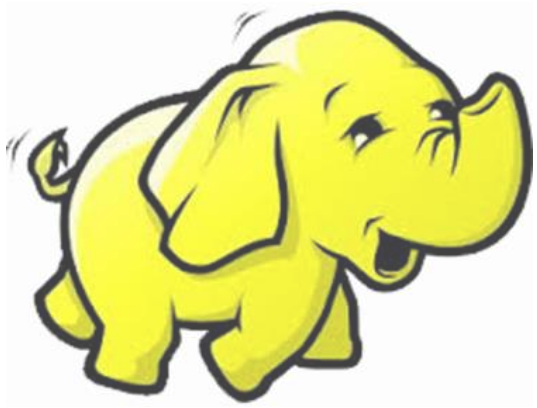
DISTRIBUTED OPERATING SYSTEMS-UNIT 4

(Meet Hadoop)

Dr.K.Geetha, Associate Professor, Dept. of Computer Science,Periyar Arts College, Cuddalore

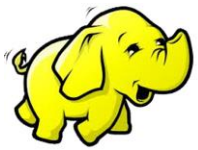


MEET HADOOP

Overview

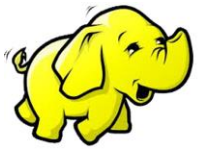
Meet Hadoop: Data - Data Storage and Analysis - Comparison with Other Systems - A Brief History of Hadoop - The Apache Hadoop Project – Map Reduce: A Weather Dataset - Analyzing the Data with UNIX Tools - Analyzing the Data with Hadoop - Scaling Out - Hadoop Streaming - Hadoop Pipes

- REFERENCE : Tom White, “Hadoop: The Definitive Guide”, Published by O’Reilly Media, Third Edition,2009



Data Storage and Analysis

- The Storage capacities of hard drives have increased massively
- Access speeds—the rate at which data can be read from drives have not kept up.
- The obvious way to reduce the time is to read from multiple disks at once.
- Imagine if we had 100 drives, each holding one hundredth of the data. Working in parallel, we could read the data in under two minutes.

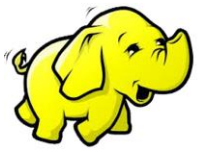


PROBLEMS

- Prolem #1
- Hardware failure: Many hardwares leads to failure and data loss

  Solution:

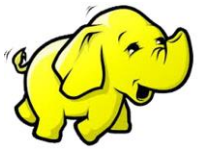
  A common way of avoiding data loss is through replication: redundant copies of the data are kept by the system so that in the event of failure, there is another copy available.
- Prolbem #2:
- Data read from one disk may need to be combined with the data from any of the other disks.
- Solution:
- MapReduce provides a programming model that abstracts the problem from disk reads and writes,



HADOOP

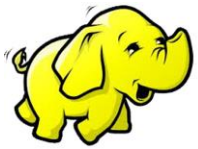
HADOOP PROVIDES

- a reliable shared storage and analysis system.
- The storage is provided by HDFS, and analysis by MapReduce
- MapReduce is a *batch* query processor,
- . It changes the way you think about data, and unlocks data that was previously archived on tape or disk.
- It gives people the opportunity to innovate with
- data..



COMPARISON

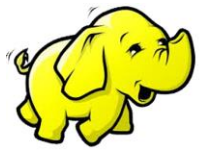
SL NO	RDBMS	MAP REDUCE
1	Data size Gigabytes	PetabyteS
2	Access Interactive	Batch
3	Read write Many times	Read once write many times
4	Structure Static schema	Dynamic Schema
5	Integrity High	Low
6	Scaling Non-Linear	Linear



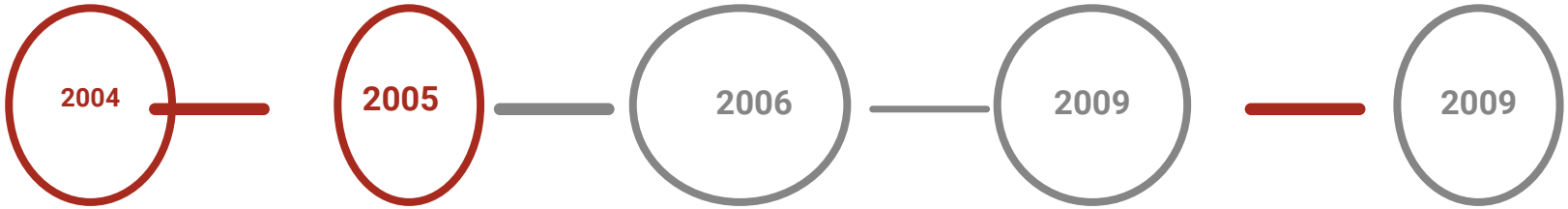
HADOOP

It is an **open source** software platform for distributed storage and distributed processing for extremely large data

1. Apache's project,
2. open-source implementation of frameworks
3. Reliable, scalable
4. Distributed computing and data storage



HISTORY



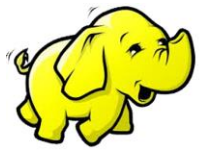
2004 — **intital version** — **Doug Cutting and Mike Cafarella**

2005 — **Hadoop runs on 20 nodes** — **Nutch**

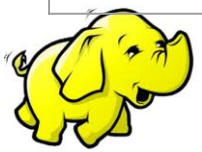
2006 — **Apache Hadoop Project**

2009 — **17 CLUSTERS** — **yahoo**

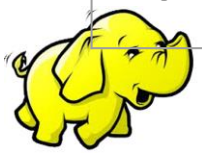
2009 — **SORT 500 GB** — **IN 2 SECONDS**



YEAR	EVENT
2002	Doug Cutting and Mike Caferella started to work for Nutch Project
2003	Google released paper on GFS
2004	Google releases another paper on Map Reduce
2005	Doug Cutting started using GFS and Hadoop in Nutch
2006	Doug Cutting joins Yahoo
2007	Doug Cutting created Hadoop

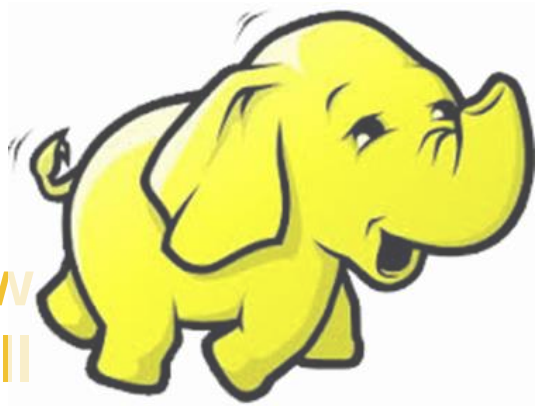


January 2008	Yahoo tests Hadoop in 1000 clusters
July 2008	Yahoo released Hadoop as open source project to ASF-Apache software foundation
2009	Hadoop was successfully tested for petabytes in less than 17 hours
2011	Hadoop version 1.0
2017	Version 3.0



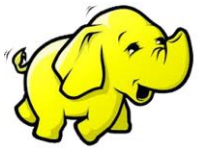
The origin of name Hadoop

The name Hadoop is not an acronym; it's a made-up name. The project's creator, Doug Cutting, explains how the name came about:



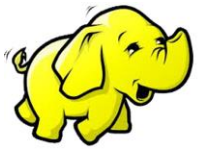
The name my kid gave a stuffed yellow elephant. Short, relatively easy to spell and pronounce, meaningless, and not used elsewhere: those are my naming criteria.

Kids are good at generating such. Googol is a kid's term.



HADOOP

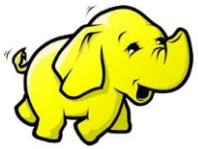
•Distributed, with some centralization
•Main nodes of cluster are where most of the computational power and storage of the system lies
•Main nodes run TaskTracker to accept and reply to MapReduce tasks, and also DataNode to store needed blocks closely as possible
•Central control node runs NameNode to keep track of HDFS directories & files, and JobTracker to dispatch compute tasks to TaskTracker
••Written in Java, also supports Python and Ruby



HADOOP

NameNode:

- Stores metadata for the files, like the directory structure of a typical FS.
- The server holding the NameNode instance is quite crucial, as there is only one.
- Transaction log for file deletes/adds, etc. Does not use transactions for whole blocks or file-streams, only metadata.
- Handles creation of more replica blocks when necessary after a DataNode failure

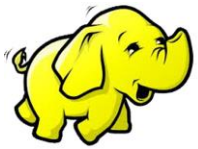


HADOOP

DataNode:

- Stores the actual data in HDFS
- Can run on any underlying filesystem

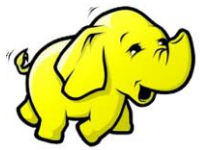
Notifies NameNode of what blocks it has



HADOOP

MapReduce Engine:

- JobTracker & TaskTracker
- JobTracker splits up data into smaller tasks("Map") and sends it to the TaskTracker process in each node
- TaskTracker reports back to the JobTracker node and reports on job progress, sends data ("Reduce") or requests new jobs



HADOOP

Core

A set of components and interfaces for distributed filesystems and general I/O

(serialization, Java RPC, persistent data structures).

Avro

A data serialization system for efficient, cross-language RPC, and persistent data

storage. (At the time of this writing, Avro had been created only as a new subproject,

and no other Hadoop subprojects were using it yet.)

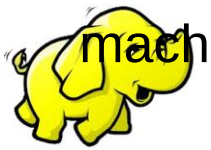
MapReduce

A distributed data processing model and execution environment that runs on large

clusters of commodity machines.

HDFS

A distributed filesystem that runs on large clusters of commodity machines.



HADOOP

Pig

A data flow language and execution environment for exploring very large datasets Pig runs on HDFS and MapReduce clusters.

HBase

A distributed, column-oriented database. HBase uses HDFS for its underlyingstorage, and supports both batch-style computations using MapReduce and point

queries (random reads).

ZooKeeper

A distributed, highly available coordination service. ZooKeeper provides primitivessuch as distributed locks that can be used for building distributed applications.

Hive

A distributed data warehouse. Hive manages data stored in HDFS and provides a query language based on SQL (and which is translated by the runtime engine to

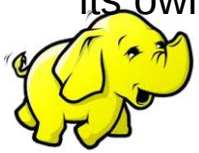
MapReduce jobs) for querying the data.

Chukwa

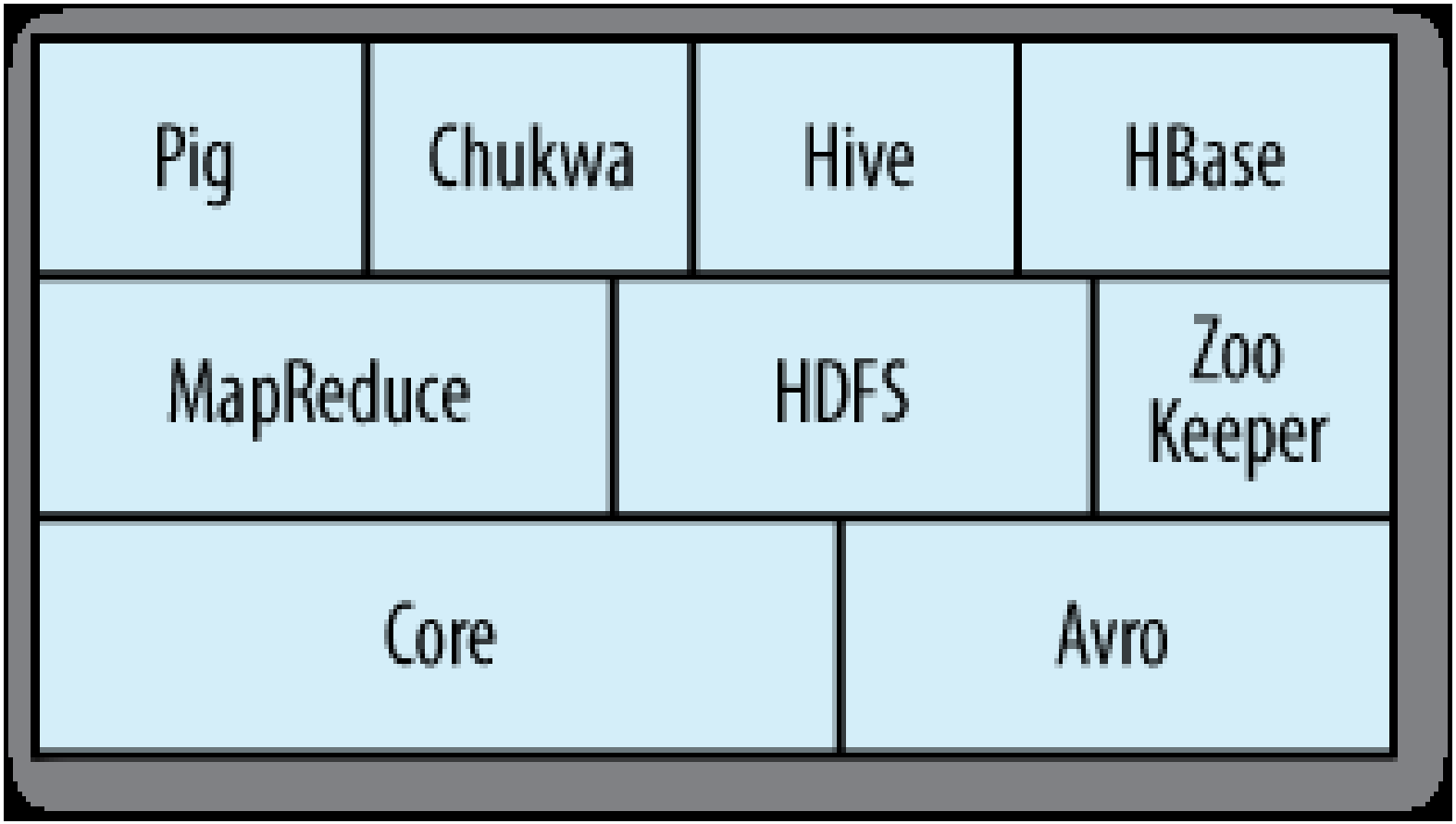
A distributed data collection and analysis system. Chukwa runs collectors that store data in HDFS, and it uses MapReduce to produce reports. (At the time of this

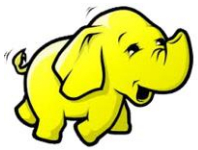
writing, Chukwa had only recently graduated from a “contrib” module in Core to

its own subproject.)



APACHE HADOOP





WHETHER DATA SET

1901 317

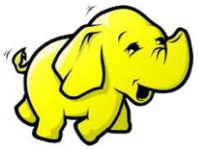
1902 244

1903 289

1904 256

1905 283

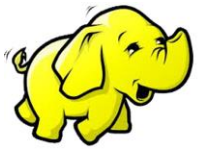
COLLECTING AND ANALYSING



WEATHER DATA SET

The temperature values in the source file are scaled by a factor of 10, so this works out as a maximum temperature of 31.7°C for 1901 (there were very few readings at the beginning of the century, so this is plausible. The complete run for the century took

42 minutes in one run on a single EC2 High-CPU Extra Large Instance



WEATHER DATA SET

To speed up the processing, we need to run parts of the program in parallel.

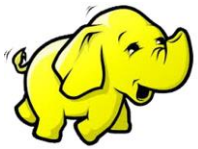
1.Dividing the work into equal-size pieces

so some processes will finish much earlier

than others.

Even if they pick up further work, the whole run is dominated by the longest file.

An alternative approach is to split the input into fixed-size chunks and assign each chunk to a process.



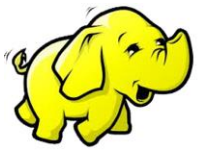
WEATHER DATA SET

2.Combining the results from independent processes can need further processing.

The result for each year is independent of other years and may be combined by concatenating all the results, and sorting by year. If using the fixed-size chunk approach, the combination is more delicate.

Data for a particular year will typically be split into several chunks, each processed independently.

We will find the maximum temperature for each chunk, so the final step is to look for the highest of these maximums, for each year.



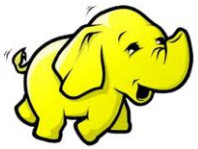
WEATHER DATA SET

3.We are limited by the processing capacity of a single machine., some datasets grow beyond the capacity of a single machine.

When we start using multiple machines, a whole host of other factors come into play, mainly falling in the category of

coordination and

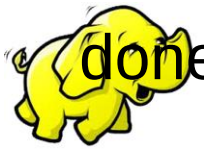
reliability.



Analyzing the Data with Unix Tools

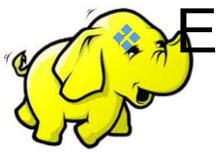
Program to find maximum temp.

```
#!/usr/bin/env bash
for year in all/*
do
echo -ne `basename $year .gz`"\t"
gunzip -c $year | \
awk '{ temp = substr($0, 88, 5) + 0;
q = substr($0, 93, 1);
if (temp !=9999 && q ~ /[01459]/ && temp > max) max =
temp }
END { print max }'
done
```



Analyzing the Data with Unix Tools

- The program first prints the year, and then processes each file using awk. The awk script extracts two fields from the data: the air temperature and the quality code.
- The air temperature value is turned into an integer by adding 0.
- Next, a test is applied to see if the temperature is valid (the value 9999 signifies a missing value in the NCDC dataset),
- If the quality code indicates that the reading is not suspect or erroneous. If the reading is OK, the value is compared with the maximum value seen so far, which is updated if a new maximum is found. The END block is executed and prints the maximum value.



HADOOP-How it works

MapReduce works by breaking the processing into two phases:

the map phase and the

reduce phase.

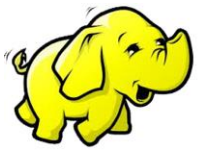
Each phase has key-value pairs as input and output, the types of which

may be chosen by the programmer.

specifies two functions:

the map function and the reduce function.

The input to our map phase is the raw data.



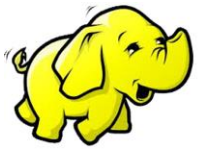
HADOOP

Our map function is simple. We pull out the year and the air temperature,

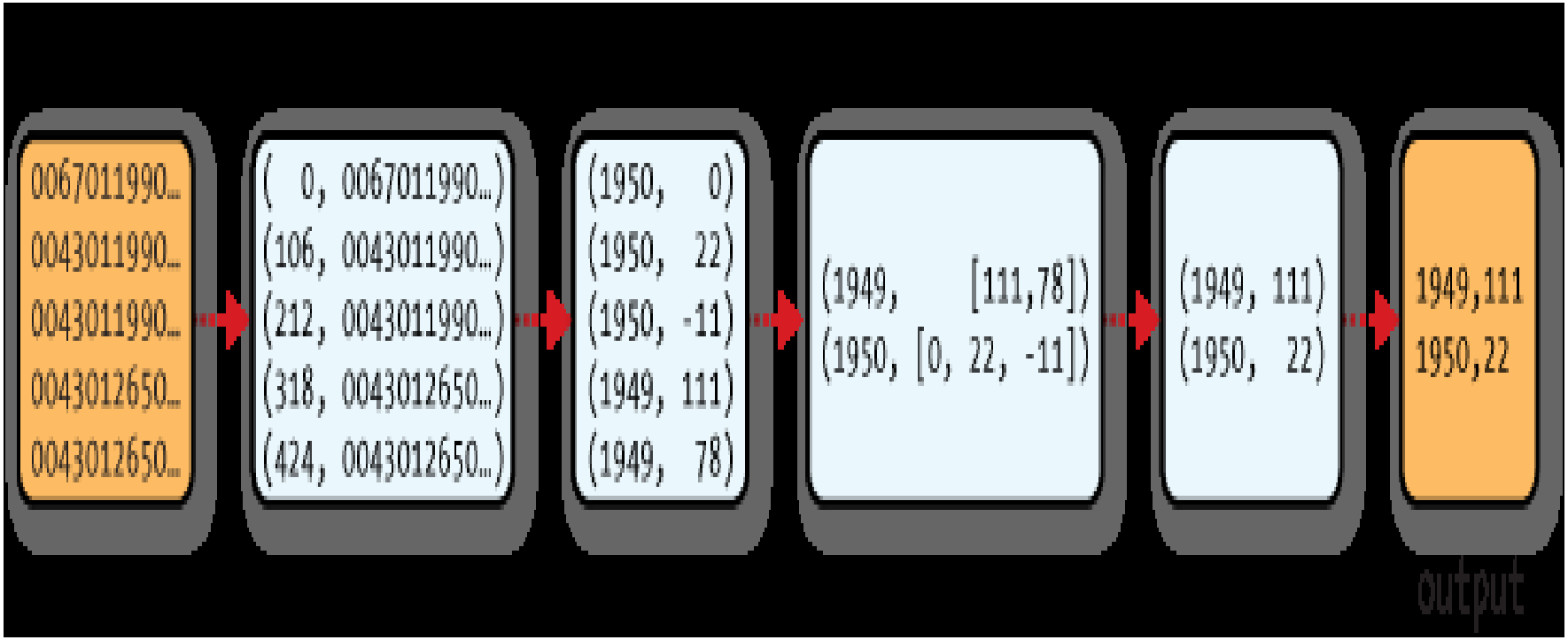
The map function is just a data preparation phase, setting up the data in such a way that the reducer function can do its work on it: finding the maximum temperature for each year.

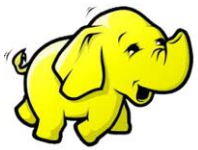
The map function is also a good place to drop bad records: we filter out temperatures that are missing,

suspect, or erroneous.

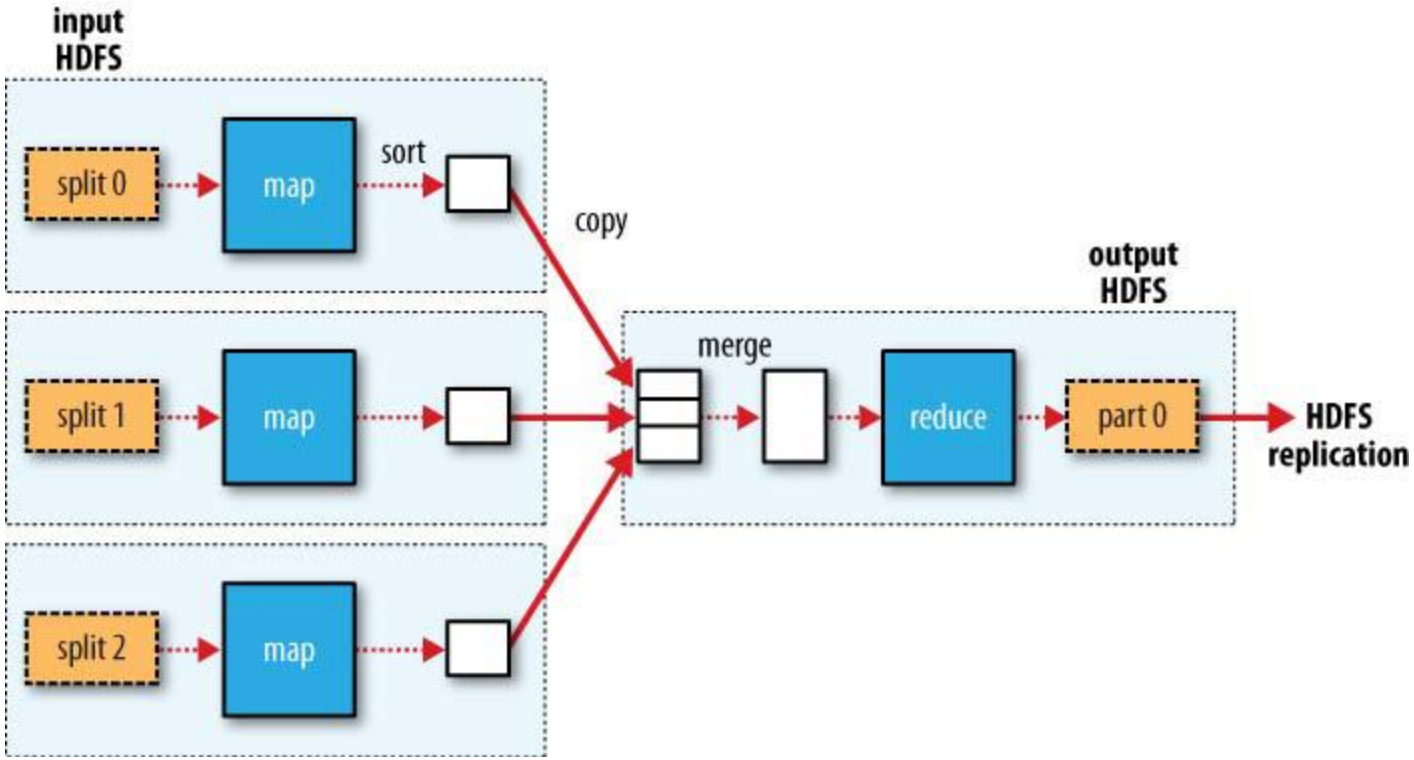


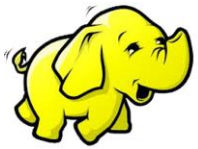
HADOOP


0067011990…
0043011990…
0043011990…
0043012650…
0043012650…
(0, 0067011990…)
(106, 0043011990…)
(212, 0043011990…)
(318, 0043012650…)
(424, 0043012650…)
(1950, 0)
(1950, 22)
(1950, -11)
(1949, 111)
(1949, 78)
(1949, [111,78])
(1950, [0, 22, -11])
(1949, 111)
(1950, 22)
1949,111
1950,22
output

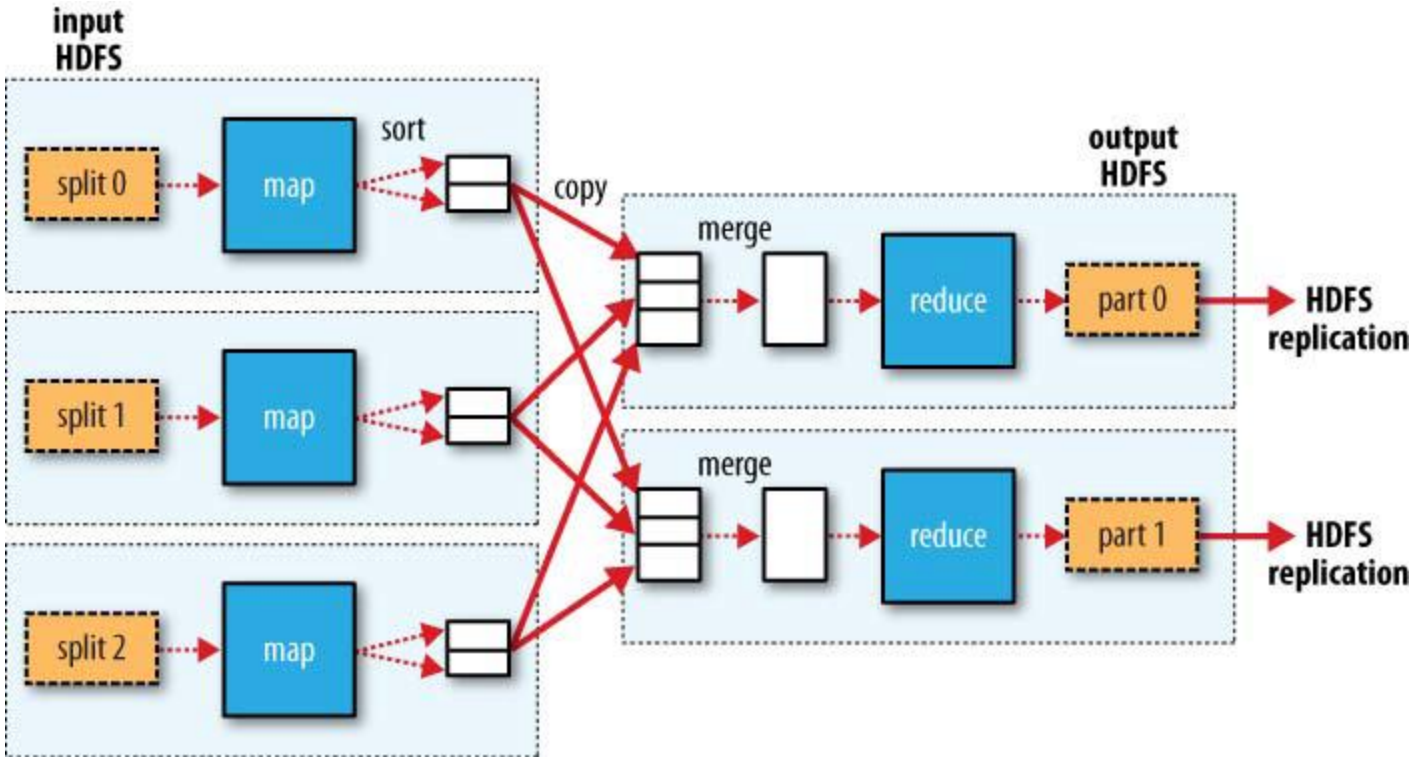


SINGLE REDUCE

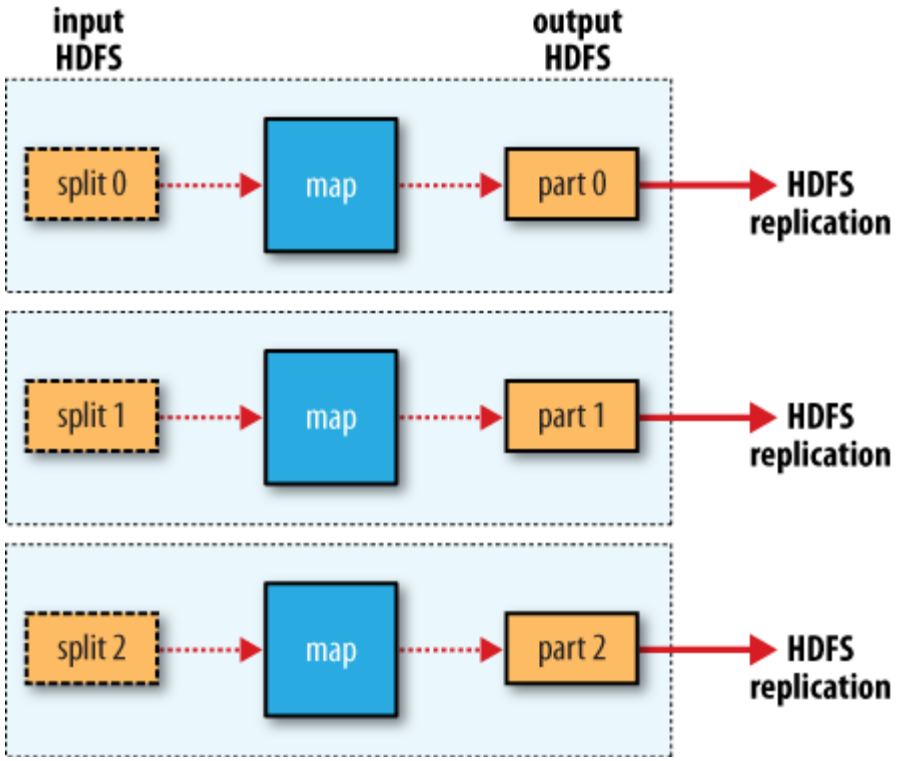


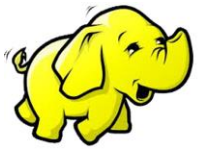


MULTIPLE REDUCE



NO REDUCE


input
HDFS
output
HDFS
split 0
map
part 0
HDFS
replication
split 1
map
part 1
HDFS
replication
split 2
map
part 2
HDFS
replication



Reference: Tom White, “Hadoop: The Definitive Guide”, Published by O’Reilly Media, Third Edition,2009