UE Large Scale Processing @ ESIGELEC
2019/2020

05 – Apache Hadoop

Abdel Dadouche
DJZ Consulting

adadouche@hotmail.com
@adadouche

Hadoop historical facts

Some historical facts

1997	First appearance of the term "**Big Data**" by NASA to designate the challenges working on large volumes of unstructured data
2001	The 3V's (Volume / Variety / Velocity)
2003 & 2004	Google releases the **GFS** (Google File System) & **MapReduce** whitepapers
2004	Doug Cutting & Mike Cafarella release Nutch, a highly extensible and scalable open source web crawler project that implemented the first MapReduce facility on a Distributed File System
2006	Based on Nutch, Doug Cutting (now at Yahoo!) is enhancing the search engine indexation. This was then moved under the new Hadoop subproject.

Where does the Elephant and Name Hadoop come from?

APACHE
hadoop
TM

His son made up the name after his toy elephant!

Source: https://en.wikipedia.org/wiki/Big_data
https://twitter.com/theregister/status/463858134683877378

Some historical facts

April 2006	Apache Hadoop 0.1.0 was released
2008	Cloudera was founded by three engineers from Google, Yahoo! and Facebook (Christophe Bisciglia, Amr Awadallah and Jeff Hammerbacher)
2008	A Yahoo! Hadoop cluster beat the Terabyte Sort Benchmark (1 TB of data in 209 seconds)
2009	MapR was founded
2010	Dhruba Borthakur claimed that Facebook had the largest Hadoop cluster with 21 PB of storage
2011	HortonWorks was founded
December 2011	Apache Hadoop 1.0.0 was released
2014	Apache Spark 1.0.0 was released
2014	Apache Spark beat the Terabyte Sort Benchmark (100 TB in 1406 seconds)

The key principles behind Hadoop

The key principles behind the Hadoop plaform

- An open & extendable platform
- Distributed architecture
- Provide API and tools

- Easy to install, deploy & maintain on a multi-platform environment
- Scalable & Reliable

- A more agile & flexible approach / philosophy to store and process data !

The key principles behind Hadoop

An open & extendable platform

- Apache Foundation Open Source project since 2009:

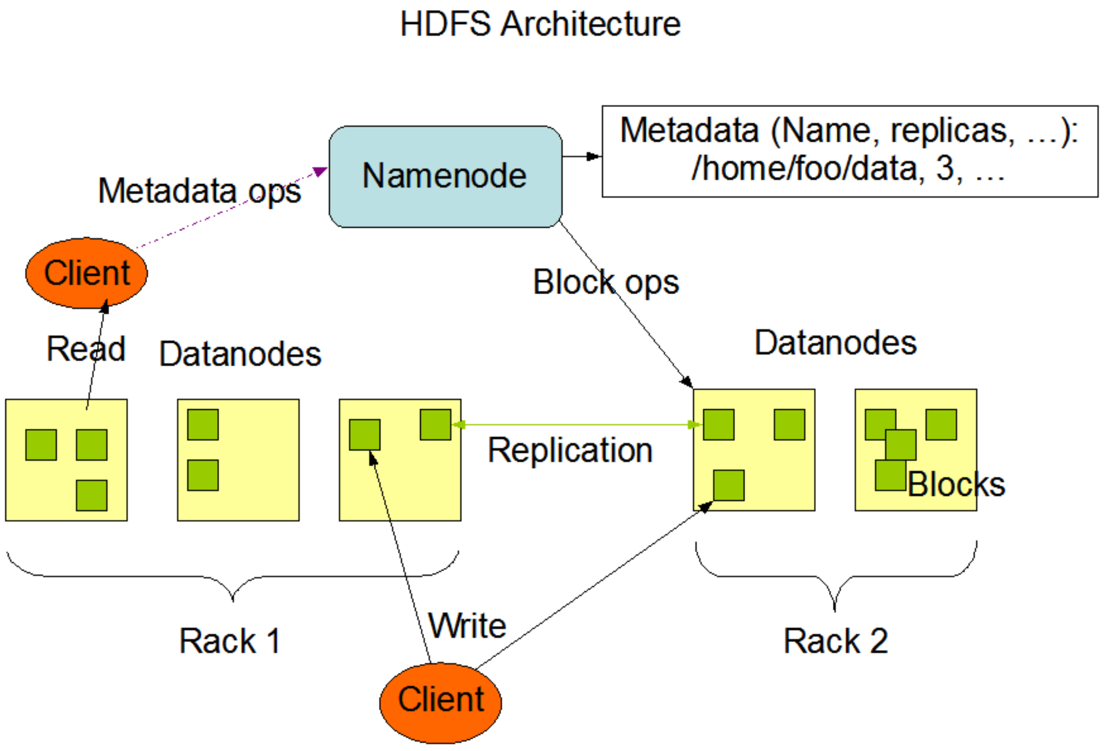
  https://hadoop.apache.org

- Developed in Java (multiplatform by design) but can now also use native libraries to improve performances
- Anyone can contribute with code, reporting issues and even start a subproject

The key principles behind Hadoop

Distributed Data Storage

- HDFS use a master/slave architecture
- The NameNode (master) manages the file system namespace and regulates access to files by clients.
- The DataNodes (slave), one per node in the cluster, manage storage attached to the nodes that they run on.

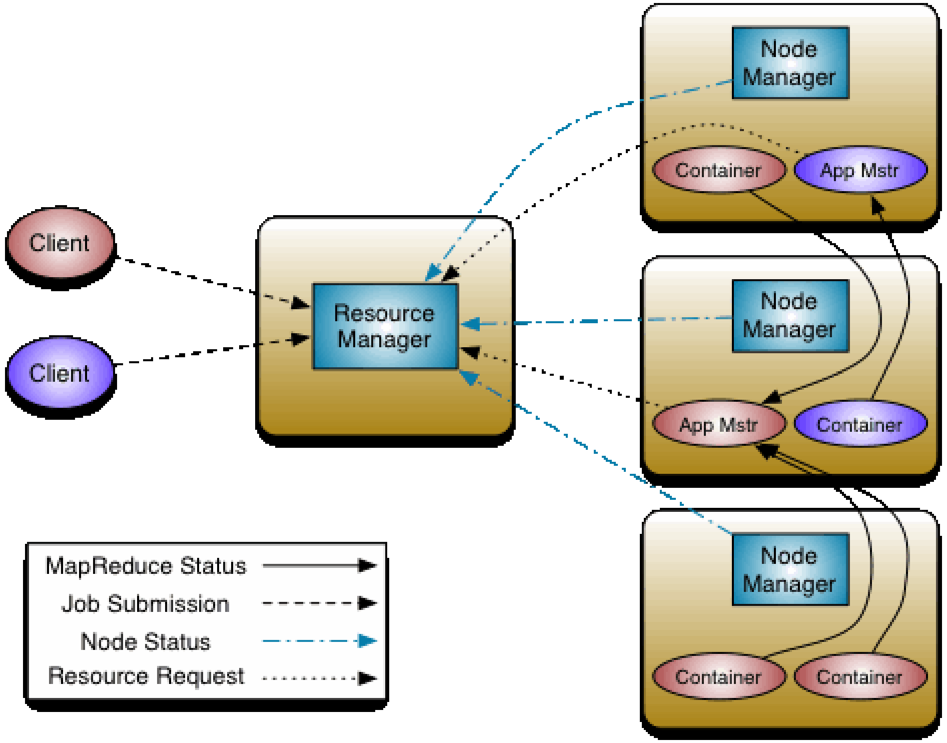


Source: https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

The key principles behind Hadoop

Distributed Computing, YARN & MapReduce

- The fundamental of YARN is to split the resource management and job scheduling/monitoring into separate daemons
- The ResourceManager arbitrates resources among all the applications
- The NodeManager is the per-machine agent who is responsible for containers, monitoring their resource usage (cpu, memory, disk, network) and reporting the same to the Resource Manager.



Source: https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

The key principles behind Hadoop

API and Tools for Developers

- Multiple API available for HDFS & MapReduce:

  Java, C++, Python, NodeJS…

- Multiple tools available to administer and monitor your Hadoop clusters:

  Command line tools (fs, job, dfsadmin, jobtracker…)

  Ambari, ZooKeeper and many vendor specific tools…

Warning:

There are plenty of duplicate APIs.

Some are not maintained or simply outdated

The key principles behind Hadoop

Easy to install, deploy & maintain on a multi-platform environment

- It's a written in Java!
- Can be downloaded and installed from :

  https://hadoop.apache.org/releases.html

- Also available for most Linux distribution as packages (via apt-get, yum, rpm etc.)
- Can be installed on Windows as well! (this will be your first assignment)

The key principles behind Hadoop

Reliable, Scalable & Highly Available

- Data is chunked then distributed across the cluster
  - Multiple copies of the same chuck exists on different nodes
- New nodes can be added without downtime
  - Data chunks will start being added
- If a node crashes, it won’t affect the availability of the all data

- Job logic (code) is distributed across the cluster
- Each node will process the data chunks locally stored
- Each data chunks will be processed by only one node (shared-nothing architecture)
- If a node crashes, the job can be executed on a different node where data chunk is available

Source: https://en.wikipedia.org/wiki/Shared-nothing_architecture

The key principles behind Hadoop

Schema on Read vs Schema on Write

- Schema on Write (RDBMS)
  - Need to build a schema before brining the data
  - Data must fit the schema before being accessed
  - Schema must be updated before any new data is loaded

- Benefits
  - Fast read
  - Governance

- Schema on Read (Hadoop)
  - Data loaded directly to the file system
  - No transformation is needed
  - Late binding when specific column are read using SerDE (Serializer/Deserilizer)

- Benefits
  - Fast load
  - Agility & Flexibility

Conclusion

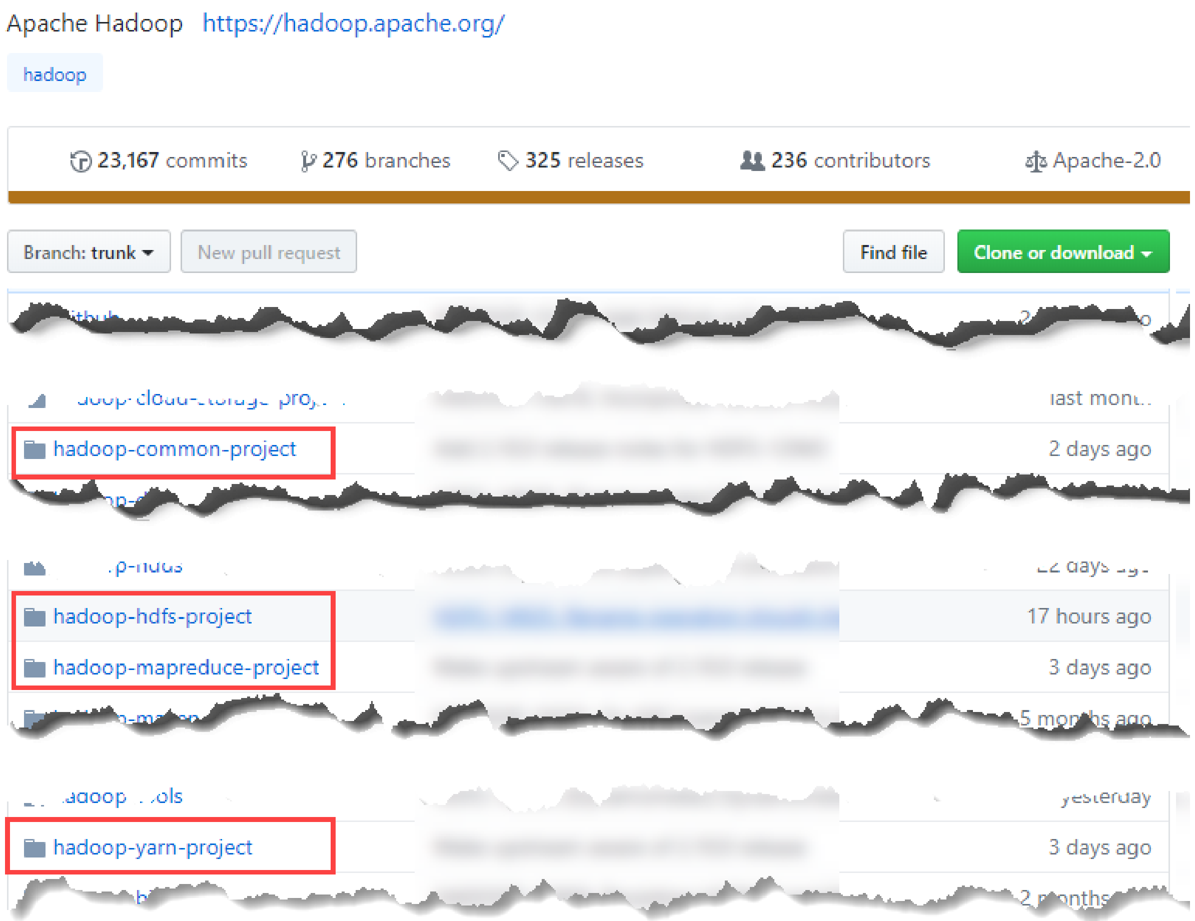
- The Hadoop platform will allow you to build your Large Scale Processing applications thanks to :
  - Distributed data storage
  - Distributed data processing
  - Fault Tolerance
  - Linear scalability
  - Hardware agnostic
  - And so much more..

- The platform / ecosystem is huge!

The Hadoop Core

The core subproject of Apache Hadoop

- Hadoop Common :
  - Utility services used by the other components
- Hadoop Distributed File System (HDFS)
- Hadoop MapReduce v2
- Hadoop YARN : Yet Another Resource Negotiator
  - Resource management and tasks scheduling

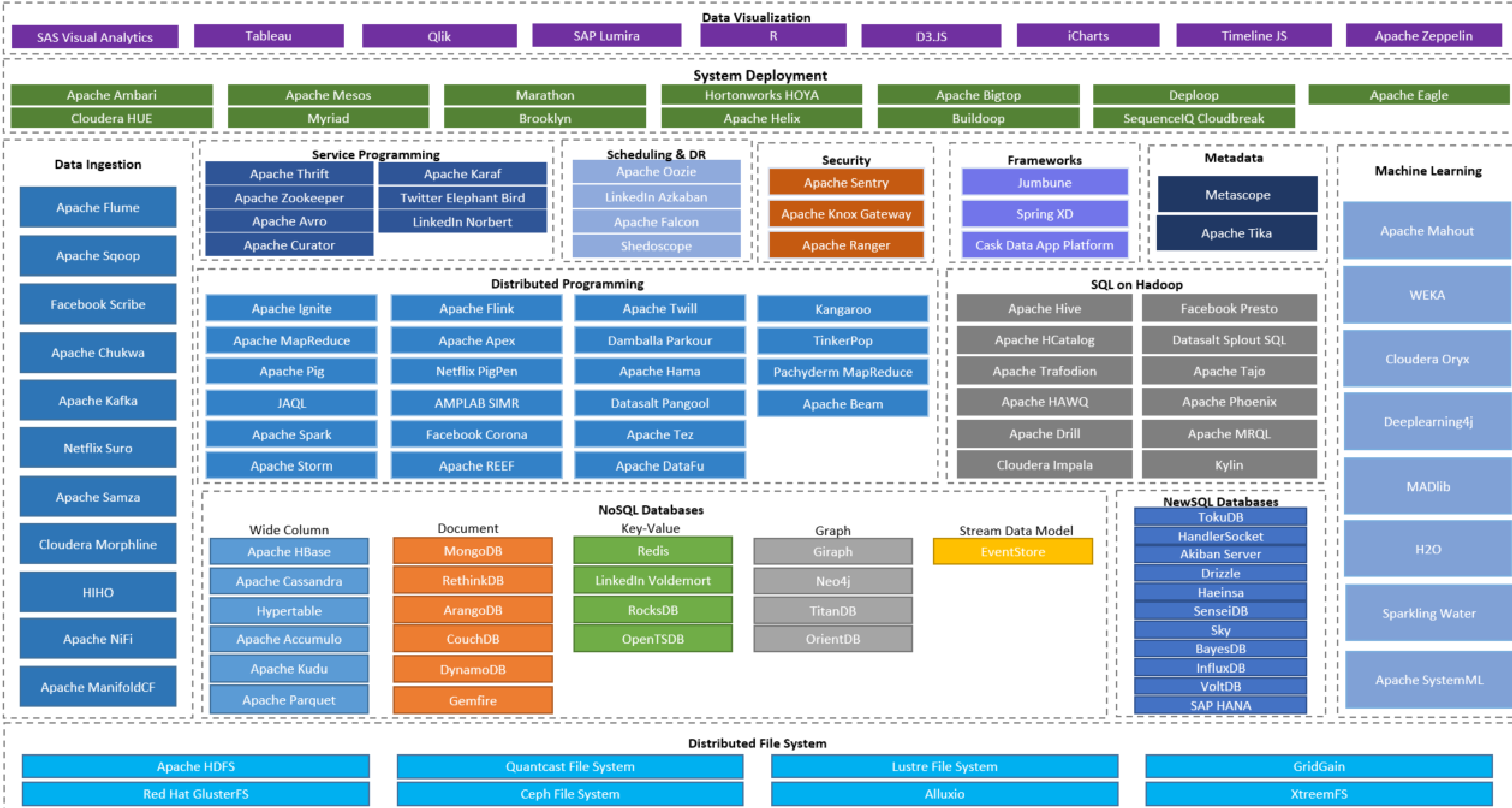


Apache Hadoop https://hadoop.apache.org/

hadoop-common-project	2 days ago
hadoop-hdfs-project	17 hours ago
hadoop-mapreduce-project	3 days ago
hadoop-yarn-project	3 days ago

An Overview of the Hadoop Ecosystem

Hadoop Ecosystem



Data Visualization

- SAS Visual Analytics
- Tableau
- Qlik
- SAP Lumira
- R
- D3.JS
- iCharts
- Timeline JS
- Apache Zeppelin

System Deployment

- Apache Ambari
- Apache Mesos
- Marathon
- Hortonworks HOYA
- Apache Bigtop
- Deploop
- Apache Eagle
- Cloudera HUE
- Myriad
- Brooklyn
- Apache Helix
- Buildoop
- SequenceIQ Cloudbreak

Data Ingestion

- Apache Flume
- Apache Sqoop
- Facebook Scribe
- Apache Chukwa
- Apache Kafka
- Netflix Suro
- Apache Samza
- Cloudera Morphline
- HIHO
- Apache NiFi
- Apache ManifoldCF

Service Programming

- Apache Thrift
- Apache Zookeeper
- Apache Avro
- Apache Curator
- Apache Karaf
- Twitter Elephant Bird
- LinkedIn Norbert

Scheduling & DR

- Apache Oozie
- LinkedIn Azkaban
- Apache Falcon
- Shedoscope

Security

- Apache Sentry
- Apache Knox Gateway
- Apache Ranger

Frameworks

- Jumbune
- Spring XD
- Cask Data App Platform

Metadata

- Metascope
- Apache Tika

Machine Learning

- Apache Mahout
- WEKA
- Cloudera Oryx
- Deeplearning4j
- MADlib
- H2O
- Sparkling Water
- Apache SystemML

Distributed Programming

- Apache Ignite
- Apache MapReduce
- Apache Pig
- JAQL
- Apache Spark
- Apache Storm
- Apache Flink
- Apache Apex
- Netflix PigPen
- AMPLAB SIMR
- Facebook Corona
- Apache REEF
- Apache Twill
- Damballa Parkour
- Apache Hama
- Datasalt Pangool
- Apache Tez
- Apache DataFu
- Kangaroo
- TinkerPop
- Pachyderm MapReduce
- Apache Beam

SQL on Hadoop

- Apache Hive
- Apache HCatalog
- Apache Trafodion
- Apache HAWQ
- Apache Drill
- Cloudera Impala
- Facebook Presto
- Datasalt Splout SQL
- Apache Tajo
- Apache Phoenix
- Apache MRQL
- Kylin

NoSQL Databases

Wide Column

- Apache HBase
- Apache Cassandra
- Hypertable
- Apache Accumulo
- Apache Kudu
- Apache Parquet

Document

- MongoDB
- RethinkDB
- ArangoDB
- CouchDB
- DynamoDB
- Gemfire

Key-Value

- Redis
- LinkedIn Voldemort
- RocksDB
- OpenTSDB

Graph

- Giraph
- Neo4j
- TitanDB
- OrientDB

Stream Data Model

- EventStore

NewSQL Databases

- TokuDB
- HandlerSocket
- Akiban Server
- Drizzle
- Haeinsa
- SenseiDB
- Sky
- BayesDB
- InfluxDB
- VoltDB
- SAP HANA

Distributed File System

- Apache HDFS
- Quantcast File System
- Lustre File System
- GridGain
- Red Hat GlusterFS
- Ceph File System
- Alluxio
- XtreemFS

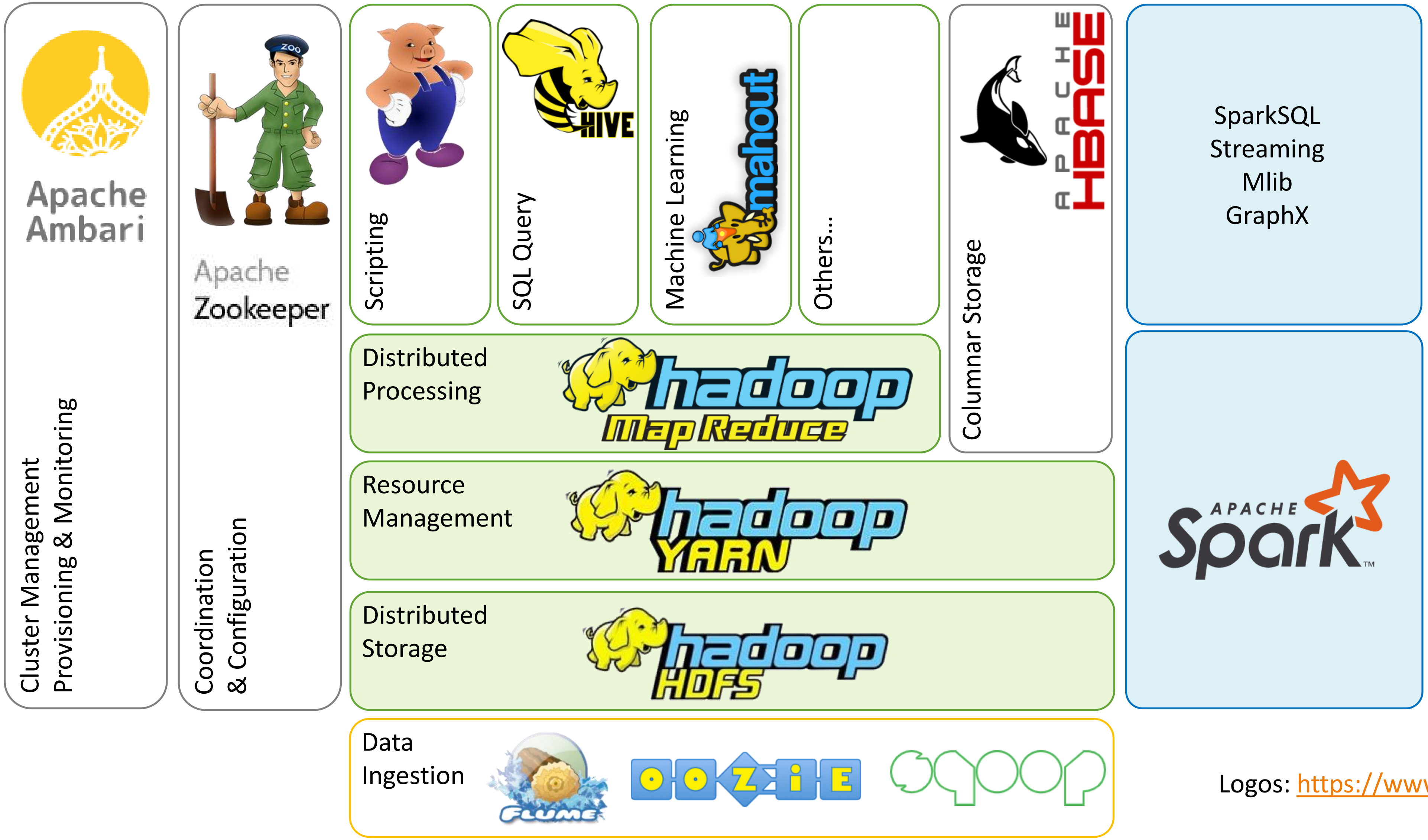
The role of ecosystem

- Data visualization
- System Deployment & Management
- Data Ingestion/Extraction
- Service Programming
- Scheduling
- Security
- Metadata extraction
- Machine Learning
- Distributed Programming

- SQL Querying
- Distributed Data Storage
- Database Models
  - Wide Column
  - Key-Value
  - Document
  - Graph
  - Stream
  - …

A Traditional Installation

A Simple View (of what you will try to use)



Cluster Management
Provisioning & Monitoring

Coordination
& Configuration

Scripting

SQL Query

Machine Learning

Others...

Columnar Storage

Distributed
Processing

Resource
Management

Distributed
Storage

Data
Ingestion

SparkSQL
Streaming
Mlib
GraphX

Logos: https://www.apache.org/logos

A Traditional Installation

- Apache Ambari
  - Enables system administrators to provision, manage and monitor a Hadoop cluster
- Apache ZooKeeper
  - provide a distributed configuration & synchronization service and naming registry
- Apache Pig
  - high-level language for expressing data analysis programs
- Apache Hive
  - a SQL-like interface to query and analyze data
- Apache Mahout
  - scalable machine learning algorithms
- Apache HBase
  - Bigtable-like capabilities on top of Hadoop
- Apache sqoop
  - Bulk data transfer with structured datastores such as relational databases
- Apache oozie
  - Workflow scheduler system to manage Apache Hadoop jobs
- Apache Flume
  - Collecting, aggregating, and moving large amounts of log data based on streaming data flows

A Traditional Installation

- Apache Spark
  - Uses resilient distributed dataset (RDD), a distributed read-only multiset of data
  - Spark Core:
    - provides distributed task dispatching, scheduling, and basic I/O functionalities, exposed through an API
  - Spark SQL
    - provides a domain-specific language (DSL) to manipulate DataFrames & SQL language support
  - Spark Streaming
    - ingests data in mini-batches and performs RDD transformations
  - MLlib Machine Learning Library
  - GraphX
    - distributed graph-processing framework

The Traditional Data Processing Approaches

Data Processing approaches

- **Batch Processing**
  - **Input**: Analyze all available data at a certain point in time
  - **Results**: The results is available at the end of the process
  - **Latency**: The processing time can be long (minutes to hours or days)
- **Micro-batch**
  - **Input**: Analyze all available data at a certain point in time every x seconds (smaller files or chunks)
  - **Results**: The results is available at the end of the process
  - **Latency**: The processing time is short (in seconds)
- **Real Time**
  - **Input**: Analyze every new data made available (streams)
  - **Results**: The results is available for each new data
  - **Latency**: The processing time is really short (in milliseconds)

The Hadoop Distributions

Open Source, but not only!

- Hortonworks
  - Complete open source Apache Hadoop distribution without additional proprietary software
- Cloudera
  - The core Apache Hadoop distribution with proprietary tools to automate the installation process and other services (monitoring)
- MapR
  - HDFS was replaced by MapRFS to enable more efficient management of data, reliability and ease of use

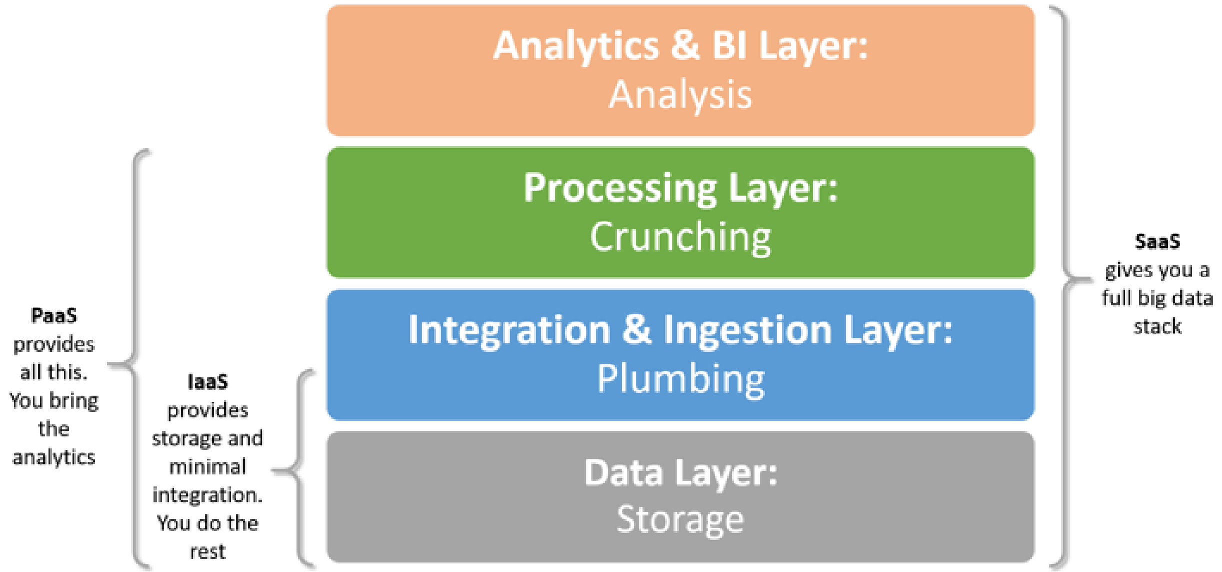
But they all contribute back to the open source project!

Also In the Cloud

- IaaS : Just compute!
  - Like on-premise but with someone else compute
  - Example: AWS EC2 & EBS

- PaaS : Cloud services to manage your cluster
  - The deployment is fully managed, you don't manage compute anymore
  - Example: Azure HDInsight, AWS EMR & Redshift

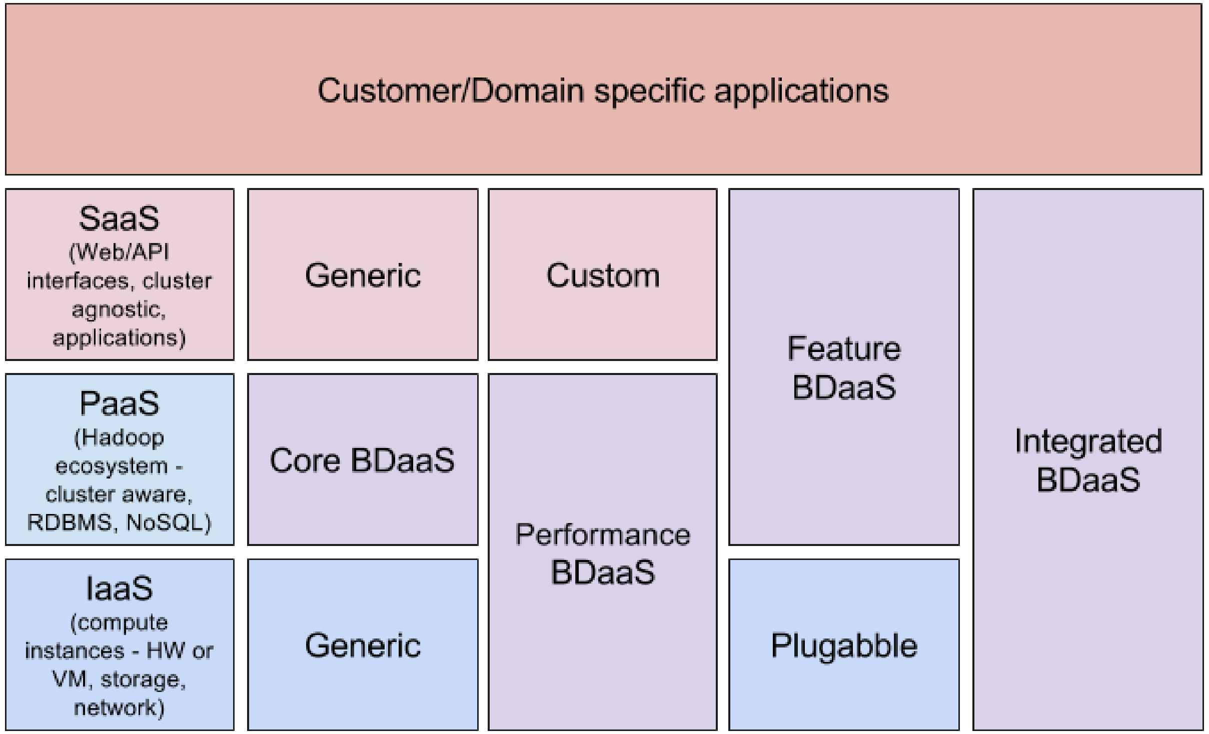
- SaaS: The full stack (Big Data + Analytics)
  - You don't care how your data is stored or processed
  - You just use API or tools to access it
  - Examples: Snowflake, Looker


Analytics & BI Layer:
Analysis
Processing Layer:
Crunching
Integration & Ingestion Layer:
Plumbing
Data Layer:
Storage
PaaS provides all this. You bring the analytics
IaaS provides storage and minimal integration. You do the rest
SaaS gives you a full big data stack

Source: https://hub.packtpub.com/big-data-as-a-service-bdaas-solutions-comparing-iaas-paas-and-saas/

What About BDaaS? A new emerging trend

- Core BDaaS :
  - implement the minimal Hadoop core & a few popular services
- Performance BDaaS:
  - Includes an optimized infrastructure with specifically build hardware
  - Example: Altiscale
- Feature BDaaS
  - Include features beyond the Hadoop core.
  - Example: Qubole
- Integrated BDaaS
  - Combines both the performance and feature



Customer/Domain specific applications

SaaS (Web/API interfaces, cluster agnostic, applications)

Generic

Custom

Feature BDaaS

Integrated BDaaS

PaaS (Hadoop ecosystem - cluster aware, RDBMS, NoSQL)

Core BDaaS

Performance BDaaS

IaaS (compute instances - HW or VM, storage, network)

Generic

Plugabble

Source: https://www.semantikoz.com/blog/big-data-as-a-service-definition-classification/

ODPi – Open Data Platform Initiative

The Context

- The Apache Hadoop framework and ecosystem has grown really fast:
  - A lot of innovation (streaming, machine learning…)
  - But also a lot of overlap
  - Some frameworks had a short life
- This is causing a slow down in the adoption!
  - People don’t know what they should use for their use case
  - People don’t know if the framework will be maintained or evolve

ODPi: Toward Simplification & Standardization?

- The Open Data Platform initiative is a nonprofit organization (but driven by industry/vendors veteran)

- Committed at creating and standardizing big data solutions that work across platforms, systems and products to:
  - Reduce redundancy, overlap and complexity
  - Ease implementation

- Uses a common reference specification called: ODPi Core

Source: https://www.odpi.org/

Hadoop : ODPi Runtime Compliant

- In June 2016, Apache Hadoop, version 2.7, has become ODPi compliant
- ODPi complements and reinforce the Apache Software Foundation work
- The validation test suite guarantees that a distribution of Apache Hadoop complies with the ODPi-defined specifications.
- The test framework and self-certification aligns closely with the Apache Software Foundation by leveraging Apache BigTop for comprehensive packaging, testing, and configuration.

Source: https://www.odpi.org/announcements/2016/06/27/apache-hadoop-distributions-now-odpi-runtime-compliant
https://github.com/odpi/specs/blob/master/ODPi-Runtime.md

Summary

- Hadoop is relatively young but yet mature
- Key principles: distributed, open, extendable, scalable & reliable
- The Hadoop Core: HDFS, MapReduce and YARN
- The type of data processing
- The Hadoop distributions, on-premise, in the cloud, IaaS, PaaS, SaaS & BDaaS
- The ODPi