Hadoop & Map Reduce

Metis Data Science Bootcamp

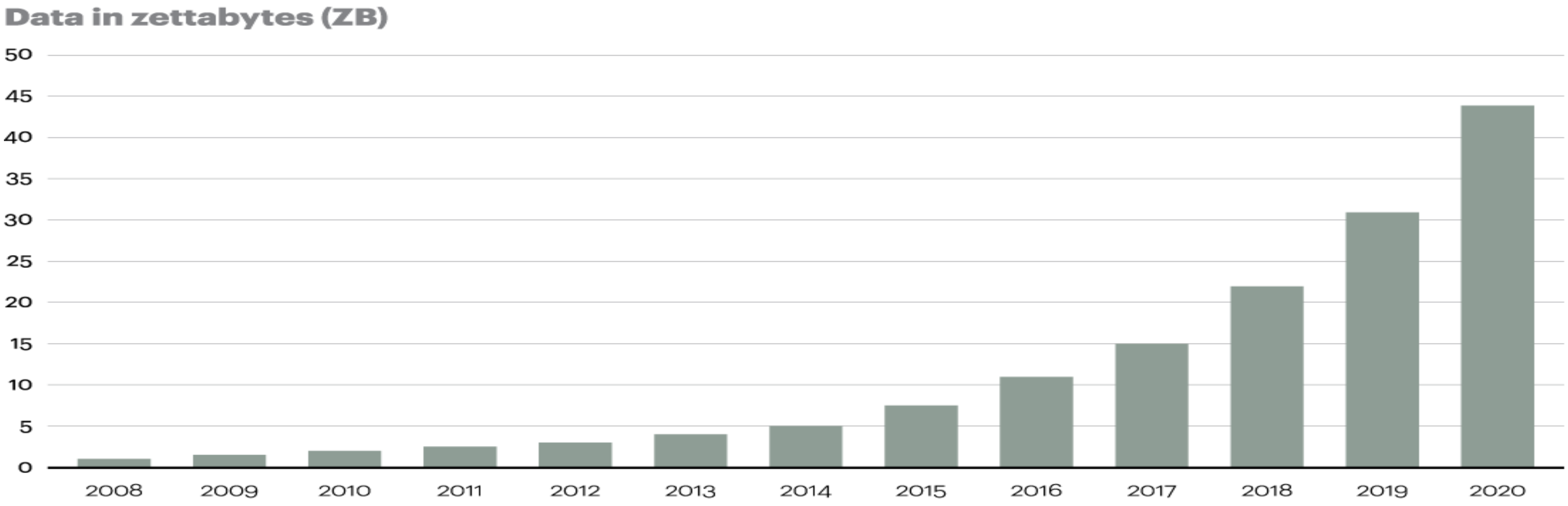
Contents

- What are: Big Data and Hadoop?
- Advantages of Hadoop
- Hadoop Distributed Filesystem (HDFS)
- The MapReduce framework
- The Hadoop environment and evolution

Big Data

Figure 1

Data is growing at a 40 percent compound annual rate, reaching nearly 45 ZB by 2020

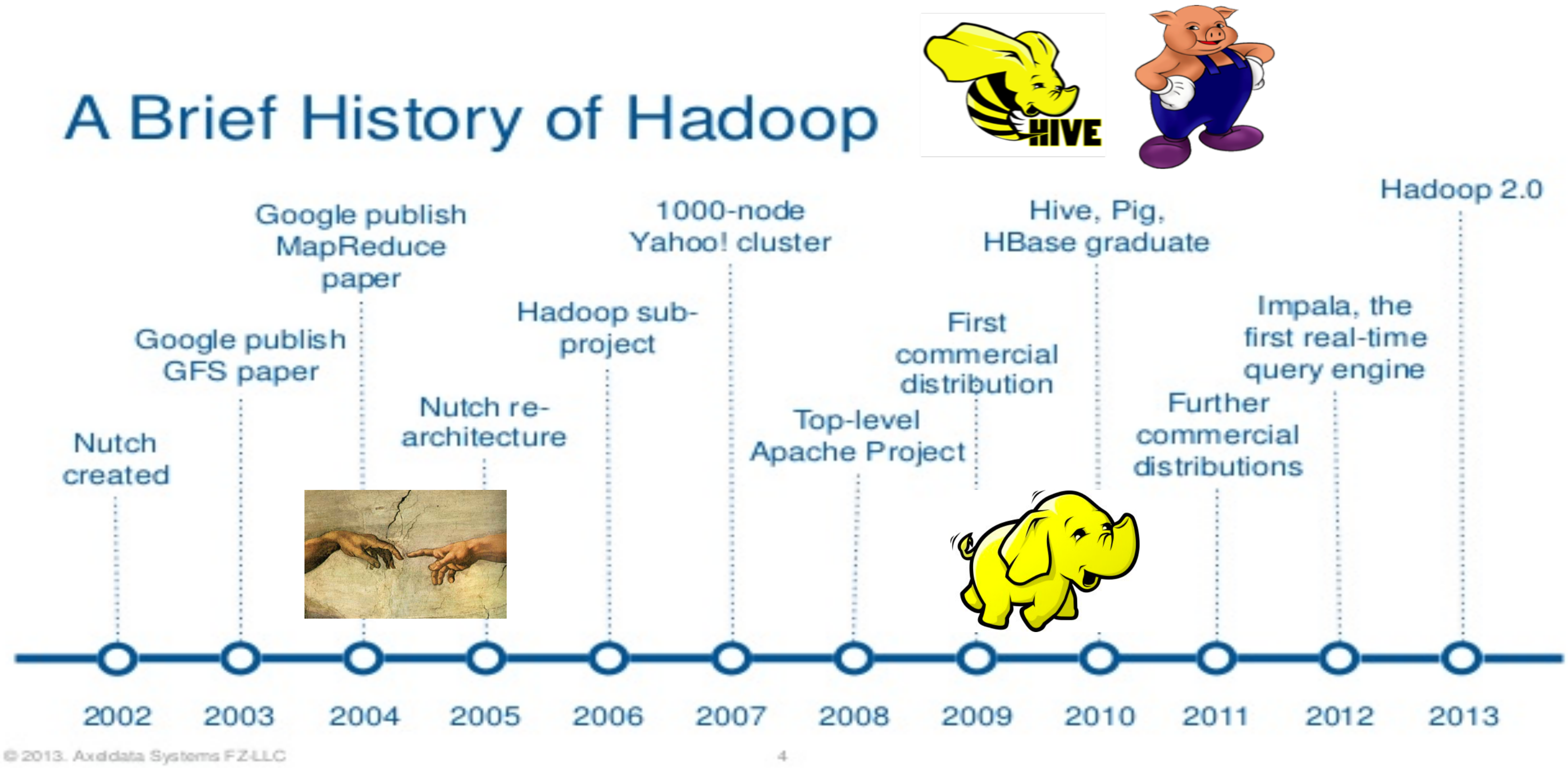


Source: Oracle, 2012

What is big data?

- Big data problems are those that can not be solved using traditional tools because of :
  - The volume of data (TB/ PB)
  - The variety of data

Some history



A Brief History of Hadoop

Nutch created

Google publish GFS paper

Google publish MapReduce paper

Nutch re-architecture

Hadoop sub-project

1000-node Yahoo! cluster

Top-level Apache Project

First commercial distribution

Hive, Pig, HBase graduate

Further commercial distributions

Impala, the first real-time query engine

Hadoop 2.0

2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013

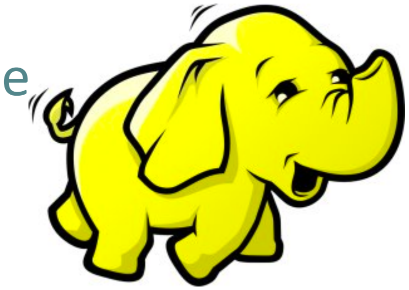
Hadoop = HDFS + MapReduce

- Hadoop Distributed Filesystem (HDFS)
  - Files sitting on different machines, but they behave like a single file system
  - This system is optimized for fault tolerance
- MapReduce
  - programming model for parallel processing
  - The implementation also makes it fault tolerant.

Hadoop Framework

The base Hadoop framework is composed of the following modules:

- ***Hadoop Common*** – contains libraries and utilities needed by other Hadoop modules;
- ***Hadoop Distributed File System (HDFS)*** – a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster;
- ***Hadoop YARN*** – a resource-management platform responsible for managing computing resources in clusters and using them for scheduling of users' applications
- ***Hadoop MapReduce*** – a programming model for large scale processing.



Hadoop

- Hadoop is a framework

Hadoop

- Hadoop is a framework
- for distributed processing

Hadoop

- Hadoop is a framework
- for distributed processing
- of large data sets

Hadoop

- Hadoop is a framework
- for distributed processing
- of large data sets
- across a cluster of many computers

Hadoop

- Hadoop is a framework
- for distributed processing
- of large data sets
- across a cluster of many computers
- that implements map and reduce functions

Hadoop

- Hadoop is a framework
- for distributed processing
- of large data sets
- across a cluster of many computers
- that implements map and reduce functions
- using a distributed filesystem (HDFS)

Hadoop Distributed Filesystem (HDFS)

Hadoop Distributed Filesystem (HDFS)

- breaks data into blocks, replicates and stores these blocks across different nodes in a cluster

Hadoop Distributed Filesystem (HDFS)

- breaks data into blocks, replicates and stores these blocks across different nodes in a cluster
- master-slave architecture

Hadoop Distributed Filesystem (HDFS)

- name node (master): maintains name system, manages data blocks

Hadoop Distributed Filesystem (HDFS)

- name node (master): maintains name system, manages data blocks
- data nodes (slave): deployed storage machines. MapReduce analyses happen here.

Hadoop Distributed Filesystem (HDFS)

- name node (master): maintains name system, manages data blocks
- data nodes (slave): deployed storage machines. MapReduce analyses happen here.
- secondary name node: performs periodic checkpoints (restart the name node from checkpoint in case of name node failure)

Hadoop Distributed Filesystem (HDFS)

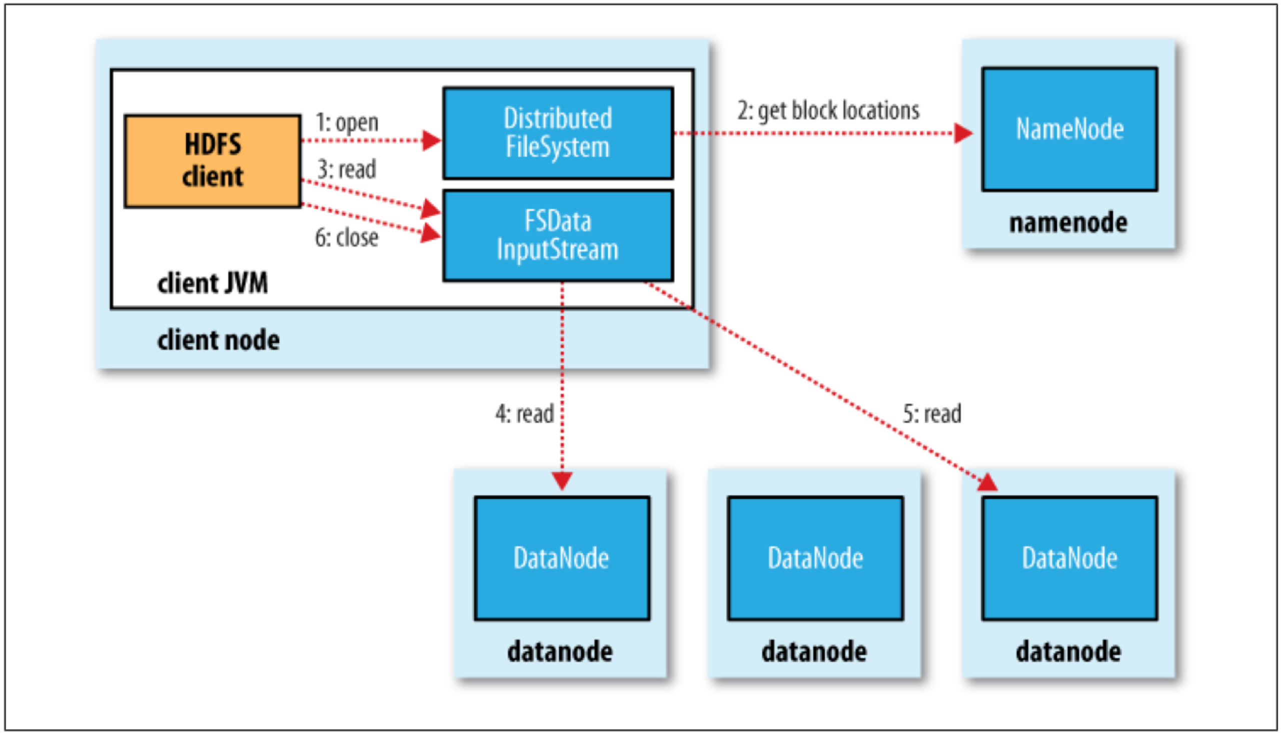


Figure 3-2. A client reading data from HDFS

MapReduce

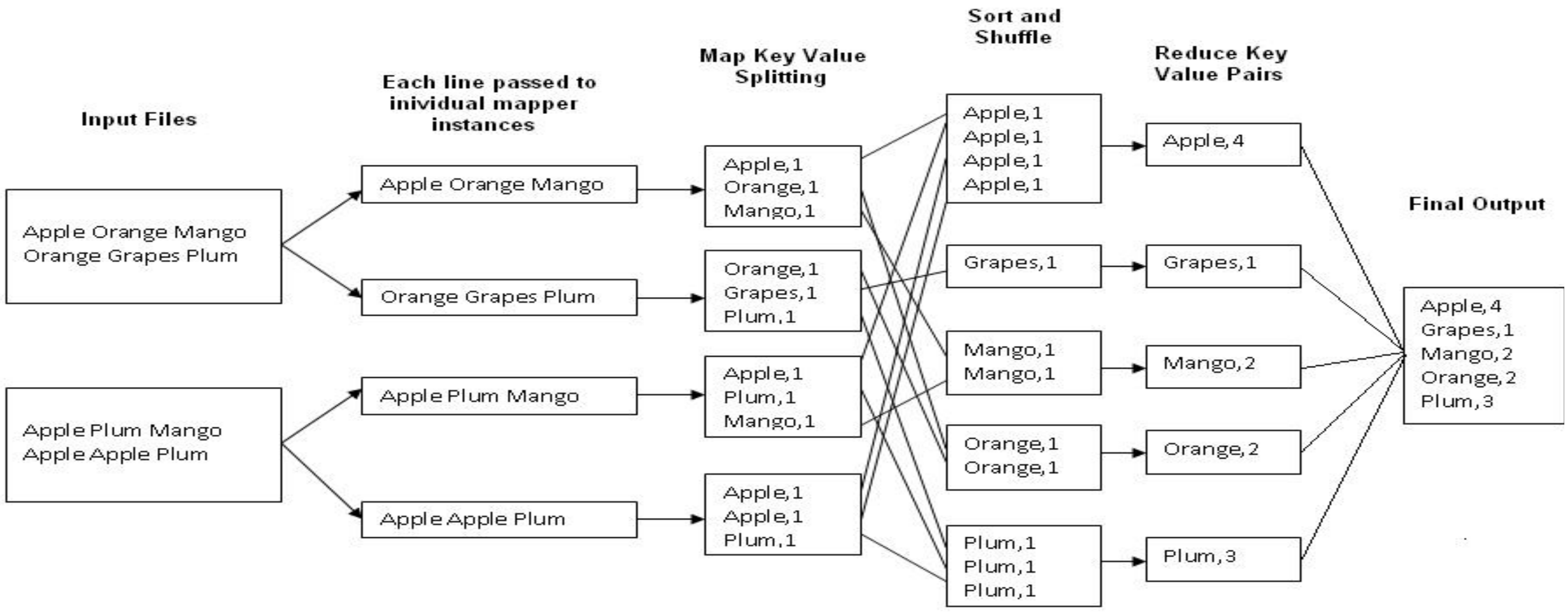
MapReduce

- MapReduce = Map + Reduce functions

MapReduce

- MapReduce = Map + Reduce functions
  - MAP function
    - Input: Loads raw data (from HDFS)
    - Filter, transform, parse
    - Output: (key, value) pairs
  - REDUCE function
    - Automatically Groups by the mapper's output key
    - Aggregate, count, etc
    - Output: to HDFS

MapReduce



Input Files

Apple Orange Mango
Orange Grapes Plum

Apple Plum Mango
Apple Apple Plum

Each line passed to inividual mapper instances

Apple Orange Mango

Orange Grapes Plum

Apple Plum Mango

Apple Apple Plum

Map Key Value Splitting

Apple,1
Orange,1
Mango,1

Orange,1
Grapes,1
Plum,1

Apple,1
Plum,1
Mango,1

Apple,1
Apple,1
Plum,1

Sort and Shuffle

Apple,1
Apple,1
Apple,1
Apple,1

Grapes,1

Mango,1
Mango,1

Orange,1
Orange,1

Plum,1
Plum,1
Plum,1

Reduce Key Value Pairs

Apple,4

Grapes,1

Mango,2

Orange,2

Plum,3

Final Output

Apple,4
Grapes,1
Mango,2
Orange,2
Plum,3

MapReduce

- Master-slave architecture
  - Master: jobtracker
    - coordinator, scheduler
    - reassigns failed jobs
  - Slave: tasktracker
    - run tasks, send reports to jobtracker
    - lack of report → failure

MapReduce

- Data locality
  - “Push the computation to the data”
  - Mapper code is sent to all data nodes and run locally
  - Thus no data is moved over the network → faster.

is that all there is to map reduce?

no!

- **map function**: turns data into (key, value) pairs
- **reduce function**: reduce all values for key to one value or set

is that all there is to map reduce?

no!

- input reader: gets data from file system
- **map function**: turns data into (key, value) pairs
- **reduce function**: reduce all values for key to one value or set

is that all there is to map reduce?

no!

- input reader: gets data from file system
- **map function**: turns data into (key, value) pairs
- **reduce function**: reduce all values for key to one value or set
- output writer: write output to file system

is that all there is to map reduce?

no!

- input reader: gets data from file system
- **map function**: turns data into (key,value) pairs
- partition function: assign keys to reducer servers
- **reduce function**: reduce all values for key to one value or set
- output writer: write output to file system

is that all there is to map reduce?

no!

- input reader: gets data from file system
- **map function**: turns data into (key,value) pairs
- compare function: sort keys, collecting pairs with the same keys
- partition function: assign keys to reducer servers
- **reduce function**: reduce all values for key to one value or set
- output writer: write output to file system

is that all there is to map reduce?

no!

- input reader: gets data from file system
- **map function**: turns data into (key, value) pairs
- compare function: sort keys, collecting pairs with the same keys
- combiner function: reduce network traffic
- partition function: assign keys to reducer servers
- **reduce function**: reduce all values for key to one value or set
- output writer: write output to file system

What makes Hadoop Awesome !

(Design Concepts)

- Distribution:
  - Instead of using a supercomputer, storage and processing are spread across a cluster of smaller machines

What makes Hadoop Awesome
(Design Concepts)

- Distribution
- Deals with hardware failure
  - With multiple machines, server failure and error rates increase dramatically: i.e. at least once/day
  - Hadoop expects these failures, and deals with them
  - Node recovery: nodes can get their act together and rejoin the party without full restart

What makes Hadoop Awesome !
(Design Concepts)

- Distribution
- Deals with hardware failure
- Linear scalability (Speed!)
  - read/write in parallel: 100 TB, 75MB/sec HD
  - 1 machine:
    - 75MB/sec → 16 days
  - 1000 machines:
    - 75,000MB/sec = 75 GB/sec →22 minutes

What makes Hadoop Awesome !
(Design Concepts)

- Distribution
- Deals with hardware failure
- Linear scalability (Speed!)
- Fault tolerance
  - Even if a process fails, it's a small part and not the entire MapReduce job
  - Data recovery: one node can pick up workload of another
    - HDFS stores each data block on 3 machines by default

What makes Hadoop Awesome !
(Design Concepts)

- Distribution
- Deals with hardware failure
- Linear scalability (Speed!)
- Fault tolerance
- Data locality
  - Don't move large datasets to where the application is running, but run the application to where the data already is!

Hadoop environment

- **Mahout:** a data mining library (clustering, regression, & statistical modeling & implements via MapReduce)
- **Hive**: takes over management of HDFS storage, interprets & runs jobs via SQL queries
- **Pig**: write tasks more easily (for MapReduce+HDFS)
- **HBase**: take HDFS and add some good stuff from relational databases (make hadoop more SQL-like)
- **Sqoop**: import SQL →HDFS
- **Avro**: serialization (allows for encoding schema of Hadoop files)

Hadoop environment

- Flume – stream logs into Hadoop
- Whirr – cloud platform libraries
- HCatalog – treat disparate data storage as one thing
- MRUnit – unit testing for Hadoop
- BigTop – attempt to re-define "core Hadoop"
- Oozie – workflow to integrate Hadoop + other tasks

Apache Spark

- Popularized in ~ 2013-2014
  - load all the data into memory
  - up to ~100 times faster than MapReduce
  - iterative algorithms become reasonable!
- Popular configuration:
  - HDFS / S3
  - YARN / Mesos
  - ~~MapReduce~~ Spark

Try It!

- Hadoop_Setup.md (Set Hadoop up on your cloud)