What the heck are data clusters and Hadoop?

medium.com/it-s-a-data-world/what-the-heck-are-data-clusters-and-hadoop-60aa4477cf82

I recently started working for Dataiku, a French Startup making predictive analysis accessible to everyone. Since I've arrived I've had to cozy up to all the technical jargon to try and figure out what my coworkers where talking about.

Luckily, in the past month, I've had the opportunity to speak to all of our brilliant data scientists and developers, as well as a couple of data experts. I've learned so much about Data Science and Big Data infrastructure, picking up words like Hadoop, Hive, Spark, Mesos, or Daze along the way. So I figured I could share my investigation into understanding the world of Big Data with people out there who are just as clueless as I was.

Today, I'll begin my journey into big data by talking about distributed data storage and Hadoop. It's probably a bad place to start but hey, you gotta start somewhere.

Btw, I originally wrote about this over here.

It all begins with a definition

The first step to understanding Hadoop and distributed data storage was of course googling it, and checking Wikipedia:

> "**Apache Hadoop** is an open-source software framework written in Java for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware. All the modules in Hadoop are designed with a fundamental assumption that hardware failures (of individual machines, or racks of machines) are commonplace and thus should be automatically handled in software by the framework."

Suffices to say that was not helpful. Luckily we have lots of people at Dataiku who took some time to clear things up for me.

The first part of the definition is that Hadoop is an open source cluster computing framework. That means that it's used to process data that is distributed across clusters. So step 1 is understanding data clusters.

Why do you distribute data storage?

The general idea is that when you start collecting, storing, and analysing really large quantities of data, storing it on one machine makes it really inefficient if your data doesn't fit in your machine's RAM (Random Access Memory—a part of your computer's storage that's accessible fast but volatile).

You also don't want to have all of your data stored in just one place because if that one computer goes down you lose everything. And one of the rules when working with computers is that they always crash. Moreover, the more data you have stored in one server, the longer it takes to read through it to find the relevant information.

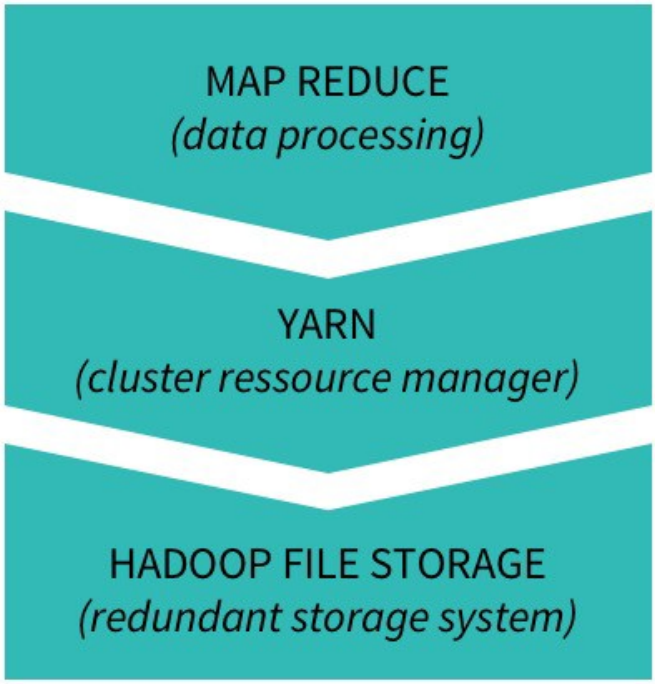
The origins of Hadoop

These issues first started preoccupying Google, whose business model is based on storing huge amounts of data and having access to all of it at one time. Makes sense. Their system was adapted into the Hadoop Distributed File System (HDFS for experts).

Yes, Hadoop has one of the cutest logos in Big Data.

The basis of the system is to store the data on a cluster of servers. Instead of having to invest in a really large server, the system functions on the hard drives of lots of computers that you can compare to your PC. This system is infinitely scalable.

The data is stored redundantly (meaning the same information is written on several different servers, usually 3), so that the system isn't weakened by one server going down. The data is processed with the processing system Hadoop MapReduce which operates over the cluster resource manager Hadoop YARN. MapReduce can take distributed data, transform it (Map), and aggregate it (Reduce) into other useful data.

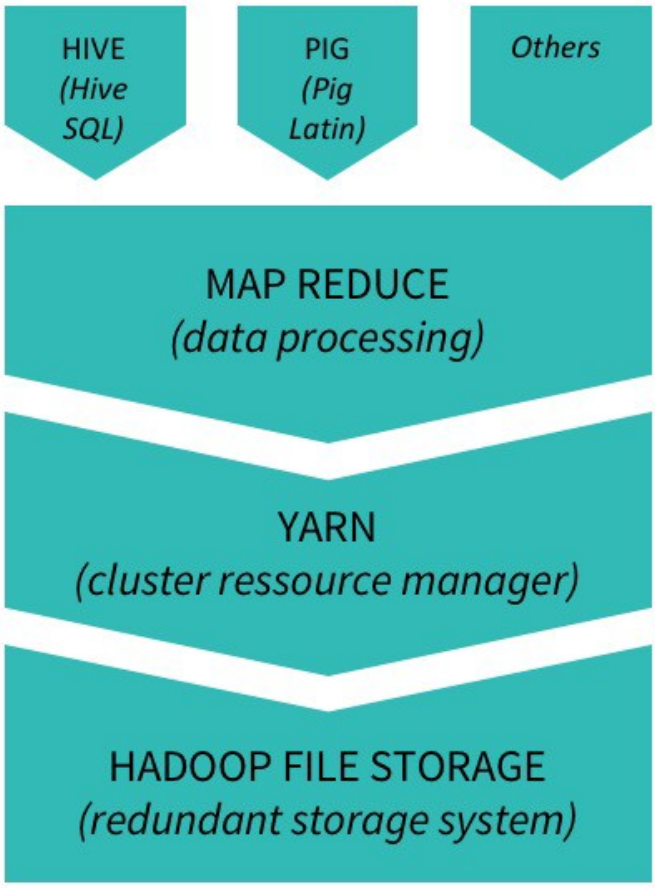


An easy Hadoop Stack

The cluster of servers is constituted of a client server, a driver node, and worker nodes (they're also often called master nodes and slave nodes, but that's not super politically correct). A node is a server by the way, which is basically a computer. I am grossly simplifying but the driver node distributes jobs and tracks them across the worker nodes who execute the jobs. The client server is where HDFS, MapReduce, and YARN are installed. Data is loaded into the client server when it is written by HDFS or processed by MapReduce, then it is written on the worker nodes. And YARN manages the resource allocation or the jobflow. Basically, YARN is da boss.

Whenever you write a new record or when you make a query, the driver clusters know exactly where to write or where to find the info in the worker clusters thanks to the metadata ("data about data") stored by HDFS.

There are then a whole lot of layers on top of MapReduce as well, like Hive that allows you to write SQL queries and Pig to code in Pig Latin. This is important because MapReduce is in Java and pretty hard to program without those additions.



A little bit more advanced Hadoop Stack.

Hadoop rules

The Hadoop ecosystem allows you to store large amounts of data and access it faster. It's cheaper to set up than a ginormous server and also much faster for processing the data. Indeed, when you multiply the number of machines, you're exponentially multiplying the processing speed as well as the number of jobs you can run at the same time. The redundancy principle built in HDFS means the system is protected against server failures.

And that’s it for today! Stay tuned for part 2 of my investigation into the Data World: when Hadoop-the-great meets a new competitor, Spark-the-underdog, in an epic battle for big data domination.