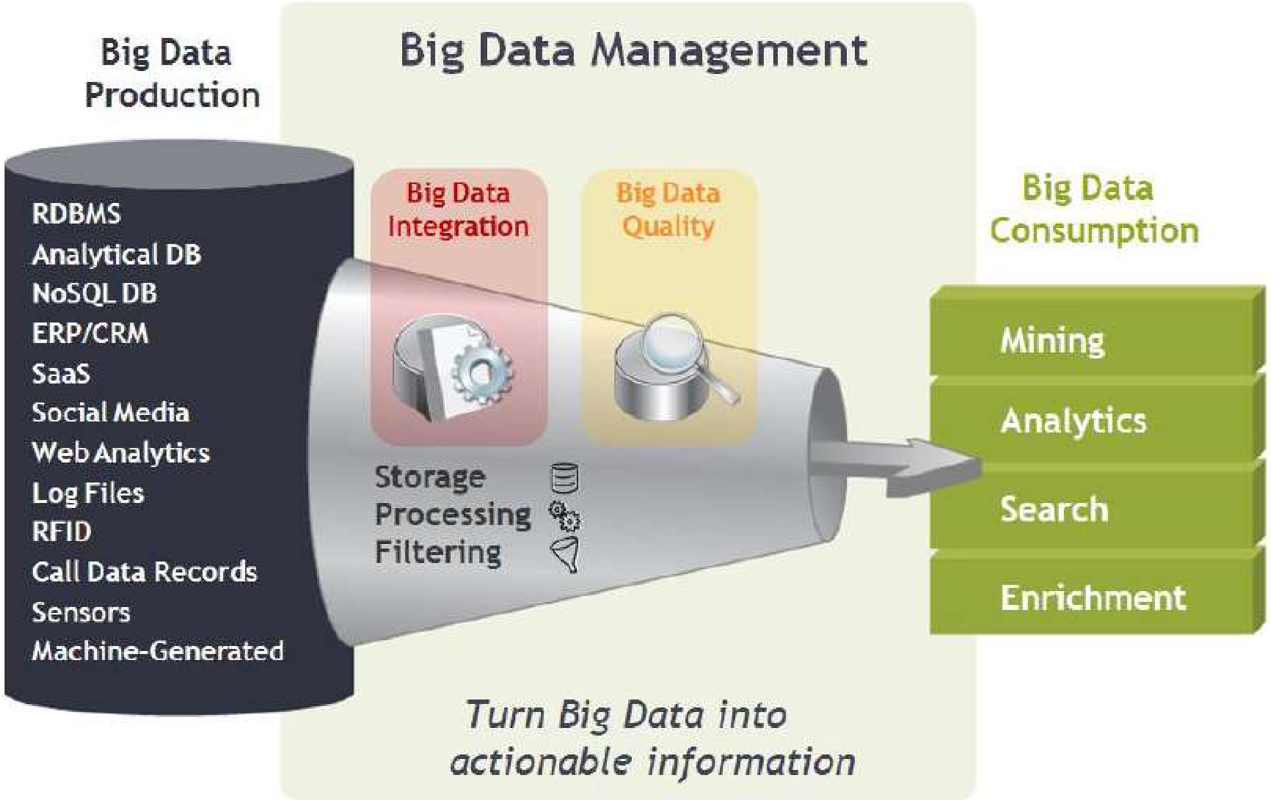
Introduction to Apache Hadoop and HDFS

Cristóbal Barba González– University of Málaga
Antonio J. Nebro – University of Málaga

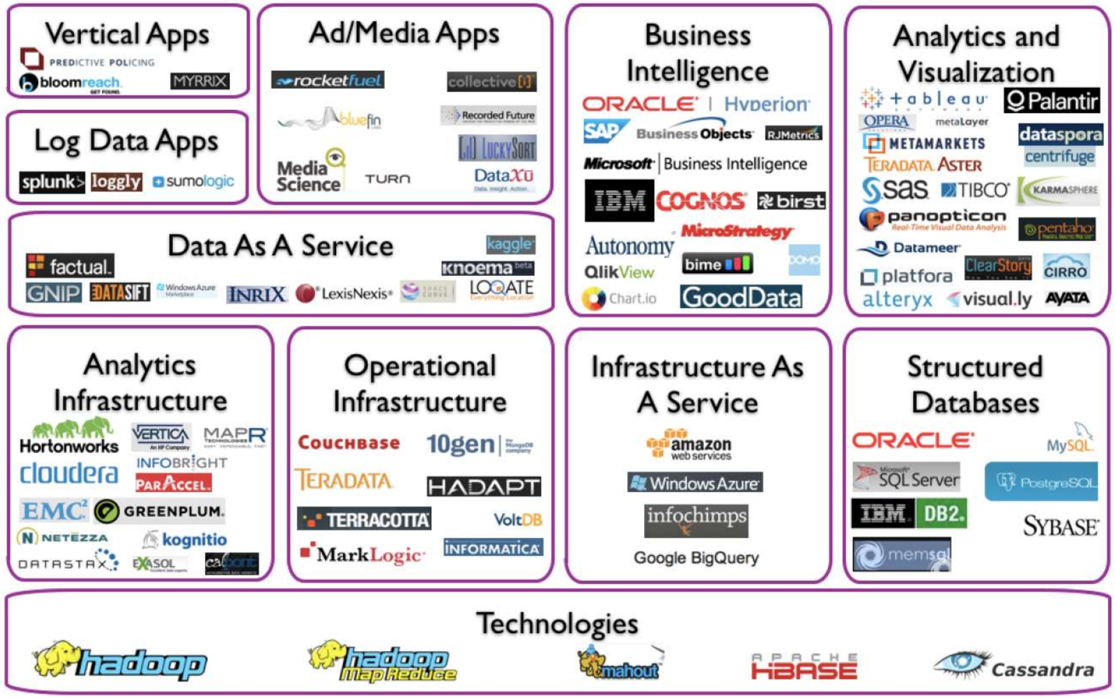
Table of contents

- Big Data.
- Introduction to Apache Hadoop.
- Hadoop distribution.
- Hadoop cloud.

Big Data Management

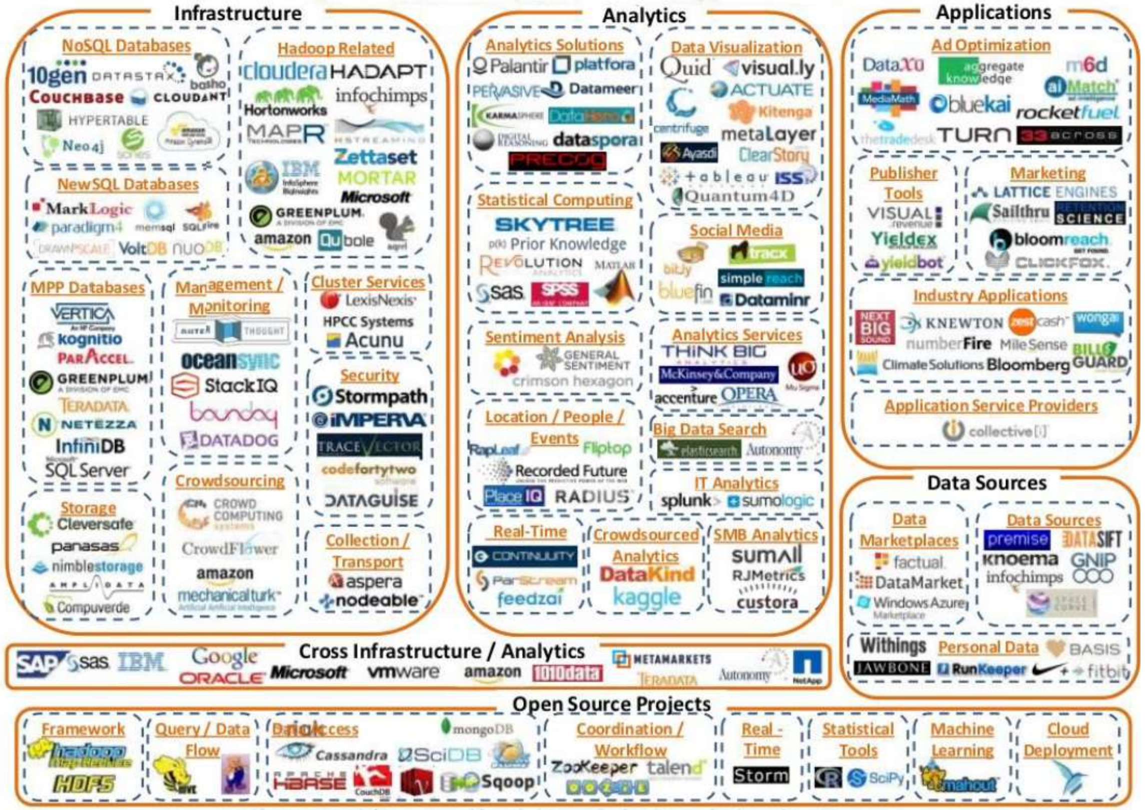


Big Data Landscape



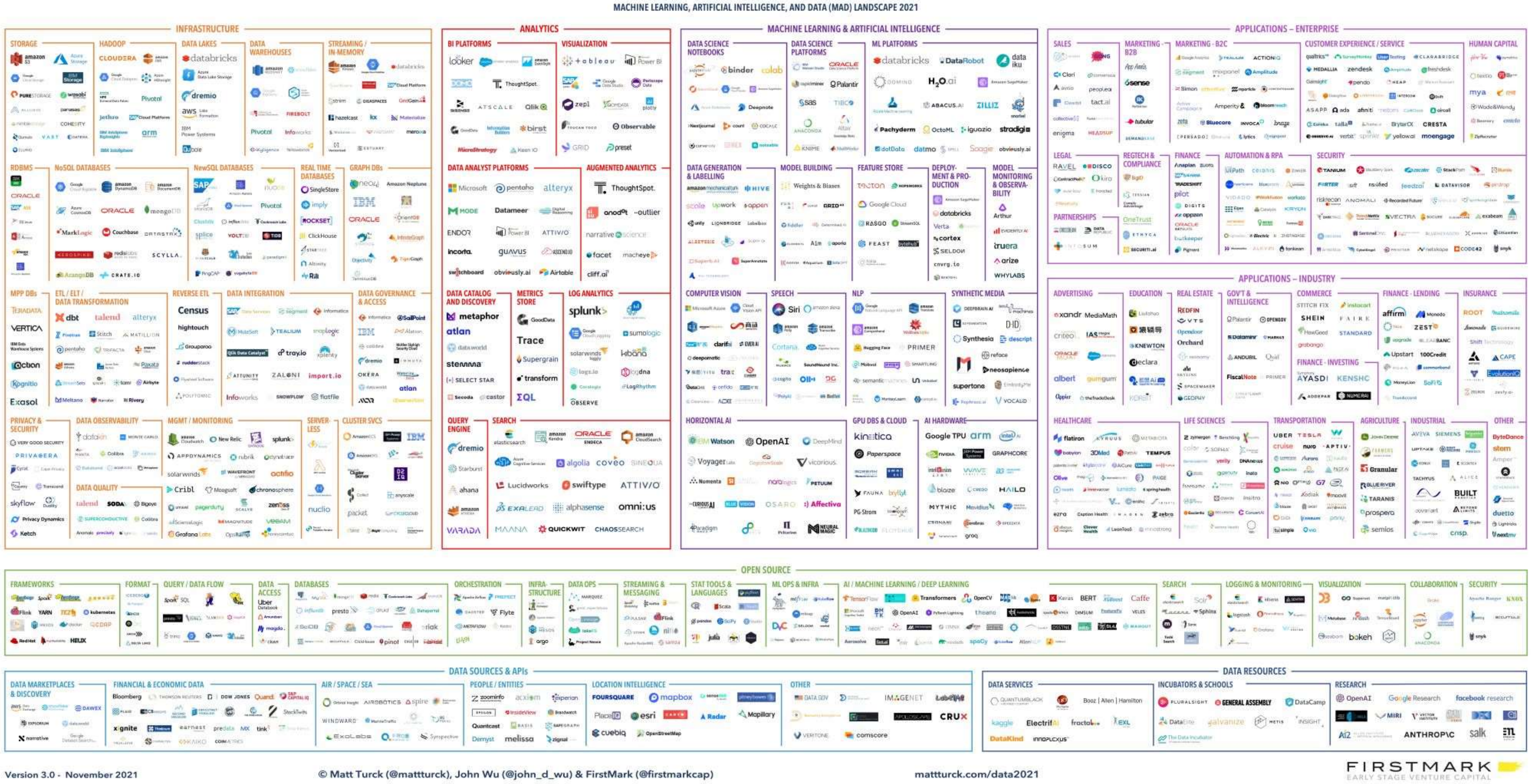
Copyright © 2012 Dave Feinleib dave@vcdave.com blogs.forbes.com/davefeinleib

Big Data Landscape

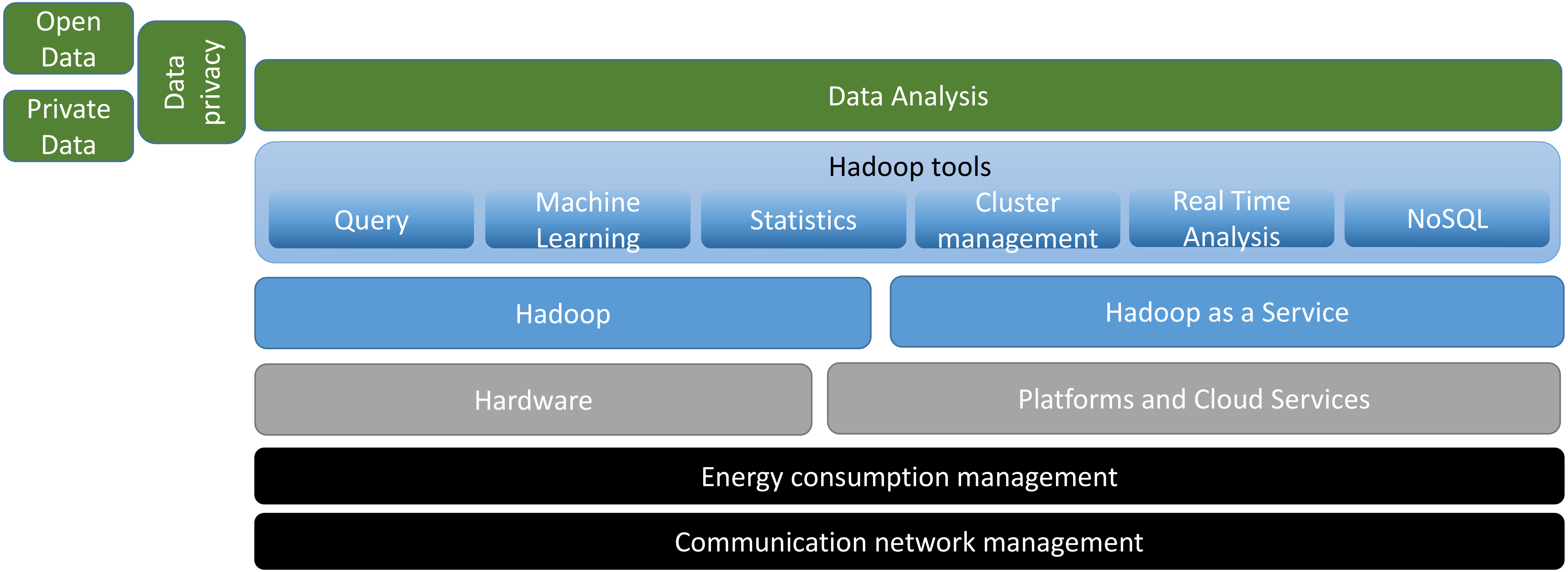


© Matt Turck (@mattturck) and ShivonZilis (@shivonz) Bloomberg Ventures

Big Data Landscape



Global Vision



Open Data

Private Data

Data privacy

Data Analysis

Hadoop tools

Query

Machine Learning

Statistics

Cluster management

Real Time Analysis

NoSQL

Hadoop

Hadoop as a Service

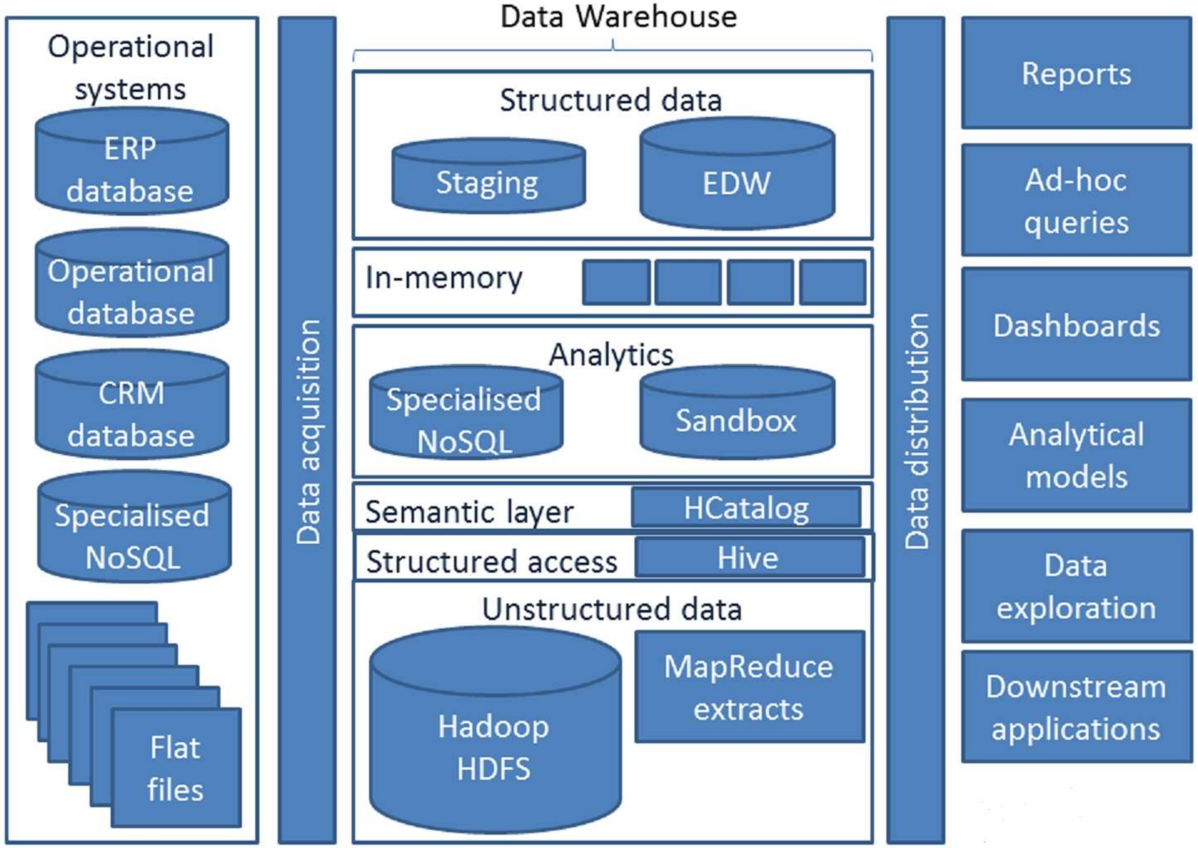
Hardware

Platforms and Cloud Services

Energy consumption management

Communication network management

Big Data Business Intelligence

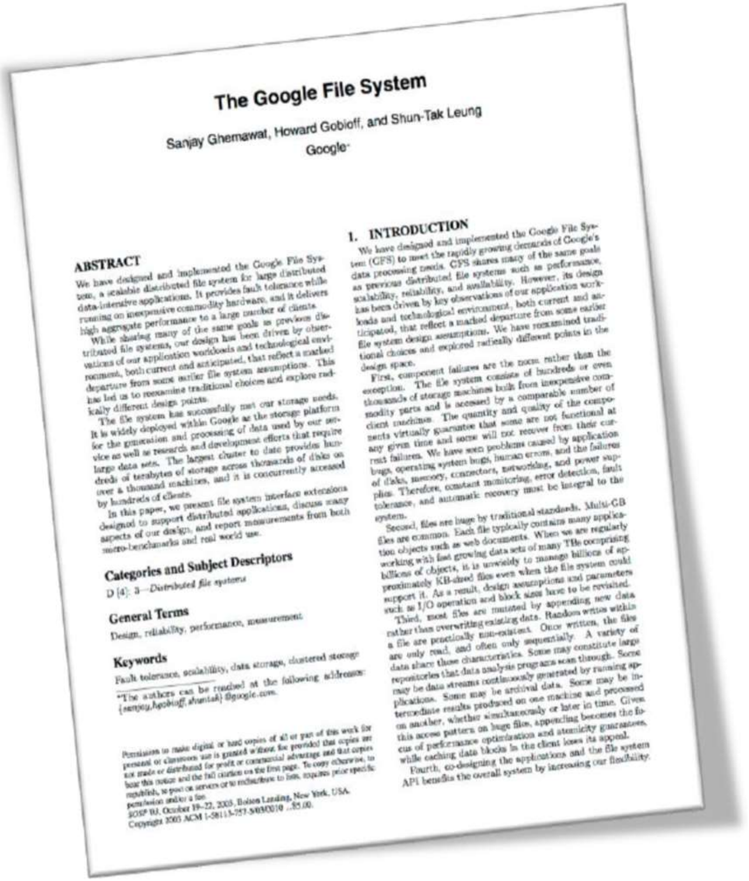


Massive Data Storage

- On a PC, we store data on our hard drive.
- The data can be of any type:
  - Tables, text files, web pages...
  - ...images, videos,
- BIG DATA: What happens when the data does not fit on a hard drive?

The Google Approach

- In 2003, Google presents Google file system
- It is a distributed file system. It allows:
  - Store large amounts of data.
  - Distribute (partition) the data among several computers.
  - Allow more efficient file readings.
  - Prevent data loss if a computer crashes spoils.



The Google File System

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

ABSTRACT

We have designed and implemented the Google File System, a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients.

While sharing many of the same goals as previous distributed file systems, our design has been driven by observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system assumptions. This has led us to reexamine traditional choices and explore radically different design points.

The file system has successfully met our storage needs. It is widely deployed within Google as the storage platform for the generation and processing of data used by our service as well as research and development efforts that require large data sets. The largest cluster to date provides hundreds of terabytes of storage across thousands of disks on over a thousand machines, and it is concurrently accessed by hundreds of clients.

In this paper, we present file system interface extensions designed to support distributed applications, discuss many aspects of our design, and report measurements from both micro-benchmarks and real world use.

Categories and Subject Descriptors

D [4]: 3—*Distributed file systems*

General Terms

Design, reliability, performance, measurement

Keywords

Fault tolerance, scalability, data storage, clustered storage

*The authors can be reached at the following addresses: {sanjay,hgobioff,shuntak}@google.com.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SOSP'03, October 19–22, 2003, Bolton Landing, New York, USA.
Copyright 2003 ACM 1-58113-757-5/03/0010 ...$5.00.

1. INTRODUCTION

We have designed and implemented the Google File System (GFS) to meet the rapidly growing demands of Google's data processing needs. GFS shares many of the same goals as previous distributed file systems such as performance, scalability, reliability, and availability. However, its design has been driven by key observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system design assumptions. We have reexamined traditional choices and explored radically different points in the design space.

First, component failures are the norm rather than the exception. The file system consists of hundreds or even thousands of storage machines built from inexpensive commodity parts and is accessed by a comparable number of client machines. The quantity and quality of the components virtually guarantee that some are not functional at any given time and some will not recover from their current failures. We have seen problems caused by application bugs, operating system bugs, human errors, and the failures of disks, memory, connectors, networking, and power supplies. Therefore, constant monitoring, error detection, fault tolerance, and automatic recovery must be integral to the system.

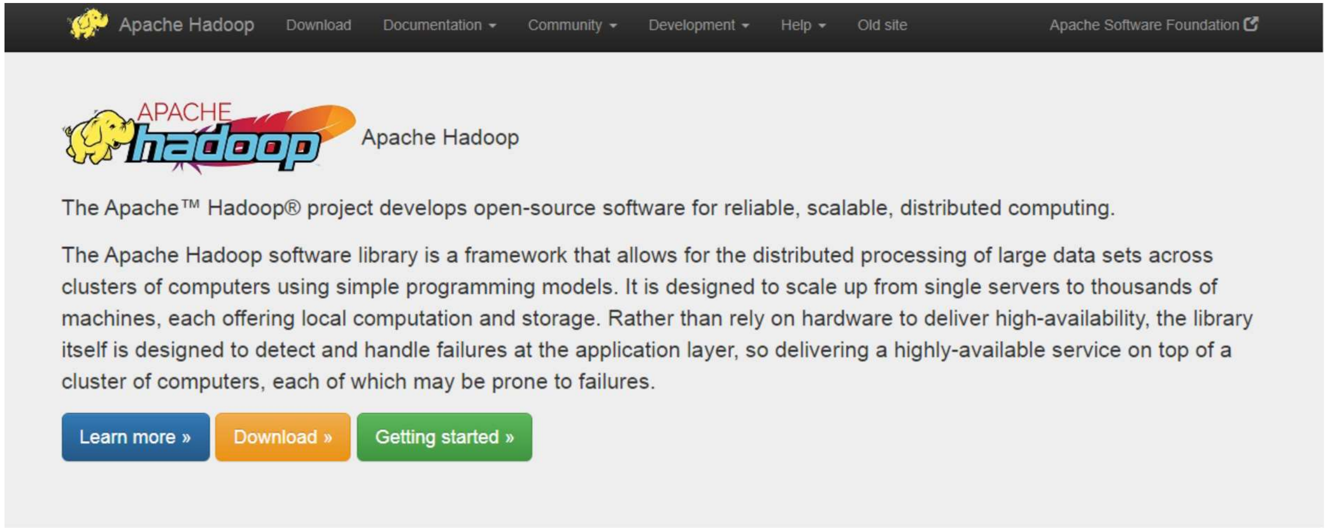
Second, files are huge by traditional standards. Multi-GB files are common. Each file typically contains many application objects such as web documents. When we are regularly working with fast growing data sets of many TBs comprising billions of objects, it is unwieldy to manage billions of approximately KB-sized files even when the file system could support it. As a result, design assumptions and parameters such as I/O operation and block sizes have to be revisited.

Third, most files are mutated by appending new data rather than overwriting existing data. Random writes within a file are practically non-existent. Once written, the files are only read, and often only sequentially. A variety of data share these characteristics. Some may constitute large repositories that data analysis programs scan through. Some may be data streams continuously generated by running applications. Some may be archival data. Some may be intermediate results produced on one machine and processed on another, whether simultaneously or later in time. Given this access pattern on huge files, appending becomes the focus of performance optimization and atomicity guarantees, while caching data blocks in the client loses its appeal.

Fourth, co-designing the applications and the file system API benefits the overall system by increasing our flexibility.

Apache Hadoop

- **Hadoop** is a framework that allows for the distributed processing of large data sets across clusters.
  - Performance.
  - Storage.
  - Scalability.
  - Fault tolerance.
  - Cost efficiency.



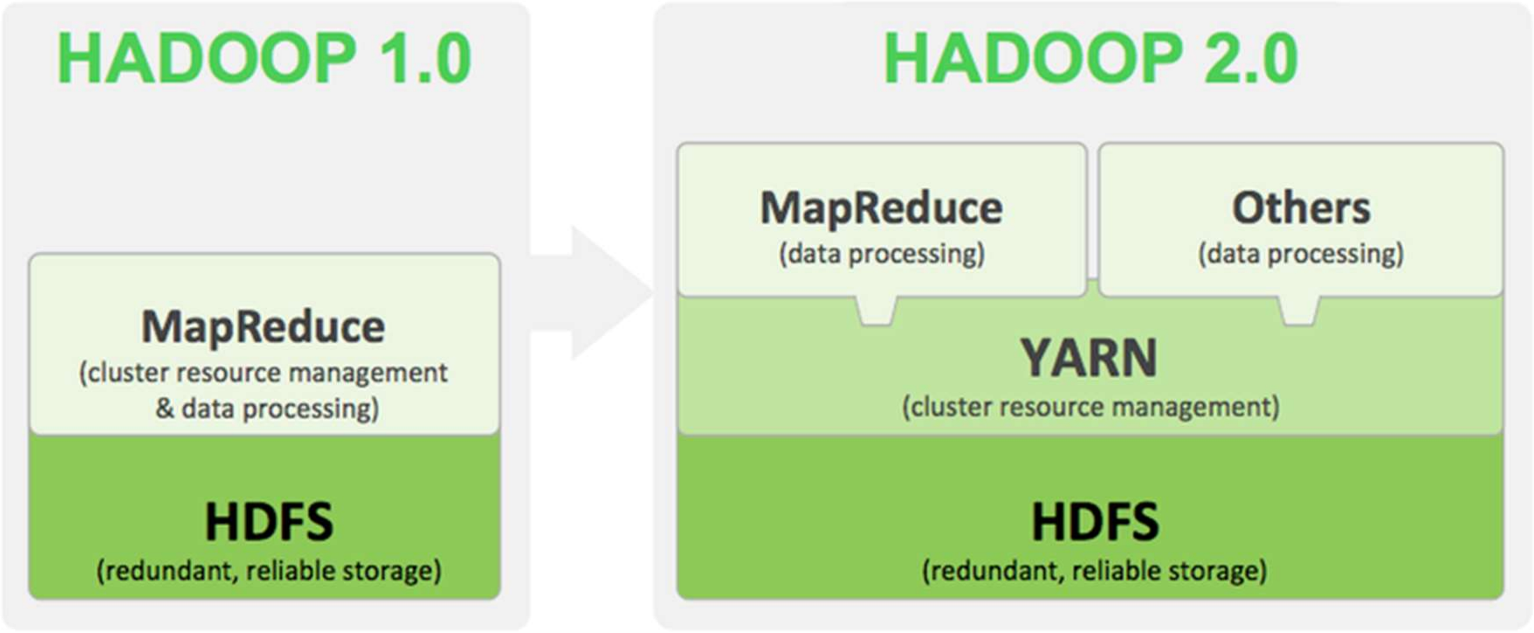
Apache Hadoop

The Apache™ Hadoop® project develops open-source software for reliable, scalable, distributed computing.

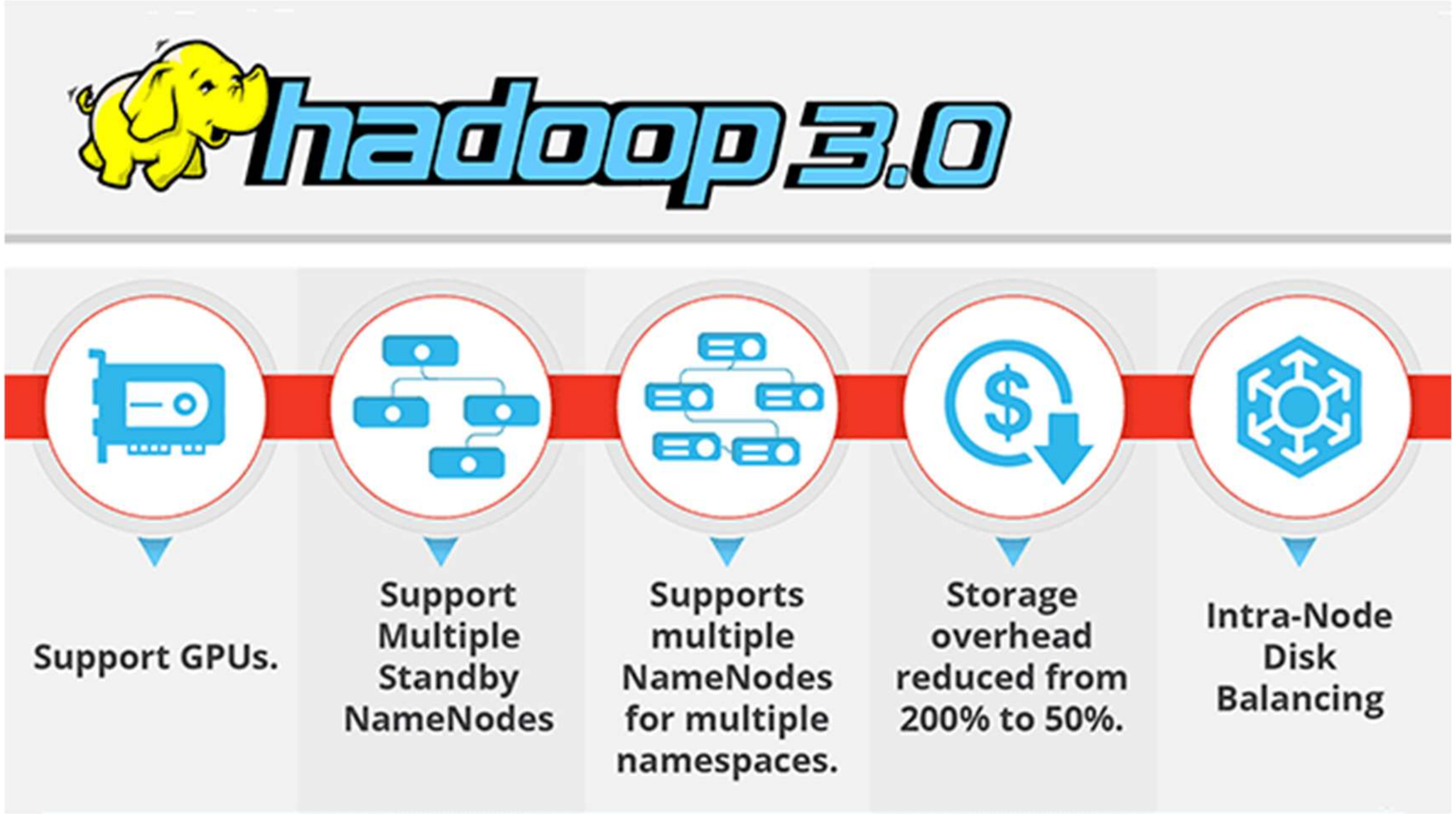
The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

https://hadoop.apache.org/

Apache Hadoop


HADOOP 1.0
MapReduce
(cluster resource management & data processing)
HDFS
(redundant, reliable storage)
HADOOP 2.0
MapReduce
(data processing)
Others
(data processing)
YARN
(cluster resource management)
HDFS
(redundant, reliable storage)

Apache Hadoop



Support GPUs.

Support Multiple Standby NameNodes

Supports multiple NameNodes for multiple namespaces.

Storage overhead reduced from 200% to 50%.

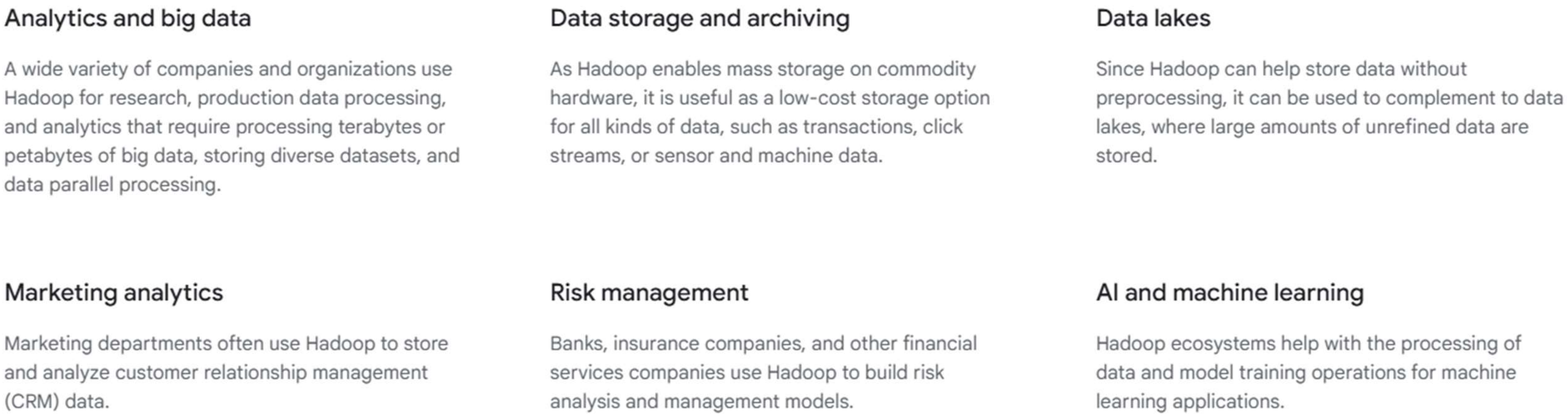
Intra-Node Disk Balancing

Apache Hadoop

- When to use Hadoop?
  - For processing really Big Data.
  - For storing a diverse set of data.
  - For parallel data processing.

- When not to use Hadoop?
  - For real-time data analysis.
  - For replacement relational database system.
  - For multiple smaller datasets.

What is Apache Hadoop used for?



Analytics and big data

A wide variety of companies and organizations use Hadoop for research, production data processing, and analytics that require processing terabytes or petabytes of big data, storing diverse datasets, and data parallel processing.

Data storage and archiving

As Hadoop enables mass storage on commodity hardware, it is useful as a low-cost storage option for all kinds of data, such as transactions, click streams, or sensor and machine data.

Data lakes

Since Hadoop can help store data without preprocessing, it can be used to complement to data lakes, where large amounts of unrefined data are stored.

Marketing analytics

Marketing departments often use Hadoop to store and analyze customer relationship management (CRM) data.

Risk management

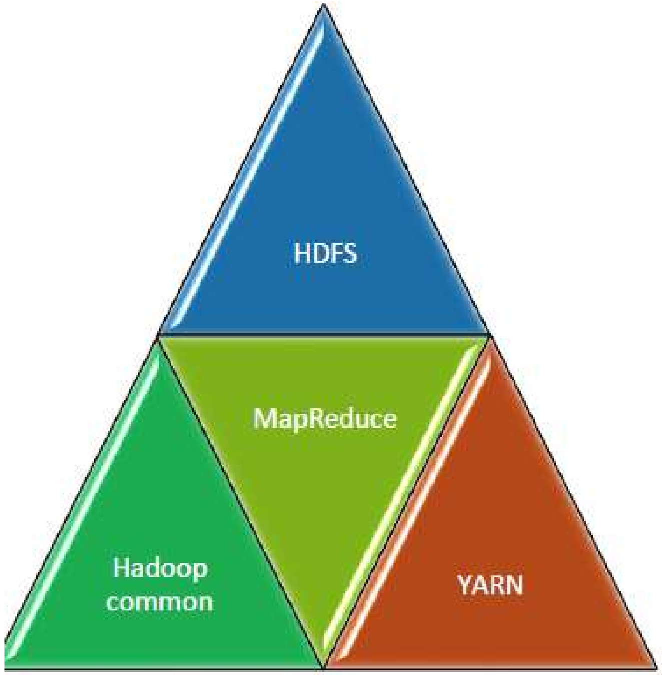
Banks, insurance companies, and other financial services companies use Hadoop to build risk analysis and management models.

AI and machine learning

Hadoop ecosystems help with the processing of data and model training operations for machine learning applications.

Apache Hadoop

- Basic components of Hadoop
  - **Hadoop HDFS** - Hadoop Distributed File System (HDFS) is the storage unit of Hadoop.
  - **Hadoop MapReduce** - Hadoop MapReduce is the processing unit of Hadoop.
  - **Hadoop YARN** - Hadoop YARN is a resource management unit of Hadoop.
  - **Hadoop Common –** Haddop common is a collection of common libraries and utilities that work with different Hadoop modules.



Hadoop HDFS

- HDFS has a master/worker(slave) architecture. An HDFS cluster consists of one or more **NameNodes**
  - NameNode is a master server that manages the file system namespace and regulates access to files by clients Simple Coherency Model.
  - HDFS applications need a write-once-read-many access model for files. A file once created, written, and closed need not be changed.
  - The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes and their replications.

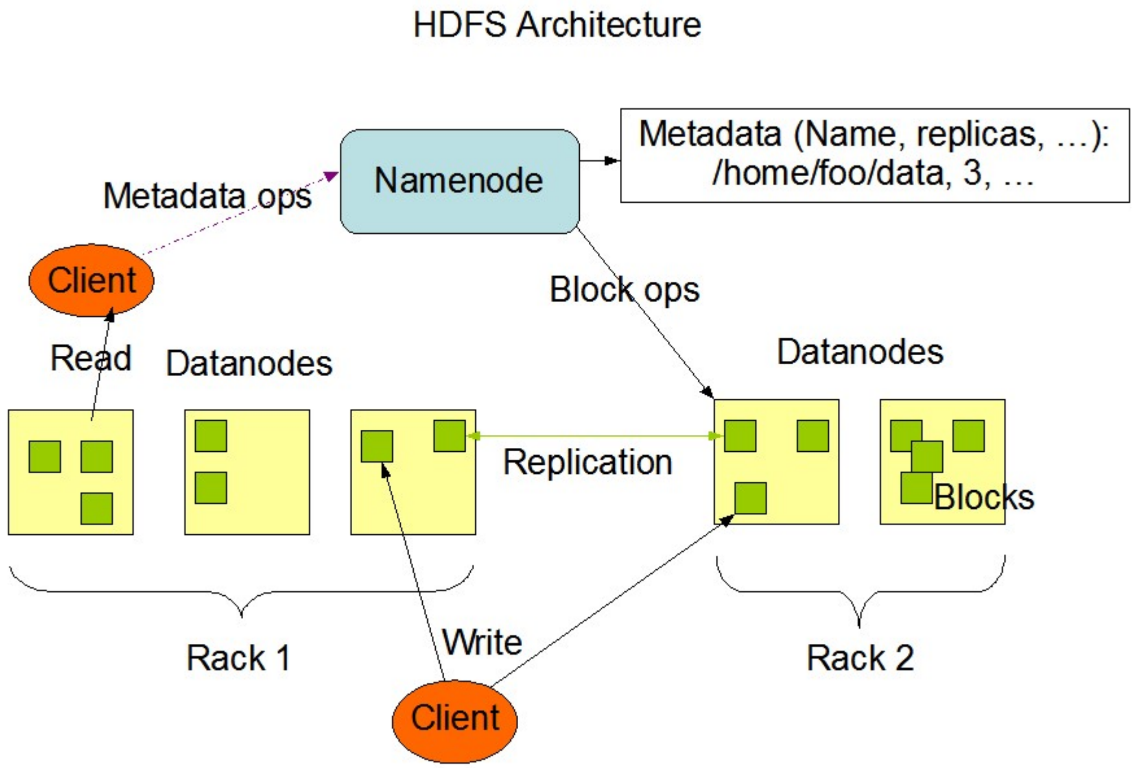
Hadoop HDFS

- There are a number of **DataNodes**, usually one per node in the cluster.
  - DataNodes manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files.
  - The DataNodes are responsible for serving read and write requests from the file system’s clients.
  - The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.

Hadoop HDFS

- Main features
  - Based on commodity hardware (not special hardware is required).
  - Fault tolerant and self healing.
  - Developed specifically for large scale data processing workloads where scalability, flexibility and throughput are critical.
  - Load Balancing: Place data intelligently for maximum efficiency and utilization.
  - Tunable Replication: Multiple copies of each file provide data protection and computational performance.
  - Based on "Moving Computation is Cheaper than Moving Data“.
  - Master/ slave architecture.
  - Written in Java.
  - Implemented in Linux.
  - TCP communication protocol.

Hadoop HDFS

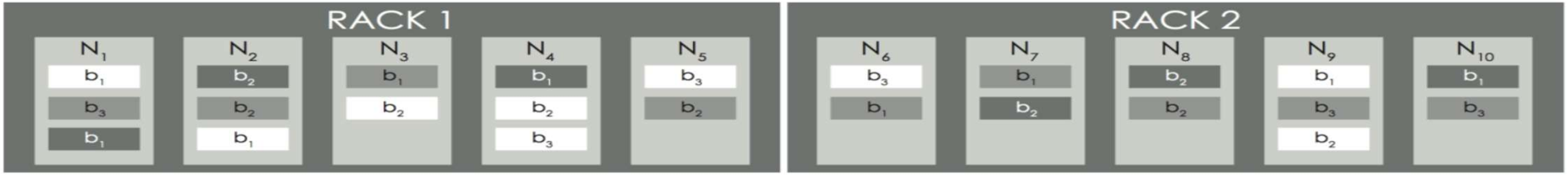

HDFS Architecture
Metadata ops
Namenode
Metadata (Name, replicas, …):
/home/foo/data, 3, …
Client
Block ops
Read
Datanodes
Datanodes
Replication
Blocks
Rack 1
Write
Rack 2
Client

Hadoop HDFS

- Each file is divided into blocks of 64 or 128 MB.
- The blocks are stored divided between several computers.
- Each computer is called a node.
- The set of computers is called a cluster.
- ADVANTAGE 1: If you double the number of nodes, you can fit twice as much data.
- ADVANTAGE 2: You can store files that do not fit on a computer.
- ADVANTAGE 3: Readings are faster: you can read a file from several computers at the same time.

Hadoop HDFS

- PROBLEM: If there are many nodes, the probability that one fails is high.
- SOLUTION: Each block is stored in several nodes, by default 3.
- Each block stored on a node is called a replica.
- Replicas are topology-aware: better to store them in different racks* or data centers.



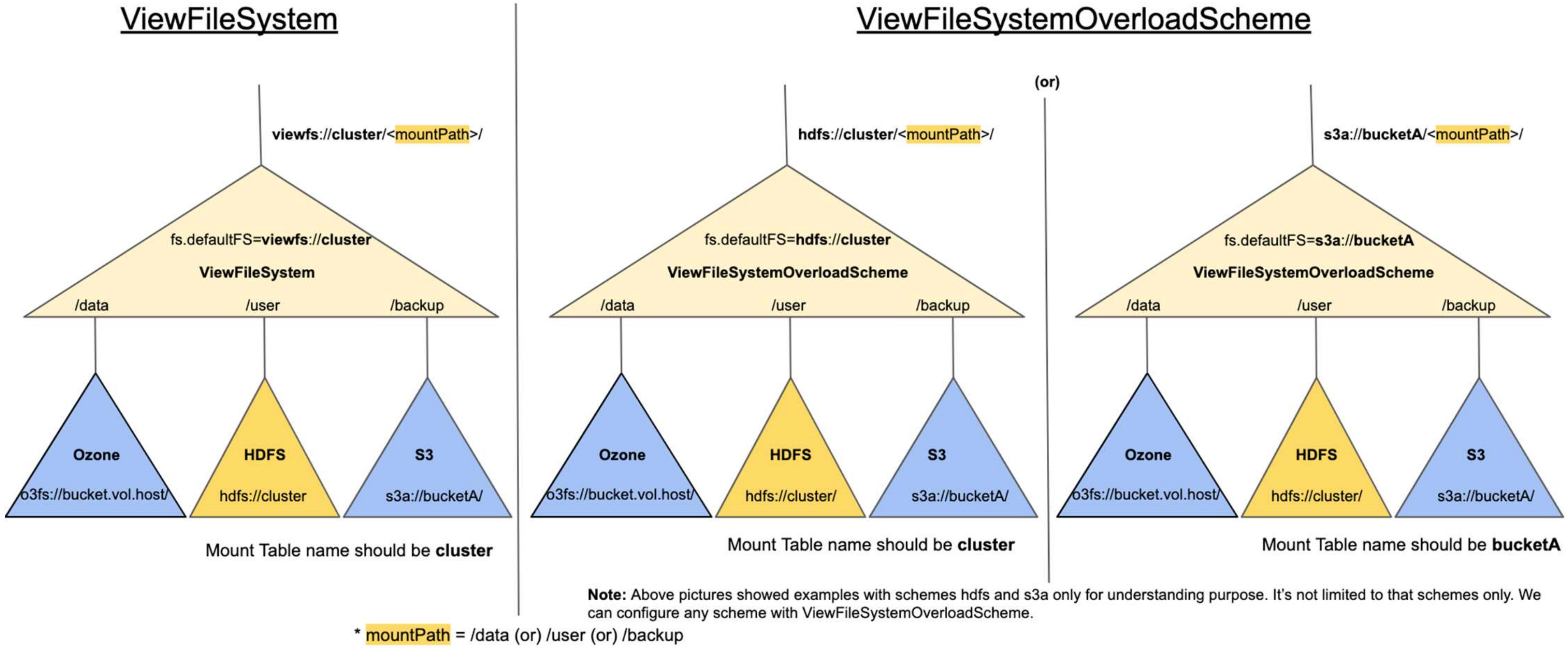
* A rack or cabinet is a container where several nodes are stored that share a power and network system

Hadoop HDFS

- The File System (FS) shell includes various shell-like commands that directly interact with the Hadoop Distributed File System (HDFS) as well as other file systems that Hadoop supports, such as Local FS, WebHDFS, S3 FS, and others

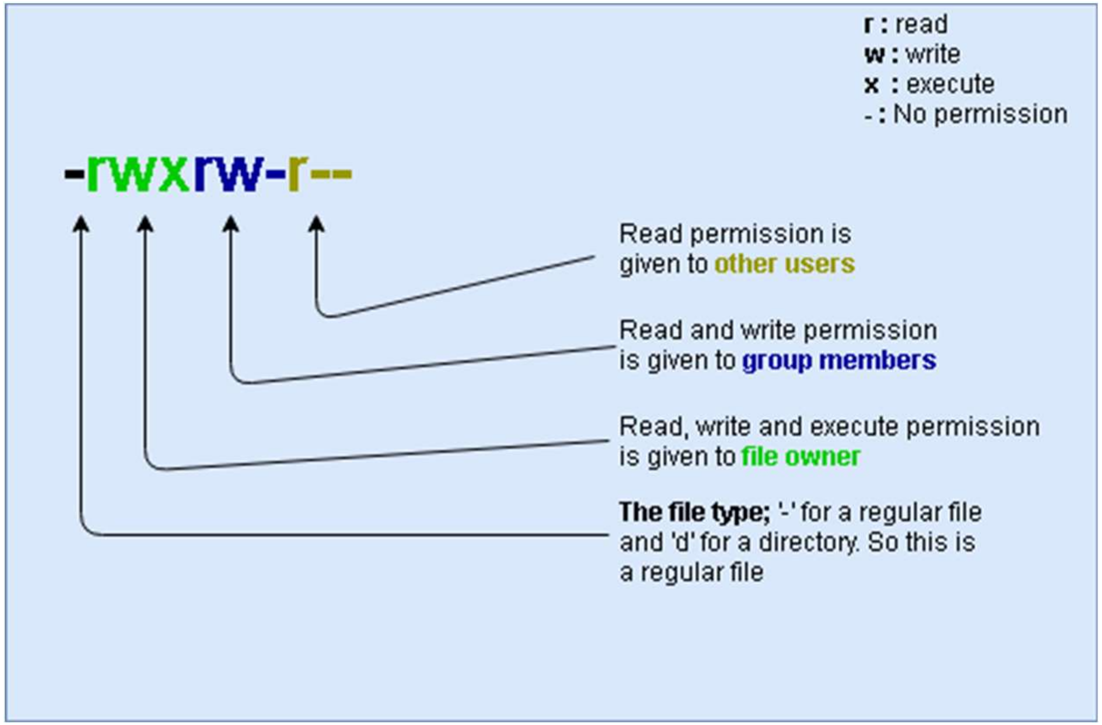
https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/FileSystemShell.html

Hadoop with S3


ViewFileSystem
ViewFileSystemOverloadScheme
viewfs://cluster/<mountPath>/
fs.defaultFS=viewfs://cluster
ViewFileSystem
/data
/user
/backup
Ozone
o3fs://bucket.vol.host/
HDFS
hdfs://cluster
S3
s3a://bucketA/
Mount Table name should be cluster
hdfs://cluster/<mountPath>/
fs.defaultFS=hdfs://cluster
ViewFileSystemOverloadScheme
/data
/user
/backup
Ozone
o3fs://bucket.vol.host/
HDFS
hdfs://cluster/
S3
s3a://bucketA/
Mount Table name should be cluster
(or)
s3a://bucketA/<mountPath>/
fs.defaultFS=s3a://bucketA
ViewFileSystemOverloadScheme
/data
/user
/backup
Ozone
o3fs://bucket.vol.host/
HDFS
hdfs://cluster/
S3
s3a://bucketA/
Mount Table name should be bucketA
* mountPath = /data (or) /user (or) /backup

Note: Above pictures showed examples with schemes hdfs and s3a only for understanding purpose. It's not limited to that schemes only. We can configure any scheme with ViewFileSystemOverloadScheme.

Linux: Commands



r : read
w : write
x : execute
- : No permission

-rwxrw-r--

Read permission is given to other users

Read and write permission is given to group members

Read, write and execute permission is given to file owner

The file type; '-' for a regular file and 'd' for a directory. So this is a regular file

- ls (list directory)
- mkdir (make a directory)
- touch (touch file)
- mv (move file)
- cp (copy files)
- rm (remove file)

Hadoop Distributed File System (HDFS)

- hadoop fs <args>
  - fs is used for generic file system, and it can point to any file system such as local file system, HDFS, WebHDFS, S3 FS, etc.
- hdfs dfs <args>
  - dfs points to the Distributed File System and it is specific to HDFS

Hadoop HDFS: Commands

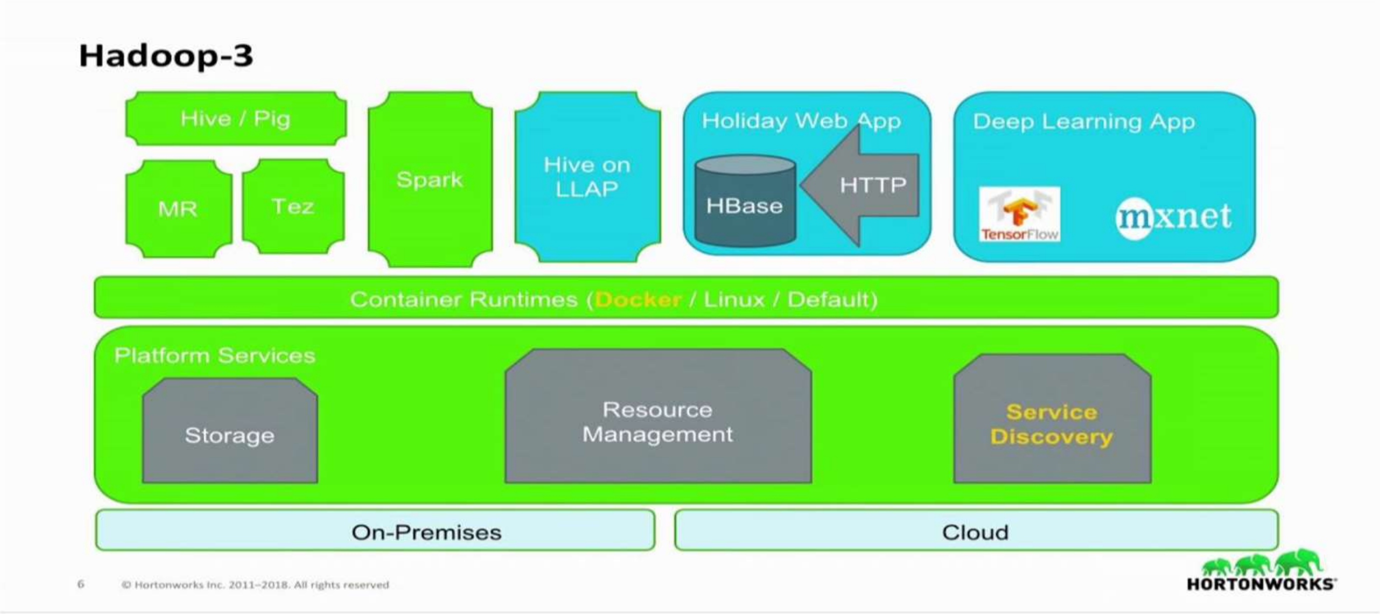
Command	Advantages
hdfs dfs –put localFile hdfsFile	Put a local file in hdfs
hdfs dfs –mkdir hdfsDir	Create a new directory in hdfs
hdfs dfs –cat hdfsFile	Display the content of a file
hdfs dfs –rm hdfsFile	Delete a file
hdfs dfs	Help

https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HDFSCommands.html

* In old versions of Hadoop : hadoop dfs cat ...

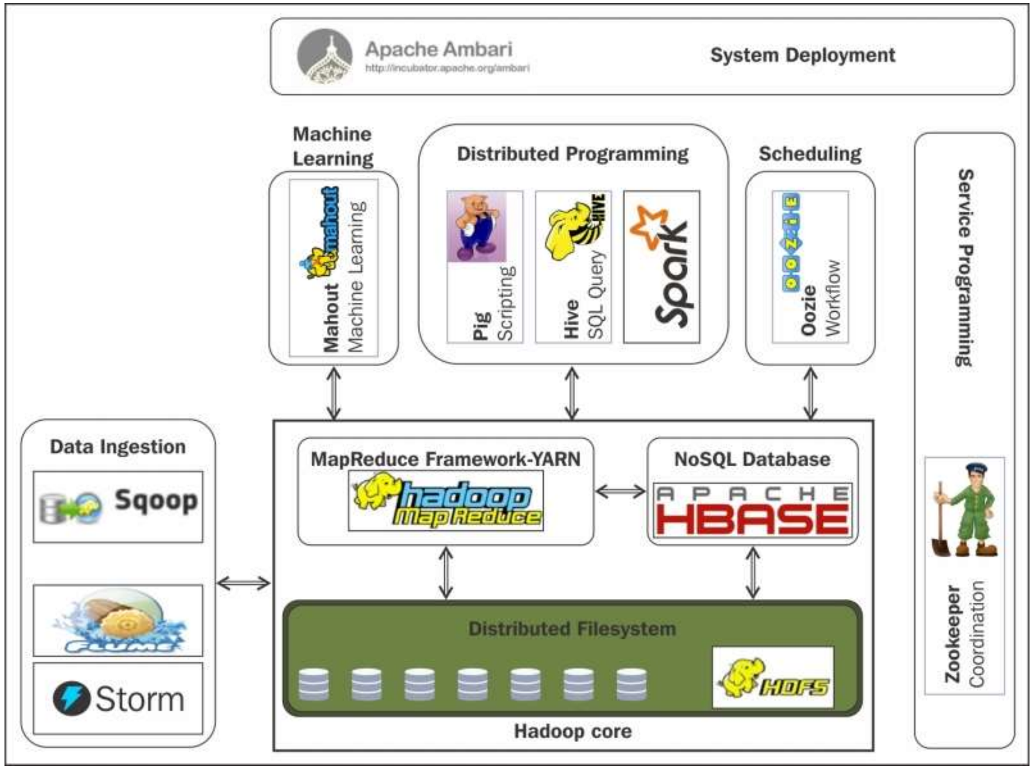
YARN: Yet another resource negotiator

- YARN — Yet Another Resource Negotiator, is a part of Hadoop, is one of the two major components of Apache Hadoop (with HDFS).
- YARN also allows different data processing engines like graph processing, interactive processing, stream processing as well as batch processing to run and process data stored in HDFS.



Apache Hadoop Ecosystem

- Hbase: A scalable, distributed database that supports structured data storage for large tables.
- Hive: A data warehouse infrastructure that provides data summarization and ad hoc querying.
- Pig: A high-level data flow language and execution framework for parallel computation.
- Zookeeper: A high-performance coordination service for distributed applications.
- Mahout: A Scalable machine learning and data mining library.
- …



Big Data processing

- Two main paradigms:
  - Batch.
  - In real time (Streaming).

- Batch processing
  - Large volumes of historical data are analyzed.
  - Processing can take minutes, hours, days...
- Real-time processing (Streaming)
  - Large volumes of data are analyzed that reach high speed.
  - Processing must be immediated.

Google's Solution: MapReduce

- In 2004, Google announces MapReduce.
- It is a framework for data processing:
  - Process data stored over GFS.
  - Do it in a distributed and parallel way.
  - Avoid data loss if a computer crashes.

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

[illegible]

Google, Inc.

Abstract

[illegible]

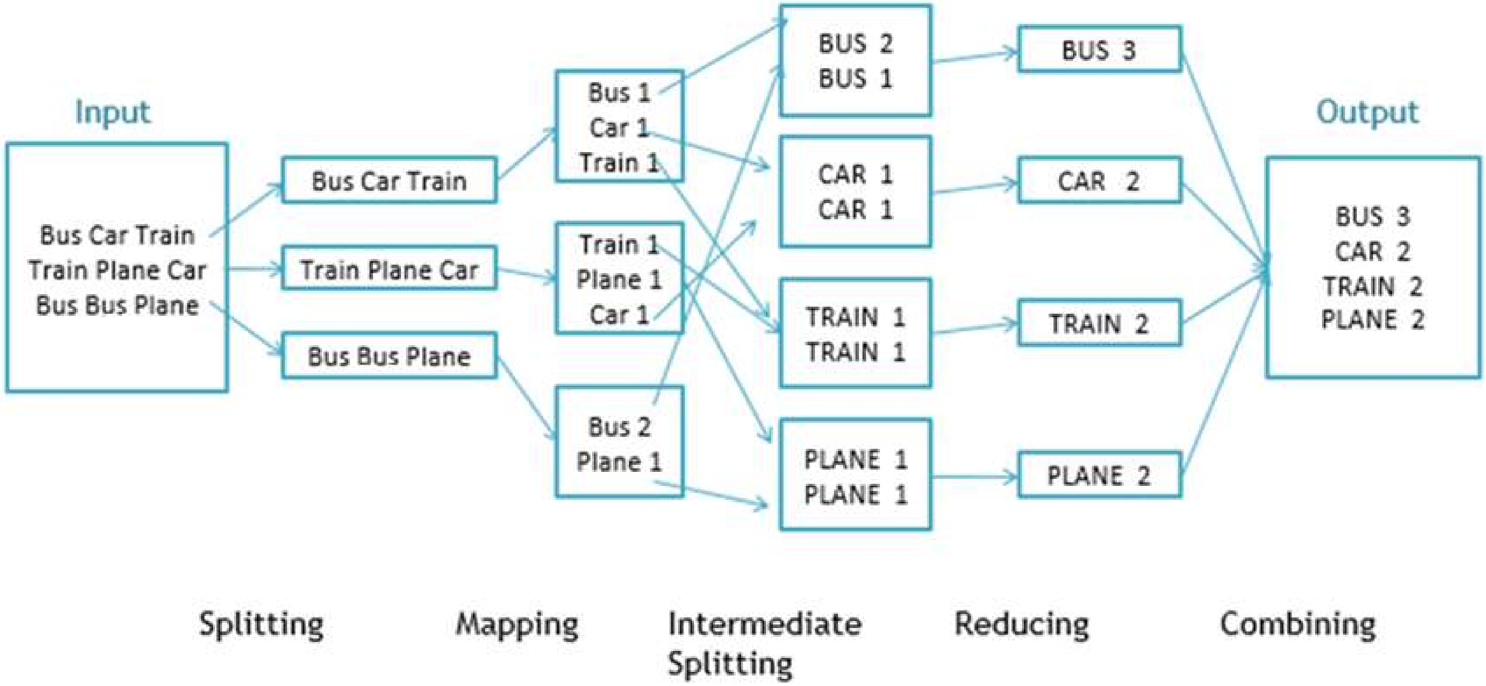
Introduction

[illegible]

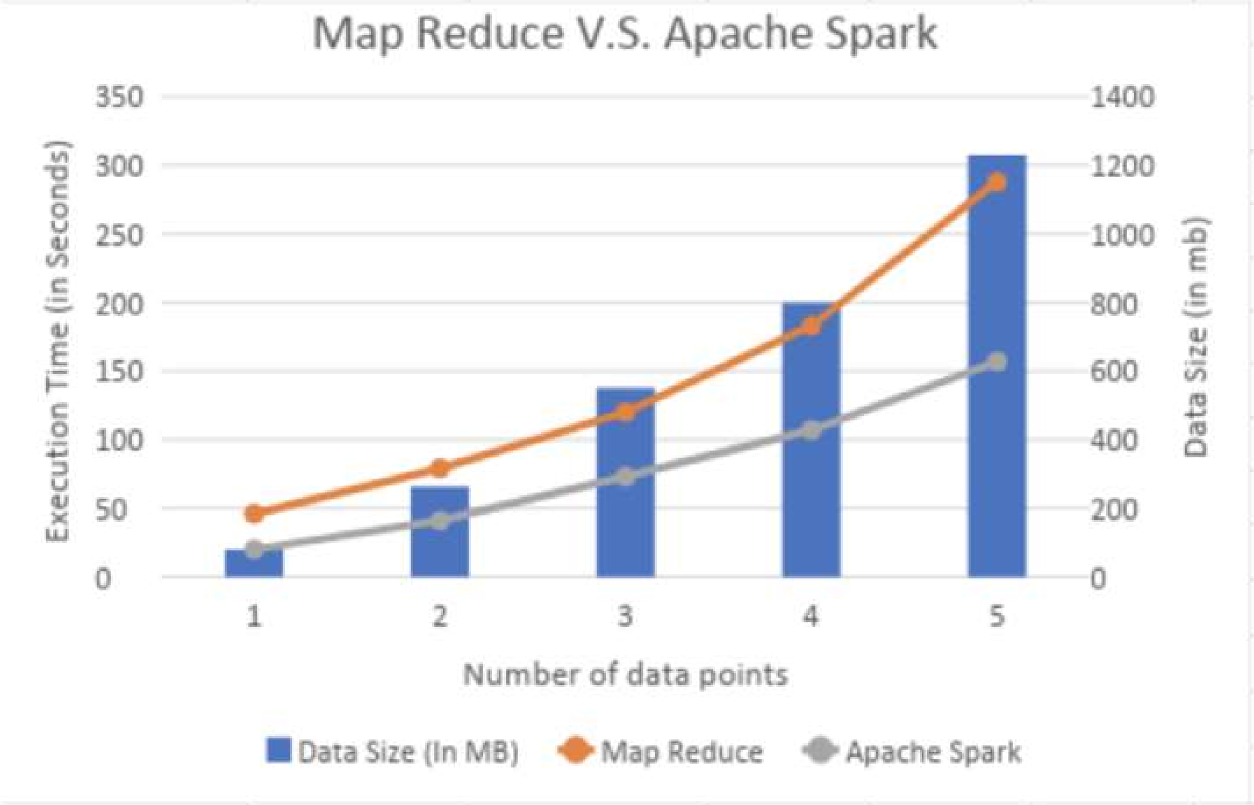
Hadoop MapReduce

- In 2005, an open-source version of MapReduce is included in HADOOP.
- It consists of two main functions.
- The **map** function performs a transformation of the data.
  - INPUT: a key value tuple.
  - OUTPUT: one or more key-value tuples.
- An intermediate phase called **shuffle** is responsible for grouping the output of the map by keys.
- The **reduce** function performs a grouping or aggregation of the data:
  - INPUT: A key, along with all its associated values.
  - OUTPUT: a value.

Hadoop MapReduce



Hadoop MapReduce



Hadoop distribution (Virtual Machines & Docker)

- Cloudera (https://www.cloudera.com/)

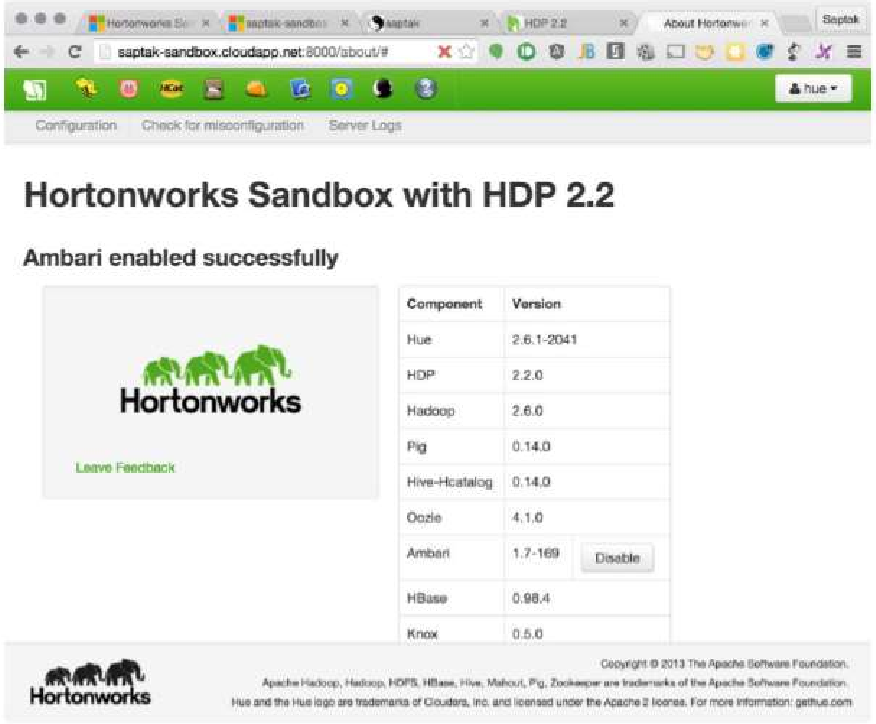
CLOUDERA

- Hortonworks (https://es.hortonworks.com/)

HORTONWORKS®
POWERING THE FUTURE OF DATA™

- MapR (https://mapr.com/)

Hadoop distribution (Virtual Machines & Docker)



Hortonworks Sandbox with HDP 2.2

Ambari enabled successfully

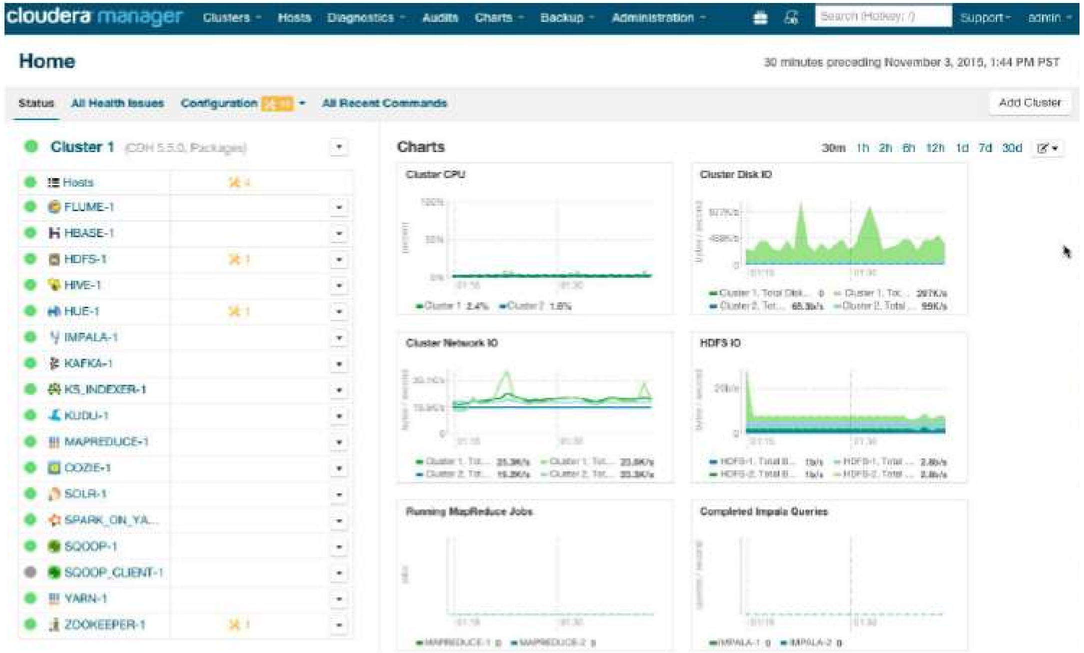
Hortonworks

Leave Feedback

Component	Version	
Hue	2.6.1-2041	
HDP	2.2.0	
Hadoop	2.6.0	
Pig	0.14.0	
Hive-Hcatalog	0.14.0	
Oozie	4.1.0	
Ambari	1.7-169	Disable
HBase	0.98.4	
Knox	0.5.0	

Hortonworks

Copyright © 2013 The Apache Software Foundation.
Apache Hadoop, Hadoop, HDFS, HBase, Hive, Mahout, Pig, Zookeeper are trademarks of the Apache Software Foundation.
Hue and the Hue logo are trademarks of Cloudera, Inc. and licensed under the Apache 2 license. For more information: gethue.com



Home

30 minutes preceding November 3, 2015, 1:44 PM PST

Cluster 1 (CDH 5.5.0, Packages)

- Hosts 4
- FLUME-1
- HBASE-1
- HDFS-1 1
- HIVE-1
- HUE-1 1
- IMPALA-1
- KAFKA-1
- KS_INDEXER-1
- KUDU-1
- MAPREDUCE-1
- OOZIE-1
- SOLR-1
- SPARK_ON_YA…
- SQOOP-1
- SQOOP_CLIENT-1
- YARN-1
- ZOOKEEPER-1 1

Charts

Hadoop distribution (Hortonworks)

DOCKER

Download HDP_3.0.1_docker-deploy-scripts.zip from
https://archive.cloudera.com/hwx-sandbox/hdp/hdp-3.0.1/HDP_3.0.1_docker-deploy-scripts_18120587fc7fb.zip
bash docker-deploy-hdp30.sh

http://127.0.0.1:8080
admin/admin

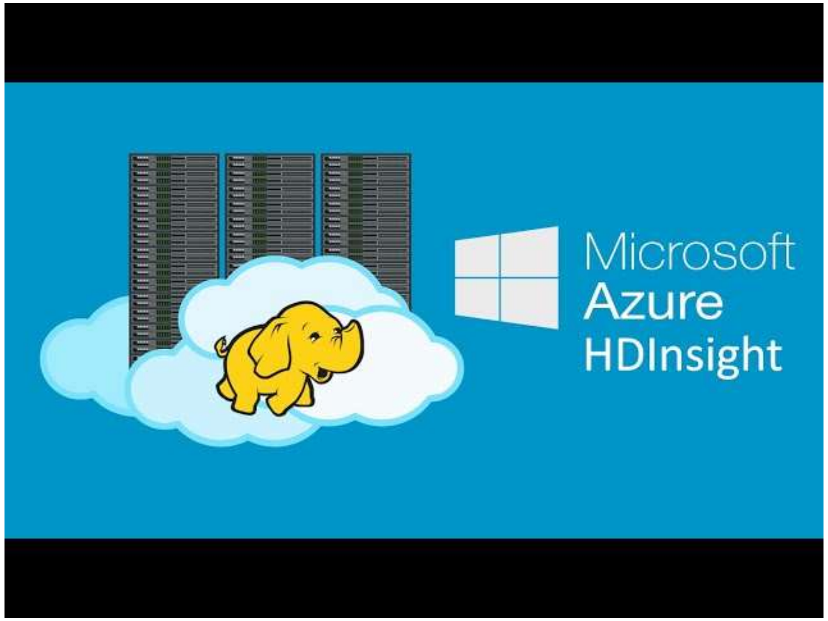
docker stop sandbox-hdp
docker stop sandbox-proxy

docker start sandbox-hdp
docker start sandbox-proxy

Máquina virtual VirtualBOX o VMWare

https://archive.cloudera.com/hwx-sandbox/hdp/hdp-3.0.1/HDP_3.0.1_virtualbox_181205.ova

Hadoop cloud



https://azure.microsoft.com/es-es/products/hdinsight

https://aws.amazon.com/es/emr/features/hadoop/

https://cloud.google.com/architecture/hadoop?hl=es-419