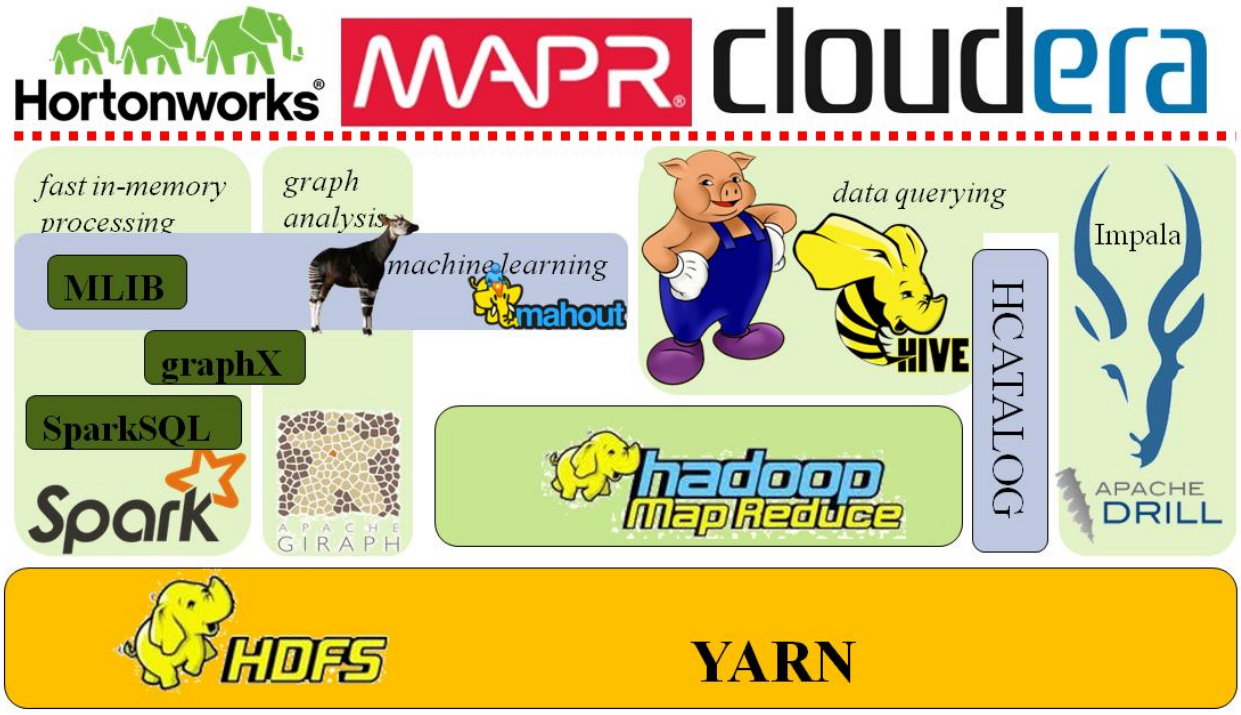
The Hadoop Ecosystem

Hadoop has evolved from just a MapReduce clone to a platform with many different tools that effectively has become the “operating system” for Big Data clusters. This short overview lists the most important components.



Companies

As of 2015, there are three companes battling to be the dominant distributor for Hadoop, namely Cloudera, Hortonworks, and MapR. There is an analogy with Linux Distributors (RedHat, SuSE, Ubuntu) in that one can consider Hadoop the open-source "operating system" for Big Data clusters; just like Linux is the open-source operating system for enterprise computers. HortonWorks and Cloudera seem to be in the lead; they distribute the standard Apache Hadoop software, of course customized in different ways and packaged with slightly different sets of tools. Some of these extra tools and GUIs are not open source and the business model of these companies is based on charging for support subscriptions. MapR is a bit different, as it built a C++ clone of Hadoop and HDFS (not in Java), so it does not distribute the standard Apache Hadoop software, and it touts its own copy as more efficient. Even though these companies are young of age, their economic power is non-negligible, being valued at billions of dollars worth and for instance Cloudera receiving an $800M investment from Intel Corp. alone in 2014.

Basic Software Components

HDFS

The Hadoop Distributed File System, is an open-source clone of the Google File System, and was originally funded by Yahoo. It provides cheap and fault-tolerant storage and therefore is the backbone of the whole of Hadoop. It consists of a *namenode*, a single process on a machine which keeps track of where all HDFS blocks are, and many *datanodes* that store the data. The data is kept replicated (default 3x) and as soon as a datanode appears to be dead, the namenode initiates a re-replication towards another datanode to remain fault tolerant.

MapReduce

MapReduce is framework to run distributed computations where you have write just two functions (Map and Reduce) in some programming language (typically Java or Python). Invented by Google, Hadoop is the open-souce MapReduce implementation. The input and output of MapReduce programs are HDFS files. Also, the Map and Reduce faces communicate data over the network by writing to HDFS and reading this data from other nodes. Huge datasets can be analyzed reliably using the user's Map and Reduce function as the MapReduce framework automatically runs many mapper and reducer jobs on the cluster, on *splits* on the input files. MapReduce was a breakthrough in programmer productivity on a large cluster, and the initial reason why Google could out-engineer its competitors in its early days.

YARN

Resource manager to share the hardware in a cluster in a coordinated manner between MapReduce and other tools. YARN determines which jobs run when on which machines and why. It keeps track of which machines have CPU and memory free. It looks for each job at where the data that it needs resides, to make this decision, trying to run jobs close to the data. This functionality used to be just a part of the first versions of MapReduce, but it was separated out in a separate service because nowadays there are so many other tools active on Hadoop clusters, generating jobs, besides MapReduce (Impala, Spark,..).

HCATALOG

Meta-data repository for registering datasets available on HDFS (it stand for: Hive Catalog). These datasets can be simple text files (CSV) in some HDFS directory, or more typically many such files forming one dataset. It can also be data in many other Hadoop data formats such as SequenceFile, Avro, RC, ORC, Parquet (roughly in order of sophistication).

HCATALOG started out as the database catalog in Hive: a repository of all tables (datasets), their names, columns, locations, etc. But, all other tools (e.g. Impala, Spark) wanted to be compatible with it and thus it evolved as the central meta-data repository of Hadoop clusters.

Spark

Spark is a new cluster computing framework that seeks to replace MapReduce. Whereas in MapReduce the Mappers and Reducers communicate through files written on HDFS, Spark in principle does not write to files and keeps all data in memory. Without HDFS, resilience to failure in it computations is realized through its concept of Resilient Distributed Datastores (RDD). RDD are essentially datasets, or

recipes to (re-)create a dataset from other RDDs. This means that if data get lost, Spark recomputes the lost parts using the recipe. Further, Spark supports many more programming concepts other than Map and Reduce (group-by), for instance also joins. It embraces modern *functional* programming languages, in particular Scala. Functional languages are generally useful for writing parallel and distributed programs. Spark itself is an expanding mini-ecosystem with for instance SparkSQL (a competitior to Hive) and the scalable data mining library MLIB (a competitor to Mahout) .

Spark is not really attempting to replace Hadoop completely, and it is most often used inside Hadoop, or rather, on top of HDFS for storage and in conjunction with YARN for resource scheduling.

Data Querying

Pig

Many problems can be solved more easily by writing a query script rather than by writing MapReduce functions in Java or Python. Pig is a system that executes scripts that process large data sets in steps. These steps roughly correspond with Relational Algebra (filter, group-by, join) operations. Such Pig scripts are translated automatically in MapReduce functions that get executed in Hadoop.

Hive

Whereas in Pig the user has control over the order of the operations (e.g. first read data, compute a value, filter, group-by and aggregate) such problems can be solved already by SQL systems (e.g, a SELECT..FROM..GROUP BY query). Hive is a system that translates SQL queries in MapReduce functions. Hive has been criticized because it is relatively slow, because executing a query using MapReduce jobs implies that all intermediate results are exchanged by writing them to disk (HDFS). Therefore, currently efforts are underway to avoid MapReduce and make Hive faster.

Impala, Drill, PrestoDB

The great potential of SQL on MapReduce is to be able to use the many existing SQL-based applications and tools already out there on Hadoop. However the high query startup latency and low speed of processing of Hive, makes it hard to run such applications interactively. Therefore, a number of initiatives got off the ground that attempt to create a fast analytical database system on Hadoop.

Cloudera funds the development of Impala, an efficient database system designed for Hadoop. It introduced its own column-oriented compressed table file format, called Parquet. Its execution engine uses just-in-time compilation to machine code. In other words, it uses the techniques that have become the hallmarks of modern analytical database systems such as MonetDB, HANA and IBM-BLU.

Apache Drill is efficient SQL system with a similar approach to Impala; it is funded mostly by MapR.

PrestoDB is yet another efficient SQL system originally intended to replace Hive at Facebook.

SparkSQL

The Spark framework (see above) also comes with its own SQL front-end named Spark SQL. Its main advantage is currently the tight integration with the other Spark components. Spark SQL as of yet is rather immature, for instance its query optimizer often cannot process queries or runs them with a bad plan making the system slow again.

Given the young age of all these SQL systems, they are immature, certainly compared to seasoned products such as Oracle, DB2, etc. The query optimizer is often a weak spot, which can lead to certain queries going taking forever, without good reason.

Graph Processing

There are quite a few Big Data analysis problems where (social) graphs, or networks must be analyzed. For instance Google analyzes the Web (a huge graph) and Facebook analyzes the interactions between people (a huge friendship graph). Algorithms that analyze graphs are often iterative, repeating an analysis multiple times, each time adjusting the results based on the results of the previous iteration. For instance, this is the way Google computes the PageRank of all web pages. Such iterative jobs would need many MapReduce jobs, each reading and writing to HDFS. Google itself proposed a system called Pregel, to do these kinds of computations more easily, keeping the data in memory between all iterations, and this idea got cloned (mostly at Facebook) into an open-source project called Giraph.

As already noted, the Spark system also keeps data in memory when possible, and its Scala programming interface allows to easily write programs that perform iterative analysis. As such, Spark is quite suited for these kinds of computations as well. There is also a small wrapper called GraphX on top of Spark that supports Giraphs's so-called *vertex-centric* programming model on Spark.

Machine Learning

In Big Data often the data is raw and variable (e.g. text) and we need to perform machine learning on this raw input to gain valuable information. Whereas machine learning tools in the past used to fit on a single server, the machine learning tools in Big Data need to be able to parallelize the work on clusters.

Mahout is a library of machine learning algorithms written for MapReduce, however since machine learning algorithms are iterative this need many MapReduce jobs. Spark avoids HDFS, makes it easy to write iterative analysis and introduced its own scalable machine learning library called MLIB.

Some machine learning algorithms focus on learning graph structures. Giraph has its own library of graph algorithms called Okapi. Finally, another scalable system for machine learning on graphs is called GraphLab, though this is not necessarily Hadoop centered.