Hadoop

HBD - 01/2019

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.

Hadoop project: components

Hadoop Distributed File System (HDFS™):
A distributed file system that provides high-throughput access to application data

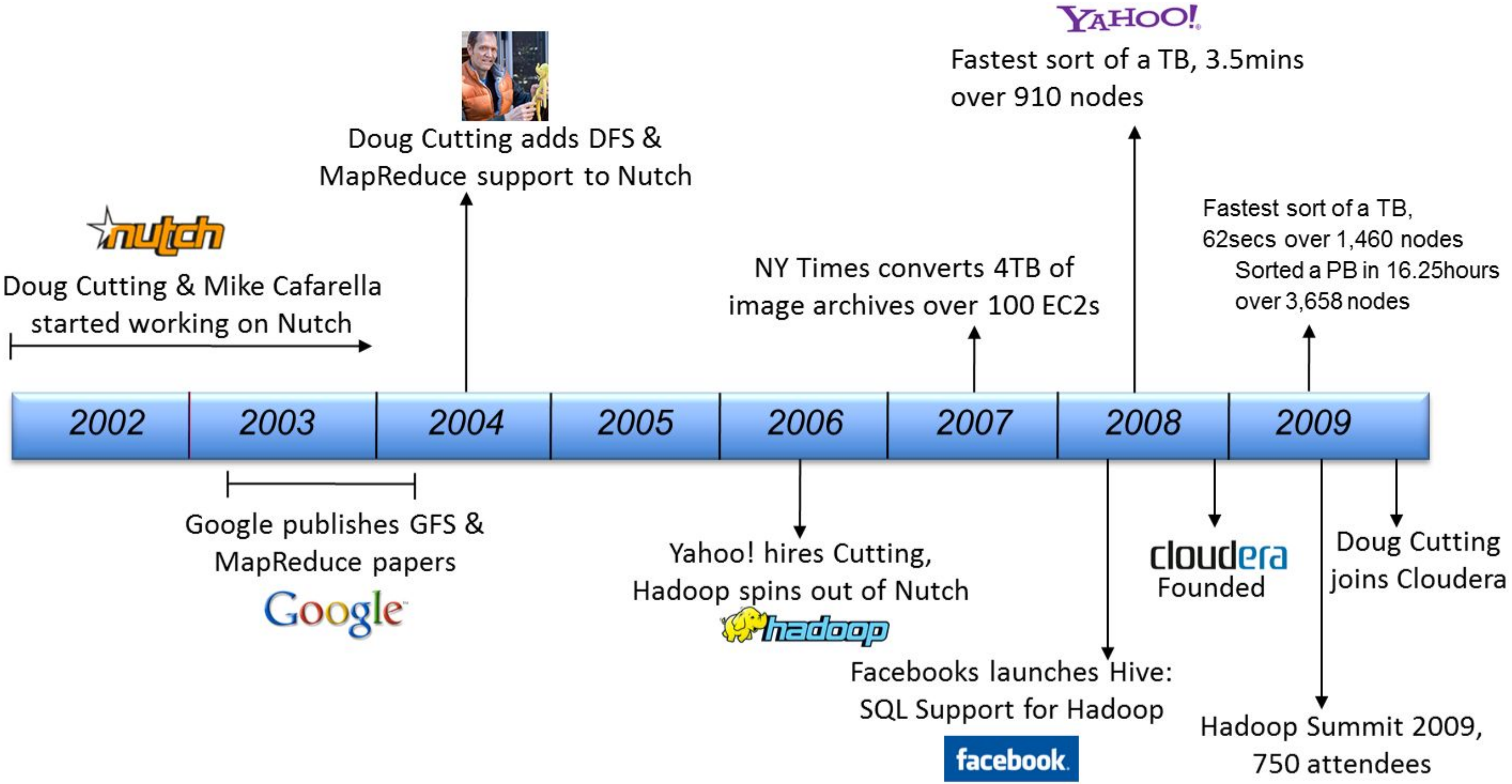
Hadoop MapReduce

A YARN-based system for parallel processing of large data sets.

Hadoop YARN

A framework for job scheduling and cluster resource management.

Hadoop project: History



Hadoop project: History

- 2000: **Cutting open sourced Lucene** with great feedback from the community.
- 2001: **Cutting started working with Mike Cafarella** in an open source project to index the entire web. **Apache Nutch** was born as a sub-project of Lucene. Nutch was designed for a single machine and as they started to scale huge complexities arose in storage first and processing later.
- 2003: **Google released the GFS paper**. They implemented Nutch Distributed File System (NDFS) using GFS paper as a guide and released in 2004.
- 2004: in December **Google released the Map Reduce paper**.
- 2005: Nutch got a Map Reduce implementation as a processing layer.
- 2006: **Cutting** separated Map Reduce and NDFS and **created the Hadoop project** under Lucene.

Hadoop project: History

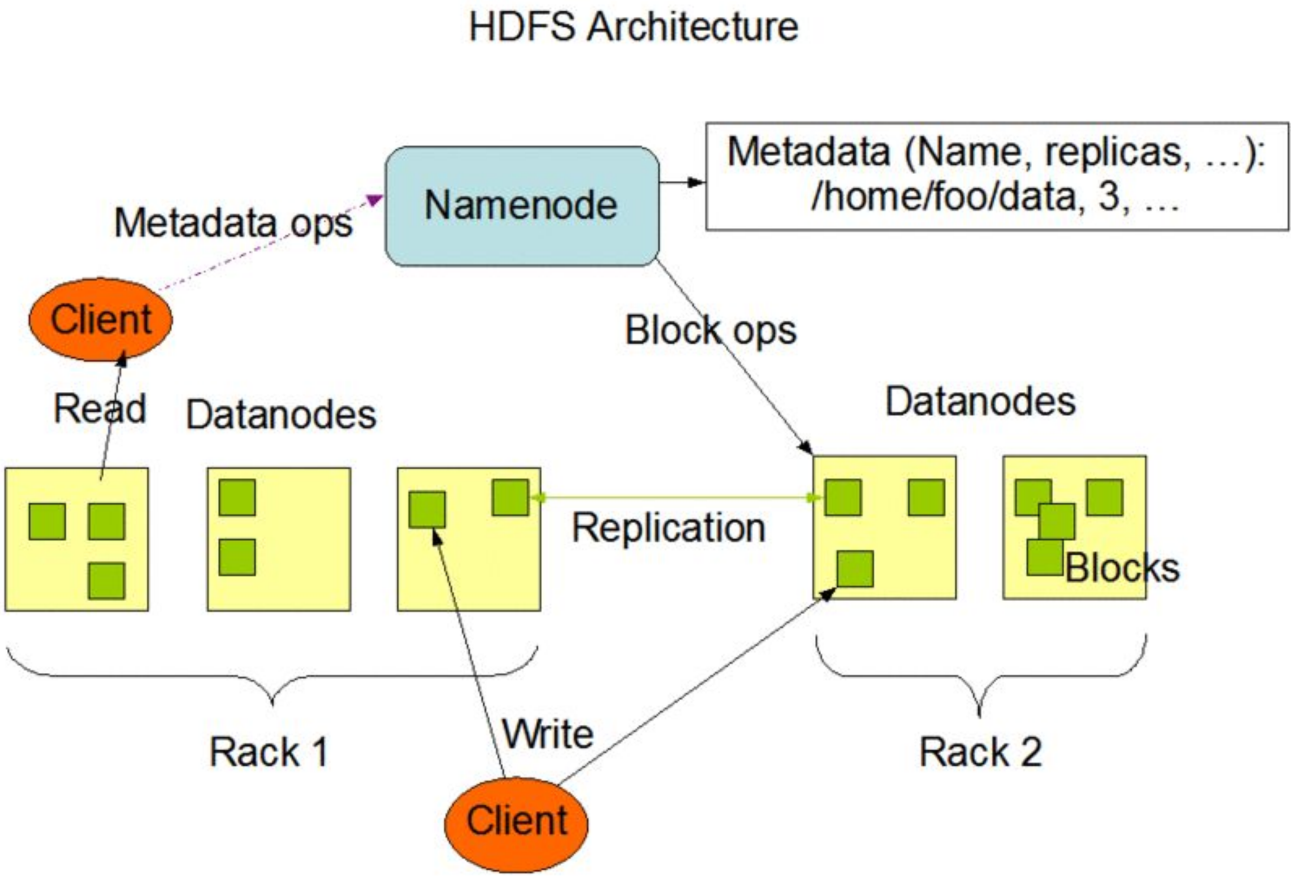
- 2006: **Yahoo**, that was going through great troubles to scale and compete with Google, **hires Cutting and adopts Hadoop to re-implement Yahoo's search backend**.
- 2007: the first web scale companies like **Twitter, Facebook and LinkedIn started adopting and contributing to Hadoop**. Yahoo already had a 1000 node cluster.
- 2008: **Hadoop graduated as Apache top level project**. Auxiliary sub projects like **HBase, Zookeeper and Hive started appearing. Cloudera was founded**.
- 2010-2012: Hadoop adoption exploded. **Hadoop team at Yahoo formed an independent company name Hortonworks**. Yahoo reports 42000 nodes cluster.
- 2013: Hadoop releases 2.2 with YARN and HDFS Federation
- 2017: Hadoop 3.0 release planned

HDFS

Hadoop distributed file system

HDFS: Characteristics.

- **Distributed file system** modeled after [Google File System paper]().
- Optimized for **high throughput** and works best when **reading and writing large files** (gigabytes plus)
- Unusually **large block size**.
- Manages file replication to achieve **fault tolerance**.
- Designed to run on **commodity hardware** (no RAID, NFS, etc.)


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

HDFS: File Blocks.

- Files in HDFS are s**tored in block-sized chunks**. (default size 128MB)
- Each block is **replicated** in different nodes according to replication factor.
- If a file is **smaller** than the configured block size it **occupies its actual size** in disk.
- **Blocks simplify file handling** because they have a maximum size and permit storing a file bigger than a single machine disk size.

HDFS: Name Node

- **Manages the filesystem namespace** and regulates access to files by clients.
- **Maintains the filesystem tree and the metadata** for all the files and directories (location, size, access rights, etc.).
- **Persists this information** in local disk as an image (FsImage) and a transaction log.
- **Knows the DataNodes** and where all the file's actual data is stored.
- Answers requests for **block locations**.

HDFS: Data Nodes.

- **Store blocks** that compose the files stored in HDFS.
- **Reports** every 3 seconds back to NameNode with a **list of blocks** that they are storing.
- DataNodes **forward block data** between in each other **when a file is being replicated**.

HDFS: Limitations

1. **The NameNode is a single point of failure**.
   a. Solved in HDFS 2.X with federations and in 3.X by HA.
2. **Bad random data-access**
   a. Needs a database like HBase to complement this use cases.
3. **Many small files degrade performance**
   a. Small files should be avoided by careful partitioning or merging.

How to use it?

```
$> hadoop fs -command <args>
```

```
$> hadoop fs -get hdfs://nn.example.com/user/hadoop/file localfile #Copy files to the local system.
$> hadoop fs -put -f localfile1 localfile2 /user/hadoop/hadoopdir #Copy from local to destination system
$> hadoop fs -cp /user/hadoop/file1 /user/hadoop/file2 #copy from source to destination.
$> hadoop fs -mv hdfs://nn.example.com/file1 hdfs://nn.example.com/file2 #Moves from source to destination.
$> hadoop fs -mkdir /user/hadoop/dir1 #creates directories.
$> hadoop fs -ls /user/hadoop/file1 #returns list of its direct children
$> hadoop fs -rm hdfs://nn.example.com/file /user/hadoop/emptydir #Delete files specified.
$> hadoop fs -rmdir /user/hadoop/emptydir #Delete a directory.
$> hadoop fs -tail pathname #Displays last kilobyte of the file to stdout.
$> hadoop fs -getmerge -nl /src /opt/output.txt #Takes a source directory and a destination file as input and concatenates files in src into the destination local file.
```

more commands

Parallel Copying with distcp

You can use the **distcp** tool to copy files or directories **between HDFS cluster using a MapReduce job** where the work of copying is done by the maps that run in parallel across the cluster.

Options:

- **overwrite**: when used between directories, keeps the same directory structure and force files to be overwritten.
- **update:** updates only the files that have changed between two directories.
- **delete:** deletes any files or directories from the destination that are not present in the source.
- **p**: preserves file status attributes like permissions, block size, and replication.
- **log**: specifies where to place the log of the executed distcp operation.

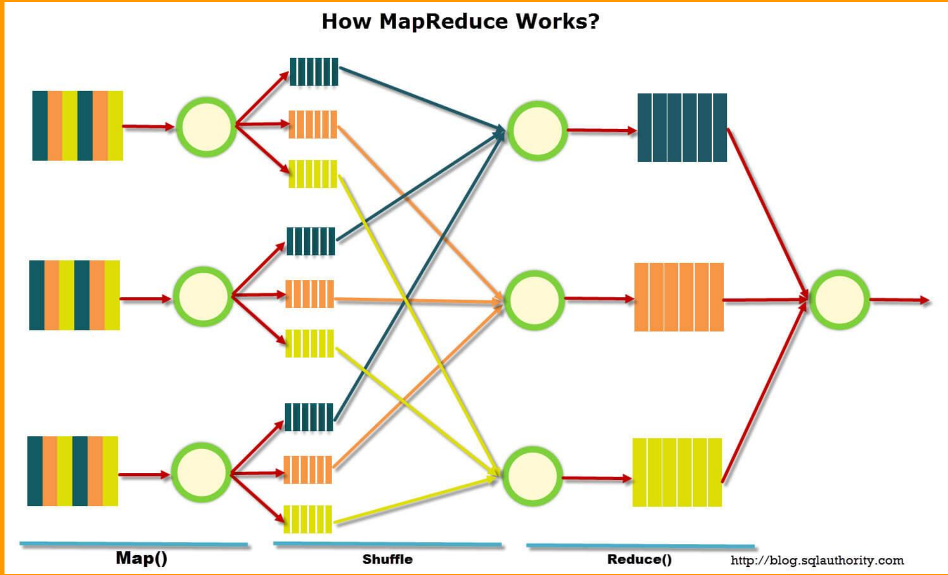
Parallel Copying with distcp

If the two clusters are running incompatible versions of HDFS, you can use the webhdfs or HttpFs protocol to distcp between them:

```
$> hadoop distcp webhdfs://namenode1:50070/foo \ webhdfs://namenode2:50070/foo
```

Real-world example of a copy between Hadoop 1.0.3 cluster and Hadoop 2.x cluster:

```
$> hadoop distcp -log /tmp/distcp_transactions -update
hftp://old-cluster-master:50070/user/hive/warehouse/transactions
hdfs://new-cluster-master:8020/data/import/transactions
```


How MapReduce Works?
Map()
Shuffle
Reduce()
http://blog.sqlauthority.com

Map Reduce

Map Reduce

Framework for **processing parallelizable problems** across huge datasets using a large number of computers.

Introduced by Google in a paper in 2004.

The model is inspired by the **map** and **reduce** functions commonly used in functional programming.

Also uses the hypothesis that most programs receive and output the data as a **pair of a key and a value**.

Map Reduce: Computation Steps

Map

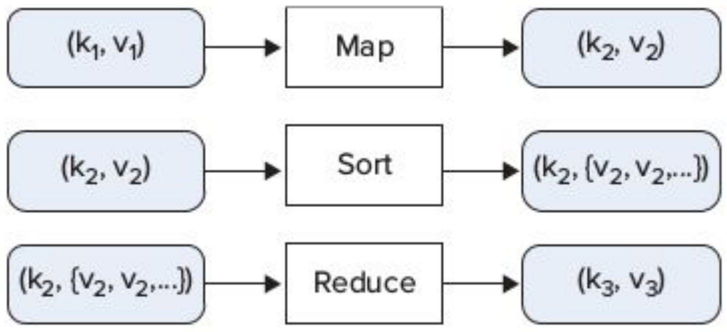
Written by the user, takes an input pair and **transforms** it into a set of **intermediate** key/value pairs

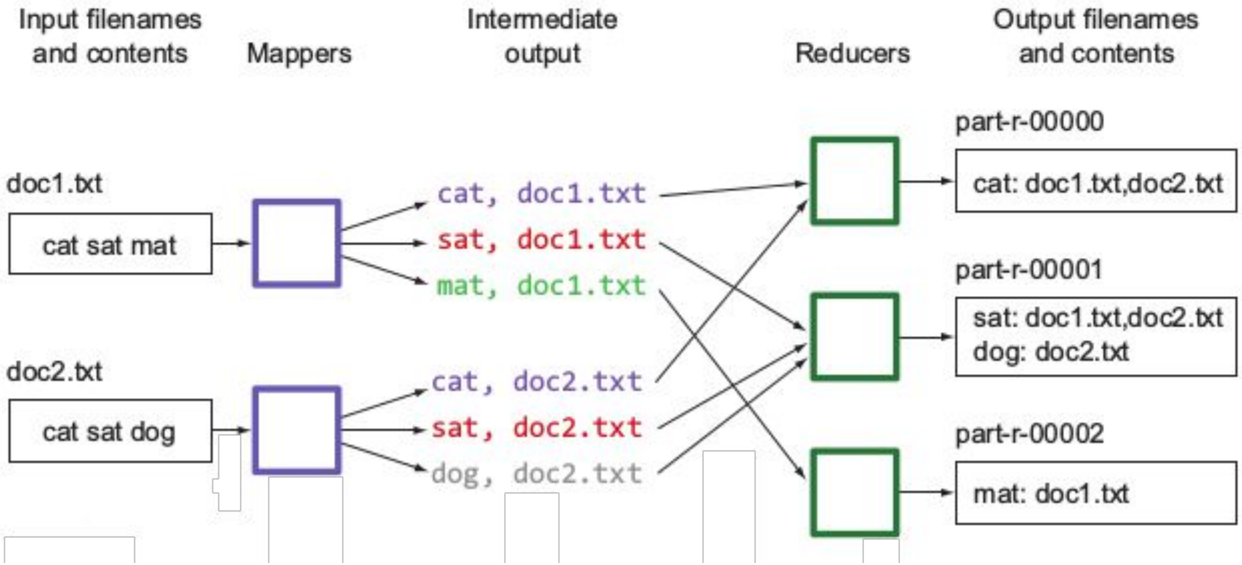
Sort

Provided by the framework, **groups** together all values associated with the same key

Reduce

Written by the user, **merges** all the values for the same key into a **final value.**

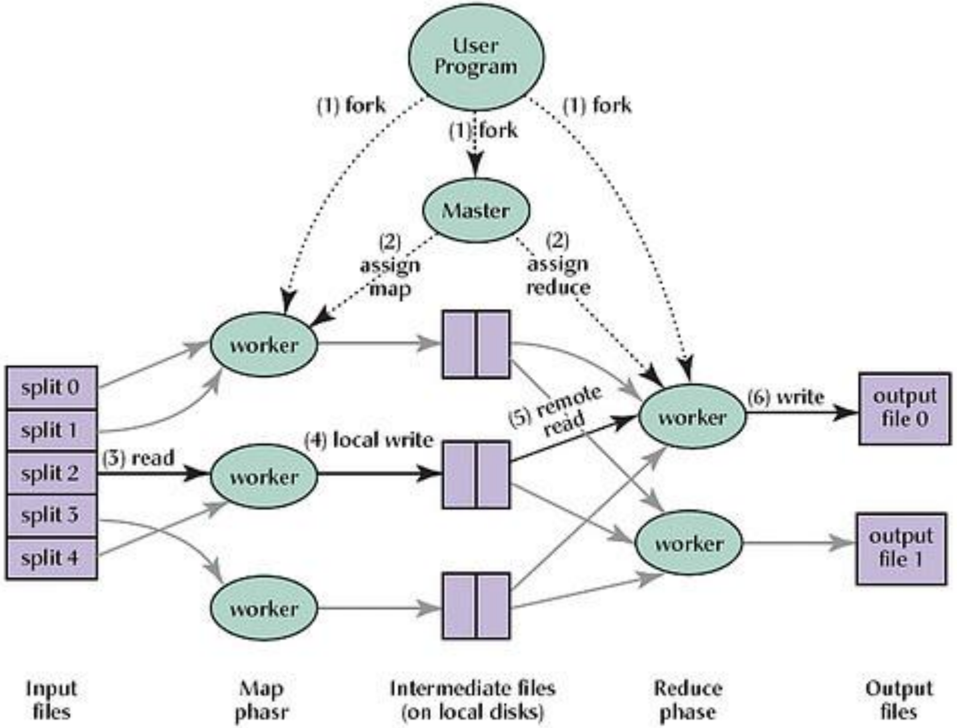




Example: inverted index

Performance Considerations

- **Optimized for high throughput** processing large volumes of data.
- **Data locality is important** though sometimes the benefits of things like virtualization outweigh the performance gains.
- Between every step data is **persisted to disk**.
- If a lot of data that needs to be **shuffled** through the different nodes it **can degrade performance**.

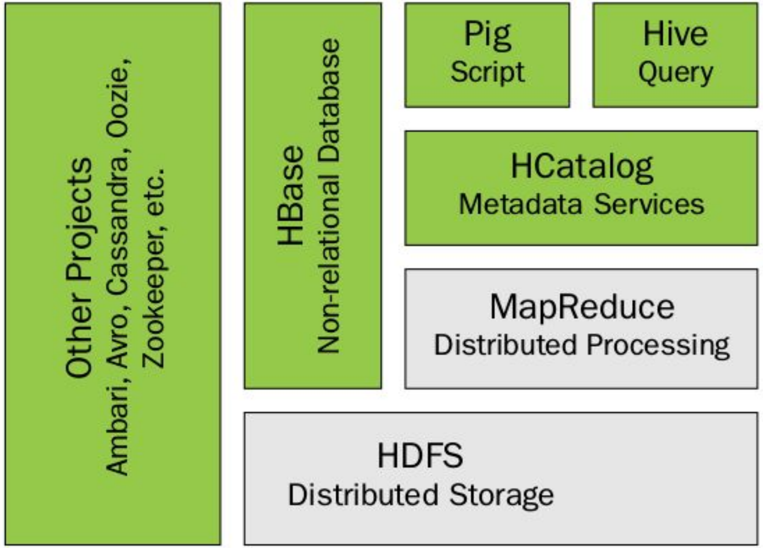


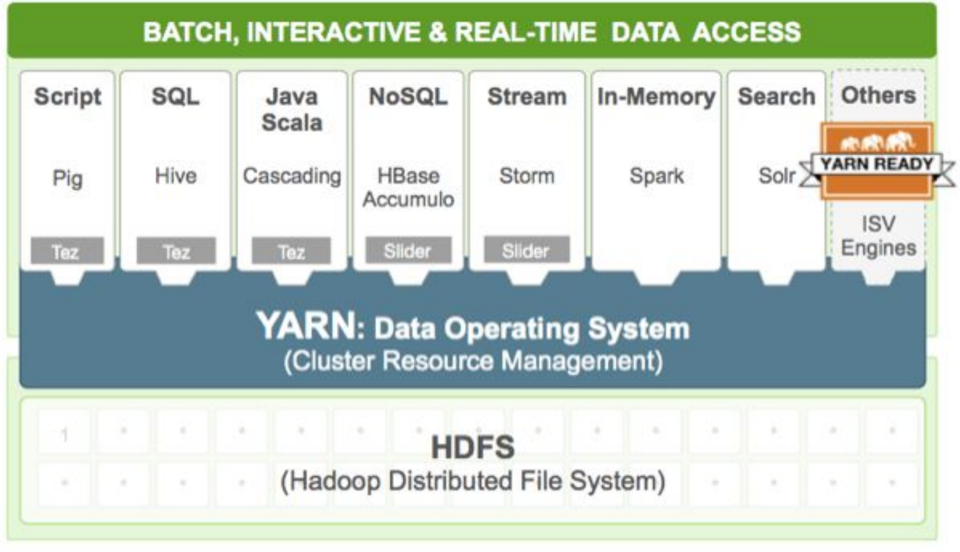
Map reduce Architechture

Yarn

YARN: History

- In Hadoop (v1) Map Reduce **was handling too many functionalities**: assigning cluster resources and managing job execution (system), doing data processing (engine) and interfacing towards clients (API).
- For Hadoop v2 a separated project was born codename **YARN** (Yet Another Resource Negotiator). it **would handle the resource management, workflow management and fault-tolerance components** <u>to enable different types of processing workflows other than Map Reduce.</u>

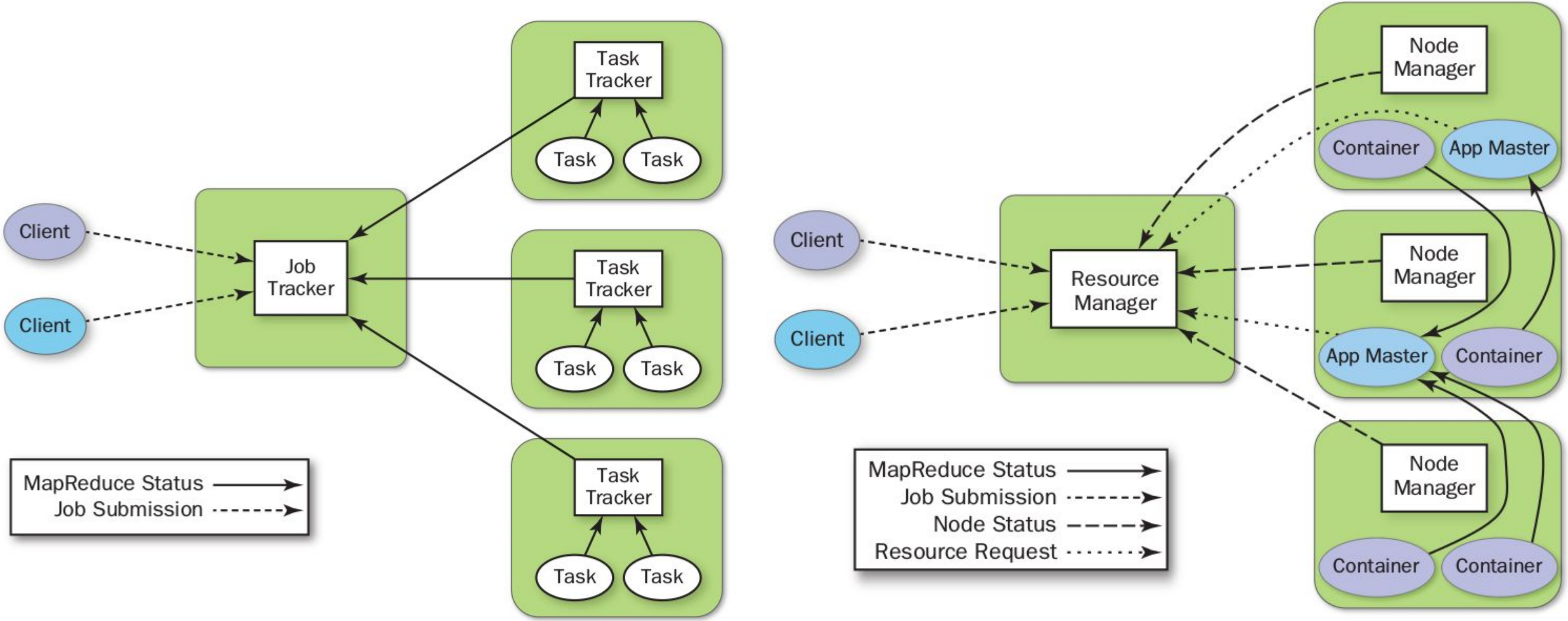




Pre-Yarn every component of a Hadoop Stack would be mostly independent in resource management

With Yarn all applications can run on the container **sharing resources**. Even **non Hadoop** applications

Hadoop 1 vs Hadoop 2



Hadoop V1

- **Jobtracker**: controls the execution of MapReduce jobs by pushing work to the TaskTrackers and trying to keep the work as close to the data as possible.
- **TaskTracker**: resides in each worker node together with the DataNode. Executes the actual computation work, monitors it’s execution and reports back to the Jobtracker.

YARN: processes

- **Resource Manager**: Handles an overall resource profile for each application, ignoring local optimizations and internal application flow.
- **Application Master**: Works with the NodeManager(s) to execute and monitor the containers of the specific task and their resource consumption during task execution.
- **Node Manager**: “worker” agent. Keeps the RM up-to-date, oversees an application containers’ life-cycle management, monitors resource usage (memory, CPU) of individual containers, tracks node health and manages logs.

Hadoop Related Projects

Hadoop ecosystem

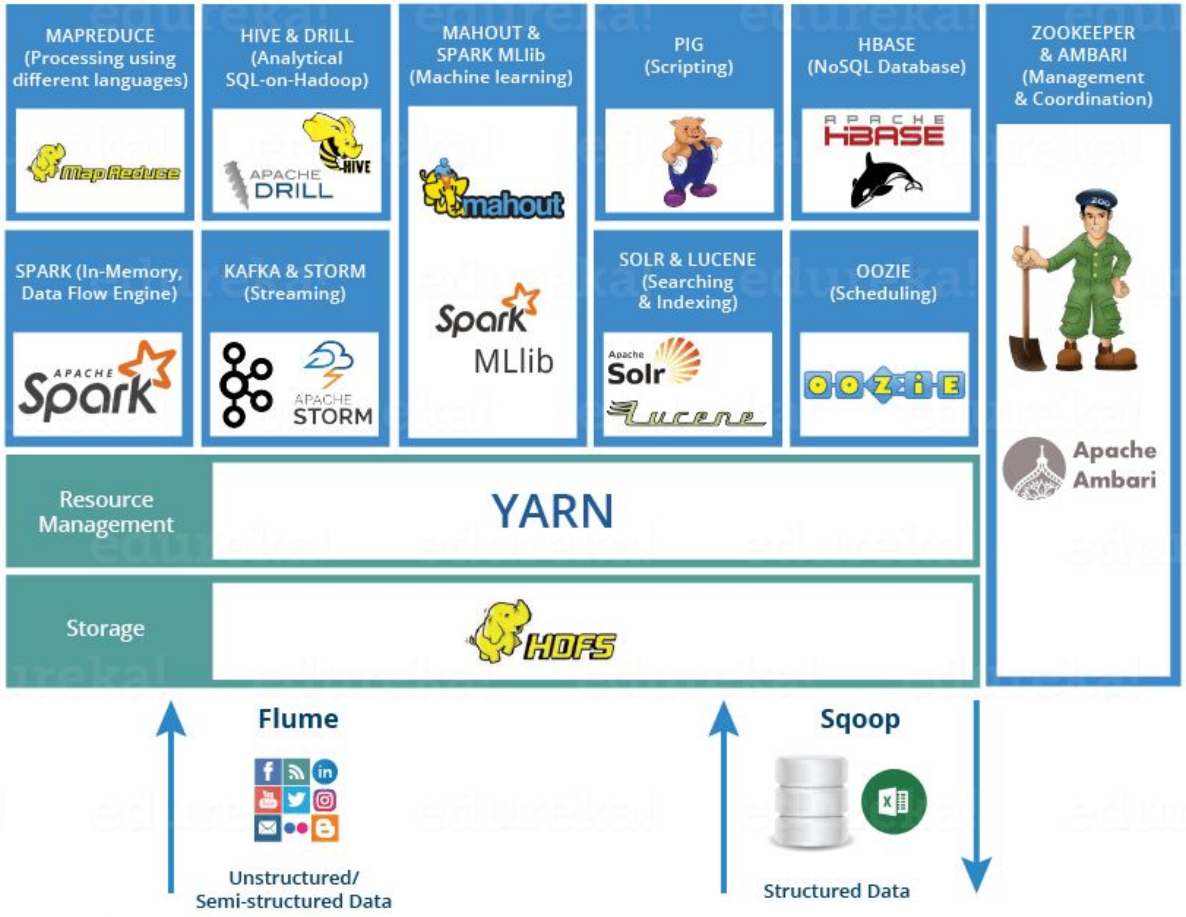
Hadoop sparked a set of **related projects** that would built-on, **improve** and provide support to the execution of a map reduce process.

Many of this projects **evolved out** of the hadoop sphere and have become part of stacks of other big data tools like spark, kafka, etc.

Hadoop ecosystem

- Ambari™: A web-based tool for provisioning, managing, and monitoring Apache Hadoop clusters .
- Avro™: A data serialization system.
- Cassandra™: A scalable multi-master database with no single points of failure.
- Chukwa™: A data collection system for managing large distributed systems.
- 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.
- Mahout™: A Scalable machine learning and data mining library.
- Pig™: A high-level data-flow language and execution framework for parallel computation.
- Spark™: A fast and general compute engine for Hadoop data.
- Tez™: A generalized data-flow programming framework, built on Hadoop YARN.
- ZooKeeper™: A high-performance coordination service for distributed applications.

Hadoop ecosystem



Final Thoughts

Map Reduce / Hadoop Critics

- **Mapreduce**: the main critic of the paper’s process is that is not generic enough. Many problems are hard to design in a mapreduce way,
- **Hadoop**: hadoop implementation (as suggested by the paper), **writes the intermediate result to disk**. Making the the process slow compared to newer frameworks.
- Early implementation of hadoop had also problems with scaling and managing resources.

Key Takeaways

- **Hadoop** with its horizontally scalable and fault tolerant capabilities **democratized Big Data processing and storage**.
- As needs evolved a new generalized Hadoop platform appeared together with new tools.
- Though the mapreduce model is loosing track against other more general algorithms and Hadoop's implementation is criticized due to being slow. Many of the stack application are being used as complement of the newer frameworks.
- Hadoop is still being used in many organizations and there is 3.X version to be released in the near future

CREDITS

Content of the slides:

- Big Data Tools - ITBA - 2017

Images:

- Big Data Tools - ITBA - 2017
- obtained from: commons.wikimedia.org

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Unsplash