Hadoop & Map Reduce

Contents

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

Big Data

What is big data?

What is big data?

- a bunch of HYPE!

… and get off my lawn!

Gary Bernhardt
@garybernhardt

Follow

Consulting service: you bring your big data problems to me, I say "your data set fits in RAM", you pay me $10,000 for saving you $500,000.

RETWEETS 812

FAVORITES 757

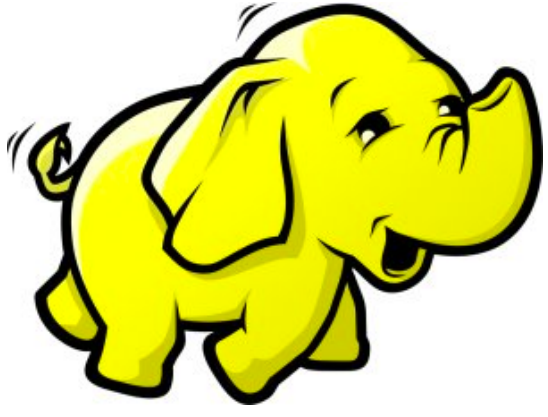
5:03 PM - 19 May 2015

What is big data?

- Various definitions:
  - Data that cannot be stored in more traditional relational databases
  - on the order of high millions to billions of records, high thousands to millions of columns, TB to PB

What is Hadoop?

- Hadoop is…
  - Hadoop Common
  - HDFS
  - YARN
  - MapReduce



HDFS + MapReduce

- “Apache Hadoop is a framework for running applications on large clusters of commodity hardware which implements the MapReduce computational paradigm and uses HDFS to store data among its nodes.”

Some history

- early ‘00s: Doug Cutting working on open-source web search engine called Nutch
  - “dang… parallel computing is hard.”
- 2003/4: Google publishes technical papers on their solution to this problem (MapReduce) and the filesystem that it runs on (Google Filesystem)
- 2004-6: Apache devs make open source version
- 2006: Doug Cutting hired to develop this project at Yahoo
- 2007: Doug Cutting open-sources Apache Hadoop

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.

Advantages

Advantages

- Distributed read/write capacity

Advantages

- Distributed read/write capacity
  - Non-distributed hard drive processing is slow. 75MB/sec processing speed – read 100TB = 16 days
  - Often, some data is more popular than other data. So some data are read more frequently. Thus most servers with data sit there unused. These resources can be more efficiently distributed.

Advantages

- Distributed read/write capacity
- But this distribution comes with a cost

Advantages

- Distributed read/write capacity
- But this distribution comes with a cost
  - With multiple machines, server failure and error rates increase dramatically: i.e. at least once/day
  - Hadoop expects these failures, and deals with them

Advantages

- Distributed read/write capacity
- 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

Advantages

- Distributed read/write capacity
- Deals with hardware failure
- Improves speed

Advantages

- Distributed read/write capacity
- Deals with hardware failure
- Improves 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

Advantages

- Distributed read/write capacity
- Deals with hardware failure
- Improves speed
- Fault tolerance

Advantages

- Distributed read/write capacity
- Deals with hardware failure
- Improves 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

Advantages

- Distributed read/write capacity
- Deals with hardware failure
- Improves speed
- Fault tolerance
- Cheaper

Advantages

- Distributed read/write capacity
- Deals with hardware failure
- Improves speed
- Fault tolerance
- Cheaper
  - computer A has power X
  - computer B has power 4*X
  - cost(B) >> 4*cost(A)

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)

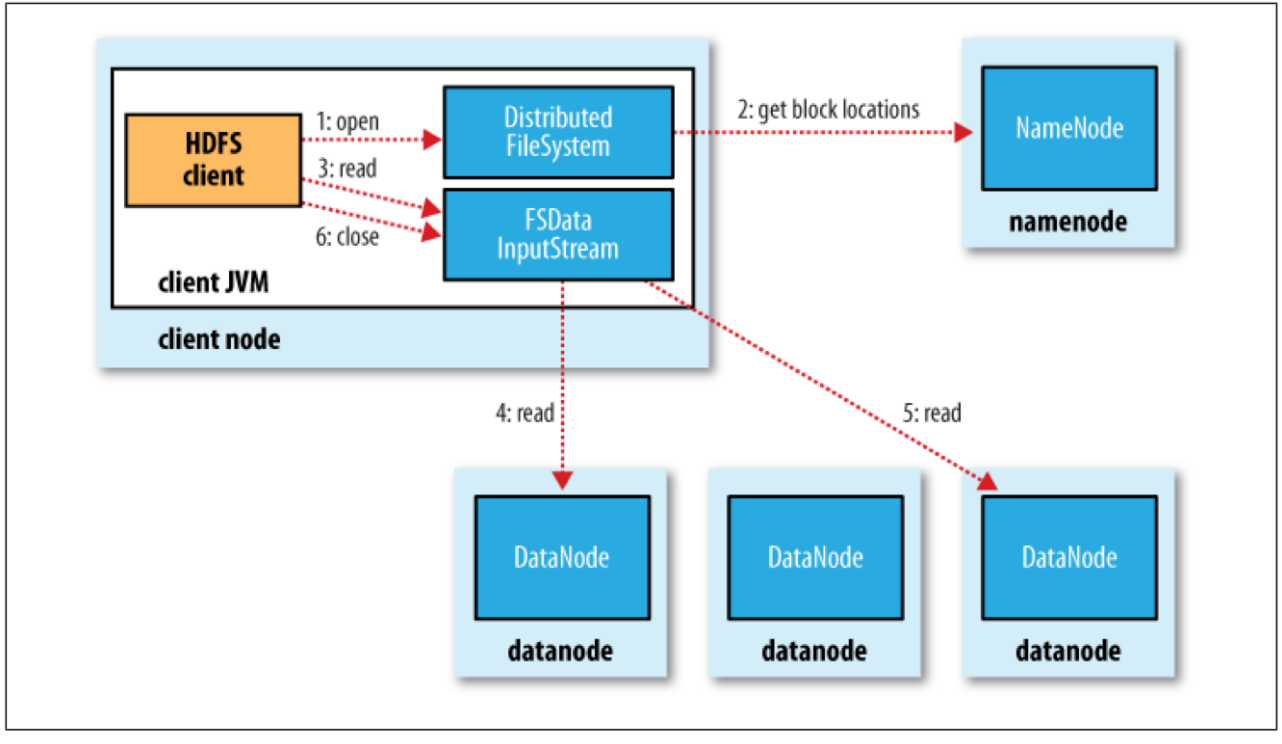


Figure 3-2. A client reading data from HDFS

MapReduce

MapReduce

- MapReduce = Map + Reduce functions

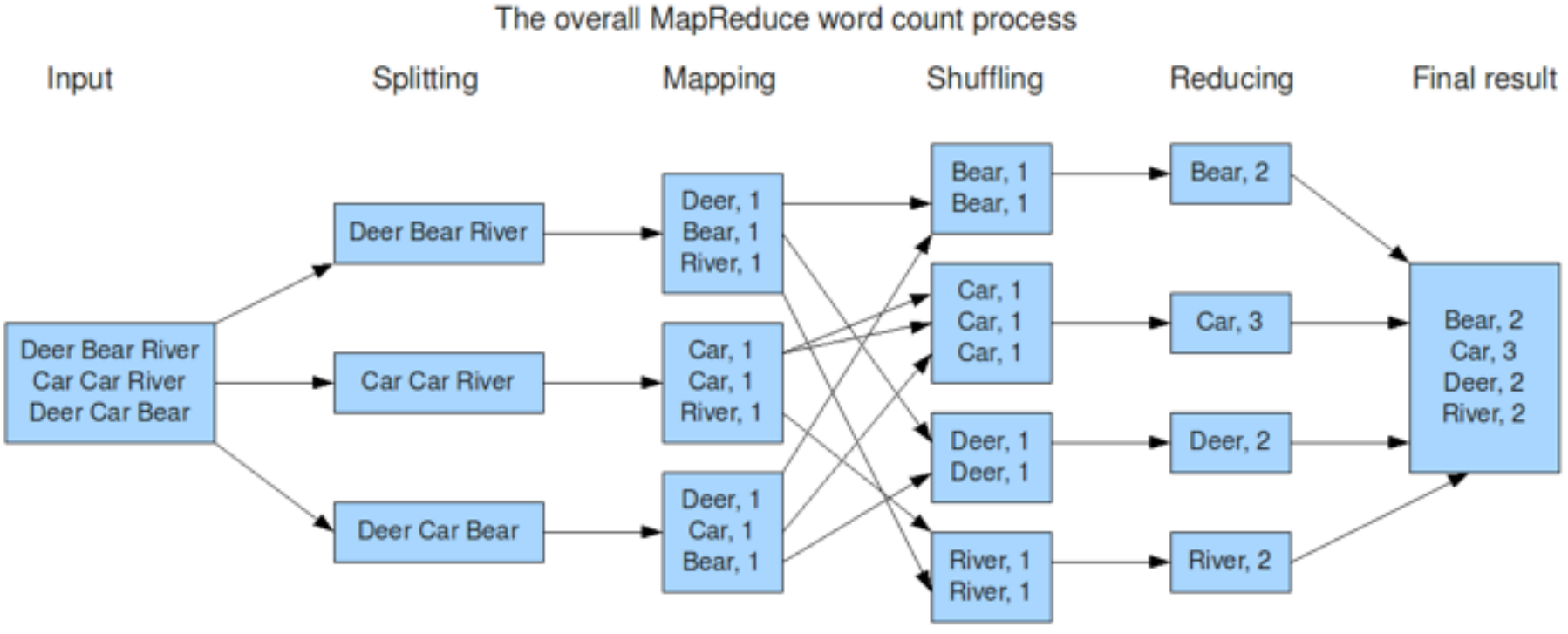
MapReduce

- MapReduce = Map + Reduce functions
  - MAP function
    - Input: raw data
    - Output: (key, value) pairs
  - REDUCE function
    - Input: all (key, value) pairs with a certain key
    - Output: summarized result for that key

MapReduce

- Compute word counts for a very large book.
  - MAP function
    - Input: raw text data
    - Output: For every word (“the”), return (“the”, 1) key-value pair
  - REDUCE function
    - Input: list of all (“the”, 1) pairs
    - Output: count of all “the”s in the book (“the”, 2343)

MapReduce


The overall MapReduce word count process
Input
Splitting
Mapping
Shuffling
Reducing
Final result
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, 1
Bear, 1
Car, 1
Car, 1
Car, 1
Deer, 1
Deer, 1
River, 1
River, 1
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2

MapReduce

- Mean precipitation over time, across cities
  - MAP function
    - Input: large table of precipitation over time, location data
    - Output: key, value pairs are like: (San Francisco, 2 cm)
  - REDUCE function
    - Input: list of all (city, precipitation) pairs for a given city
    - Output: average precipitation for that city
      - (San Francisco, 0.51 cm)

MapReduce

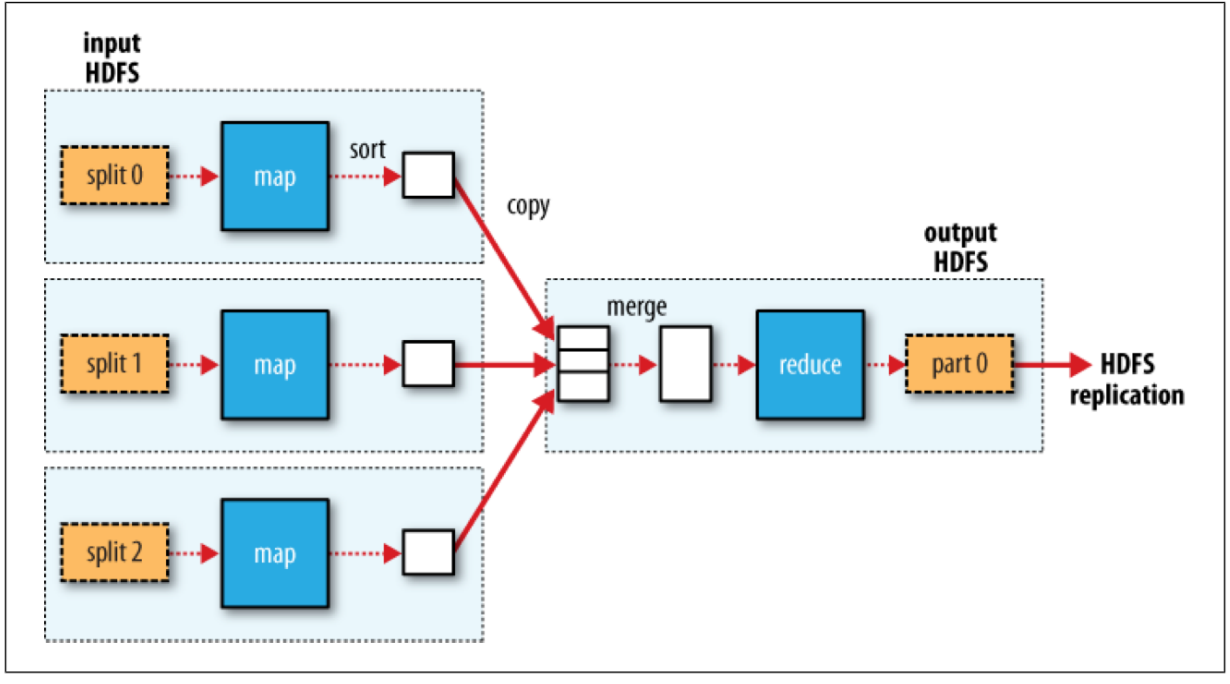


Figure 2-3. MapReduce data flow with a single reduce task

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.

MapReduce

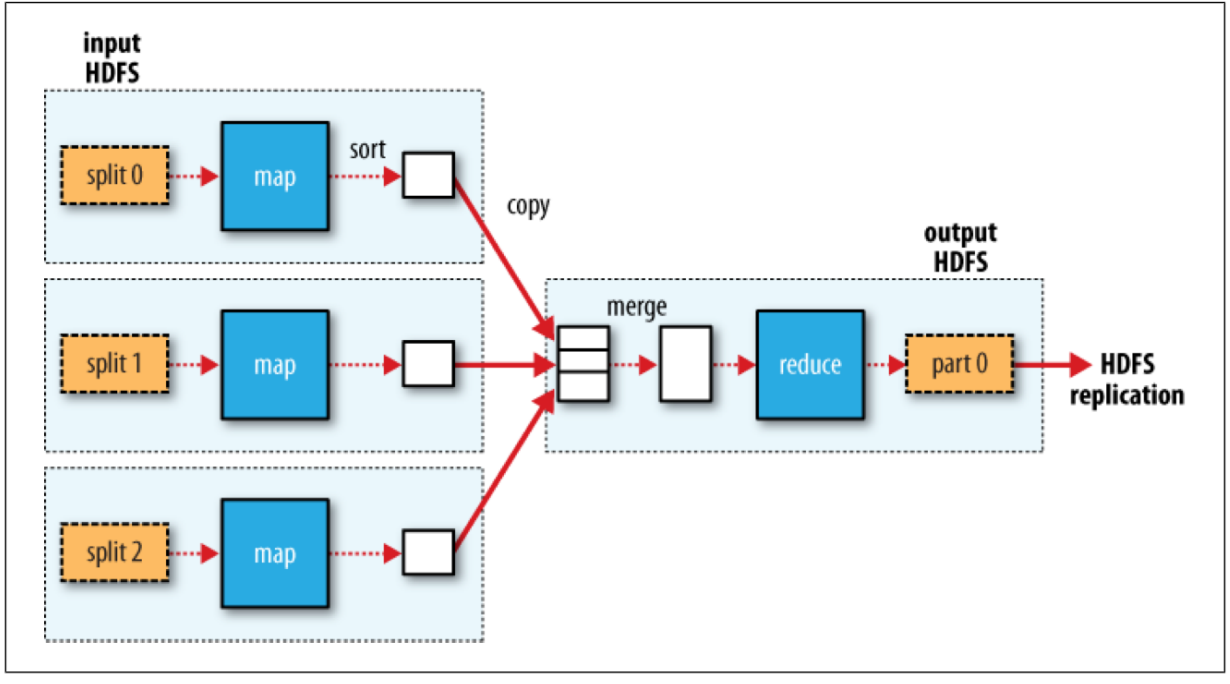


Figure 2-3. MapReduce data flow with a single reduce task

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!

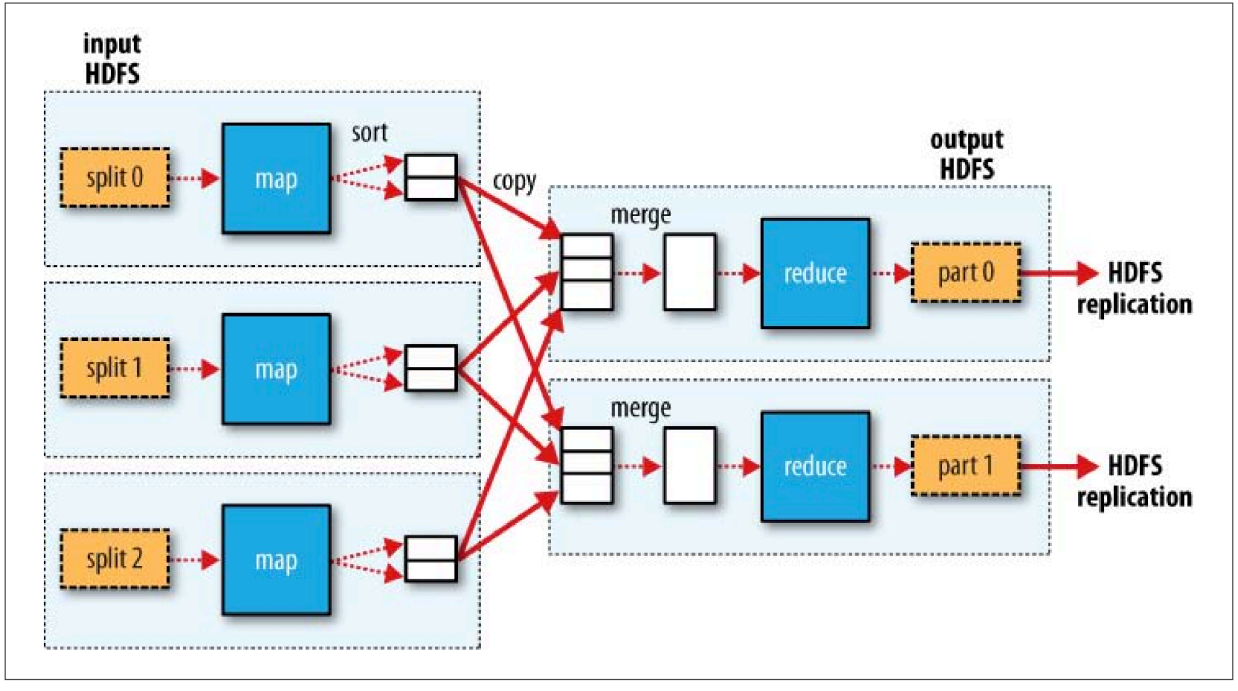
- input reader: gets data from file system
- **map function**: turns data into (key,value) pairs
- compare function: sort keys, collecting pairs with the same keys
- partition function: assign keys to reducer servers
- **reduce function**: reduce all values for key to one value or set
- output writer: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

MapReduce


input
HDFS
split 0
map
sort
split 1
map
split 2
map
copy
merge
reduce
part 0
output
HDFS
HDFS
replication
merge
reduce
part 1
HDFS
replication

Hadoop environment

- core:
  - Hadoop Common
  - HDFS
  - YARN
  - MapReduce

Hadoop environment

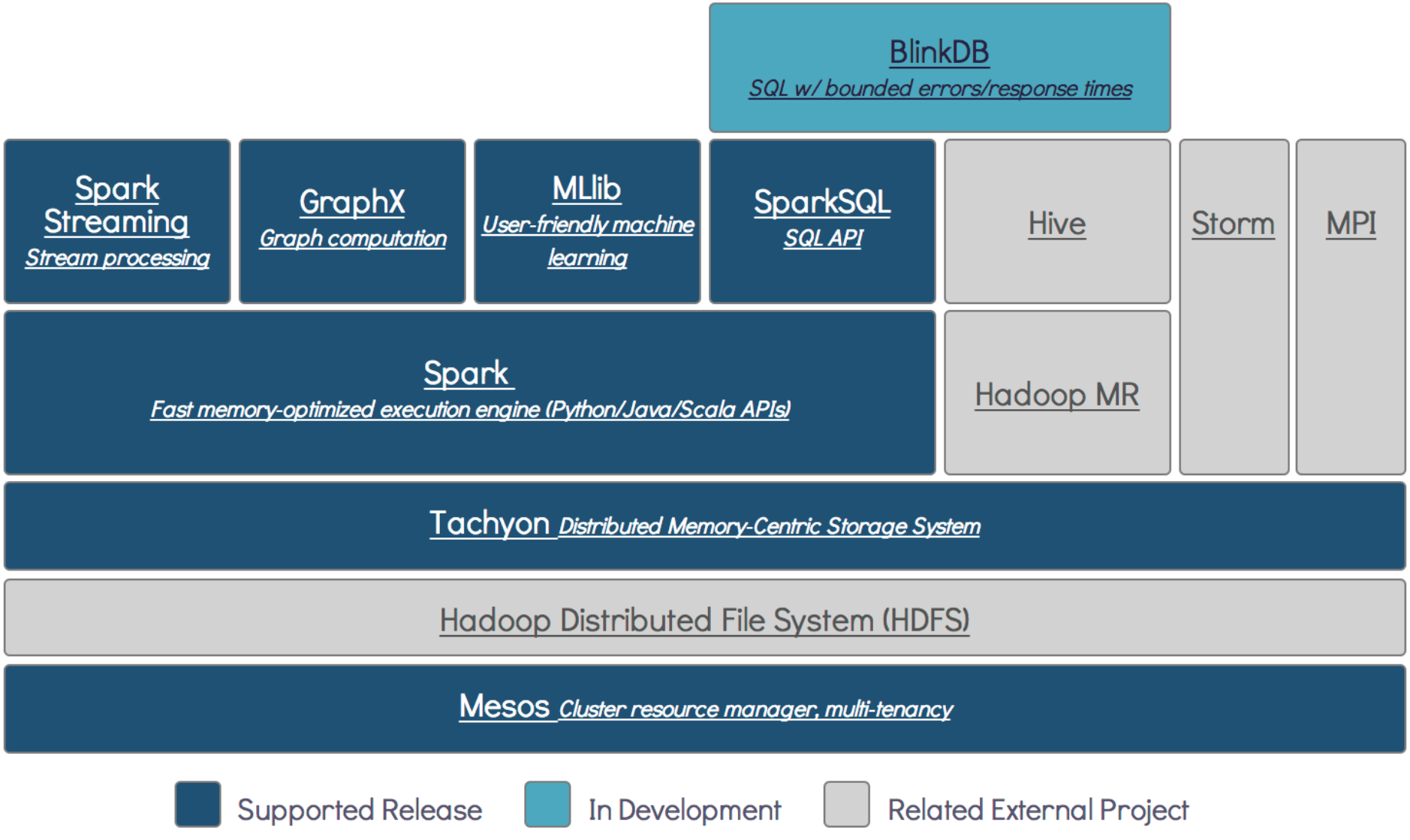
- core: Common, HDFS, YARN, MapReduce
- Hive: takes over management of HDFS storage, interprets & runs jobs via SQL queries
- Pig: write tasks more easily (for MapReduce+HDFS)
- HBase: take HDFS and add some good stuff from relational databases (make hadoop more SQL-like)
- Sqoop: import SQL →HDFS
- Avro: serialization
- Zookeeper: coordinator

Hadoop environment

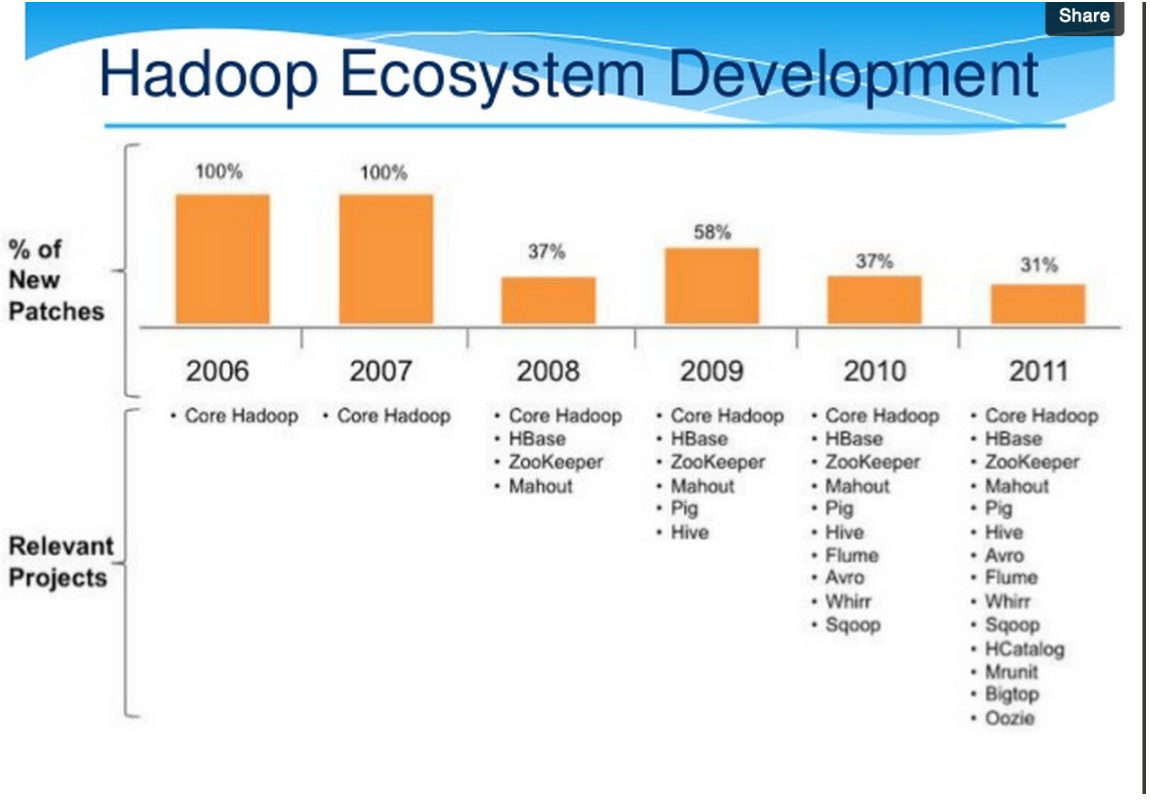
- Mahout – machine learning for Hadoop (err… Spark)
- Flume – stream logs into Hadoop
- Whirr – cloud platform libraries
- HCatalog – treat disparate data storage as one thing
- MRUnit – unit testing for Hadoop
- BigTop – attempt to re-define “core Hadoop”
- Oozie – workflow to integrate Hadoop + other tasks

Apache Spark

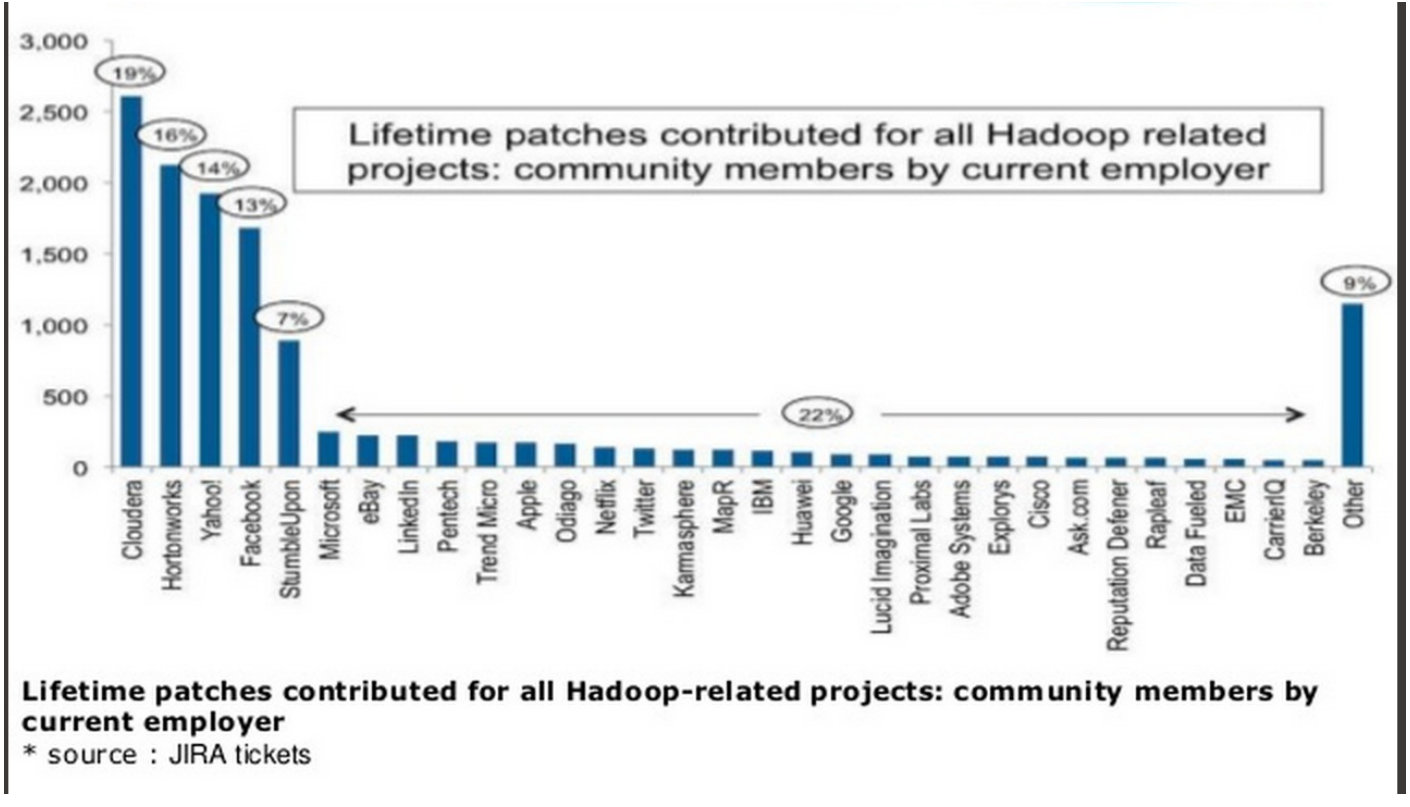
- Popularized in ~ 2013-2014
  - load all the data into memory
  - up to ~100 times faster than MapReduce
  - iterative algorithms become reasonable!
- Popular configuration:
  - HDFS / S3
  - YARN / Mesos
  - ~~MapReduce~~ Spark


BlinkDB
SQL w/ bounded errors/response times
Spark Streaming
Stream processing
GraphX
Graph computation
MLlib
User-friendly machine learning
SparkSQL
SQL API
Hive
Storm
MPI
Spark
Fast memory-optimized execution engine (Python/Java/Scala APIs)
Hadoop MR
Tachyon
Distributed Memory-Centric Storage System
Hadoop Distributed File System (HDFS)
Mesos
Cluster resource manager, multi-tenancy
Supported Release
In Development
Related External Project

Hadoop evolution



who built Hadoop?



Lifetime patches contributed for all Hadoop-related projects: community members by current employer

* source : JIRA tickets