MONASH University

MONASH
INFORMATION
TECHNOLOGY

Introduction to Apache Hadoop

Borhan Kazimipour

TP3 2018
Week 2

GROUP OF EIGHT AUSTRALIA

Module Intro

1. Introduction to Big Data
   a. intro to main concepts/definitions.
   b. the ecosystem of processing frameworks & data sources
   c. intro to popular prog. lang. Scala

2. Hadoop Ecosystem and Applications
   a. introduction to Apache Hadoop framework/architecture
   b. Hadoop distributed File System (HDFS)
   c. MapReduce prog. Model

3. NoSQL 1: Big Data Processing Concepts and Technologies
   a. theory of distributed transactions & concurrency control.
   b. columnar type databases.
   c. Apache HBase architecture/operations

4. NoSQL 2: Spark & distributed graph processing
5. Data streams
6. Advanced Topics in Big Data

What is the problem?

- To do Big Data we need:
  - rapid processors,
  - fast access to data (e.g., disk speed and CPU high bandwidth)
- In practice, there are some natural barriers:
  - upper bounds on the CPU speeds
  - tight budget
- What is the solution?
  - Parallelism!
  - If we can distribute the process and data to separate devices, we can enhance the performance of data processing without necessarily using very fast CPU/s.

What is the problem?

- To do Big Data we need:
  - rapid processors,
  - fast access to data (e.g., disk speed and CPU high bandwidth)
- In practice, there are some natural barriers:
  - upper bounds on the CPU speeds
  - tight budget
- What is the solution?
  - Parallelism!
  - If we can distribute the process and data to separate devices, we can enhance the performance of data processing without necessarily using very fast CPU/s.

What is the parallelism?

- What is parallelism?
  - To simultaneously perform many computational operations
- Any challenge?
  - To identify the right parallel structure,
  - optimal number of CPU/s,
  - communications between disks and processors,
  - architecture design
  - etc

How to parallel?

- How?
  - Using distributed systems!
  - A distributed system is a system in which components located at **networked** computers communicate/coordinate their actions only by **passing messages**.
  - Regardless of the size of a distributed system, it appears to its users as a **single coherent system**.
- Types of distributed systems:
  - loosely coupled systems
    i.
  - tightly coupled systems.
    i.

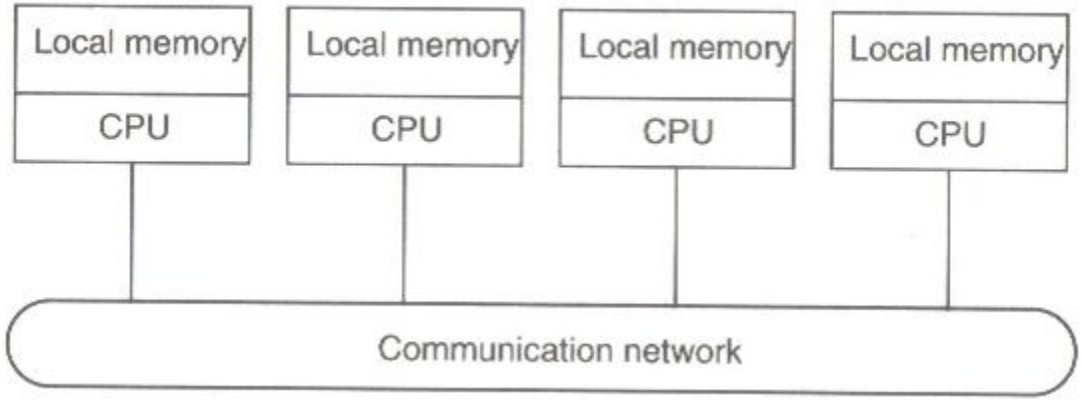
How to parallel?

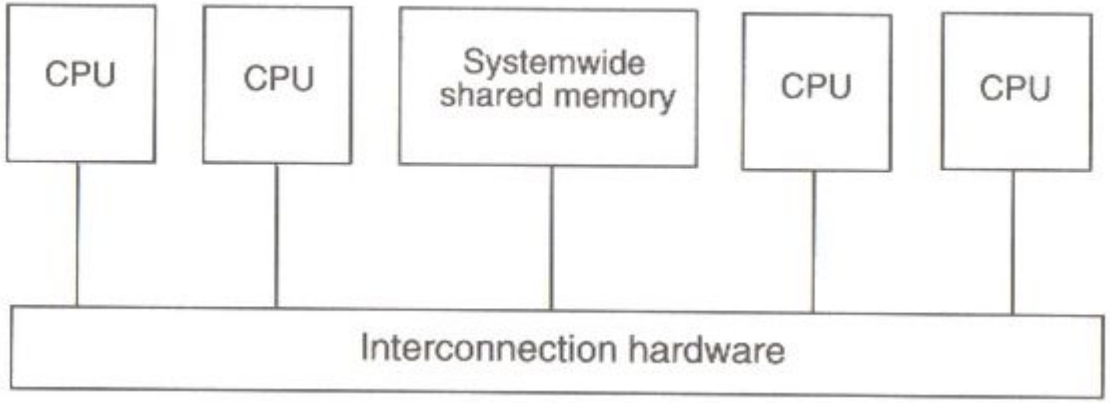
- How?
  - Using distributed systems!
  - A distributed system is a system in which components located at **networked** computers communicate/coordinate their actions only by **passing messages**.
  - Regardless of the size of a distributed system, it appears to its users as a **single coherent system**.
- Types of distributed systems:
  - loosely coupled systems
    i.
  - tightly coupled systems.
    i.

How to parallel?

- How?
  - Using distributed systems!
  - A distributed system is a system in which components located at **networked** computers communicate/coordinate their actions only by **passing messages**.
  - Regardless of the size of a distributed system, it appears to its users as a **single coherent system**.
- Types of distributed systems:
  - loosely coupled systems
    i. processors do **not** share their local memory with others
  - tightly coupled systems.
    i. all devices share a common primary memory

Which one is the looser?!





Hadoop


"Finance here - we're not sure about this Hadoop thing... Could you just dump it all into Excel for us?"
TimoElliott.com

Apache Hadoop

- Apache Hadoop is an open-source software **framework** for ***distributed storage*** and ***processing***.
  - Apache Nutch project (2003)
  - Apache Hadoop subproject (2006)
- distributed storage:
  - Hadoop Distributed File System
  - Google File System (2003)
- distributed processing:
  - MapReduce programming model
  - Google MapReduce (2004)

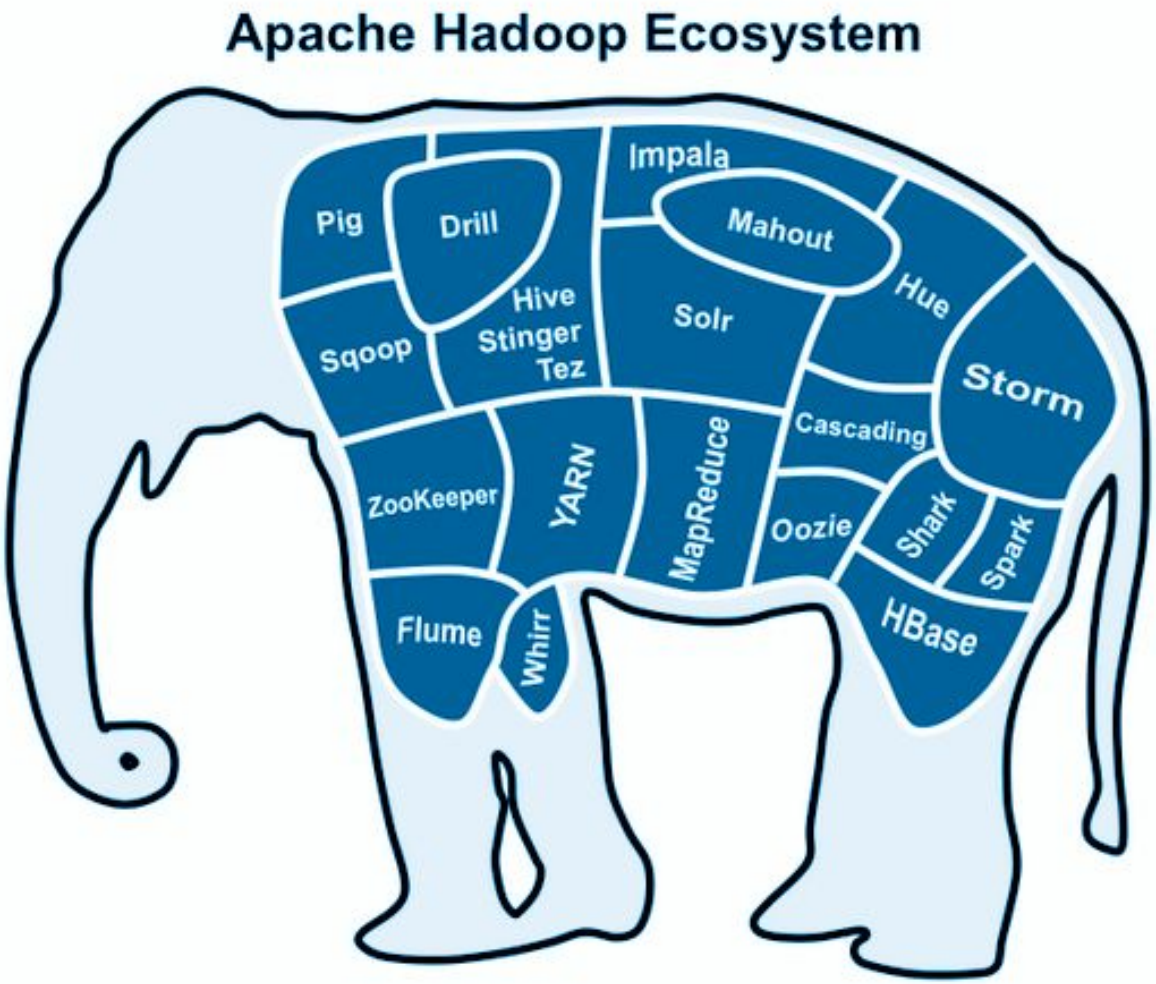
Apache Hadoop

- Apache Hadoop is an open-source software **framework** for ***distributed storage*** and ***processing***.
  - Apache Nutch project (2003)
  - Apache Hadoop subproject (2006)
- distributed storage:
  - Hadoop Distributed File System
  - Google File System (2003)
- distributed processing:
  - MapReduce programming model
  - Google MapReduce (2004)

Apache Hadoop

- Apache Hadoop is an open-source software **framework** for ***distributed storage*** and ***processing***.
  - Apache Nutch project (2003)
  - Apache Hadoop subproject (2006)
- distributed storage:
  - Hadoop Distributed File System
  - Google File System (2003)
- distributed processing:
  - MapReduce programming model
  - Google MapReduce (2004)

Hadoop-based Technologies



1. Hadoop-based Technologies: HBase (Hadoop Database)

- A non-relational, distributed, and column-oriented database
- runs on top of HDFS.
- **real-time** read/write random access to datasets
- billions rows and millions columns.
- inspired by the famous Google BigTable
- does **not** support SQL-like queries
- is covered in Module 3

2. Hadoop-based Technologies: Hive

- a query engine with an SQL-like interface HiveQL (HQL) for HDFS.
- a data warehouse infrastructure that provides data **summarization**, **ETL**, and **analysis.**
- Hive is **NOT** a relational database system.
- HQL queries are translated to MapReduce jobs to exploit the scalability of MapReduce.
- very optimized for scalability purposes (summarization).
- is **NOT** designed to minimize the latency.

3. Hadoop-based Technologies: Pig

- a platform for **analyzing** and **querying** huge datasets (similar to Hive)
- does **NOT** support SQL-like queries
- has its own scripting language: **Pig Latin**
- Pig Latin contains many built-in data manipulation operations (grouping, joining, and filtering)

4. Hadoop-based Technologies: Mahout

- a scalable machine learning and data mining library:
  - **Recommendation Sys**: collaborative filtering
  - **Classifiers**: Logistic Regression, Naive Bayes, HMM
  - **Clusterings**: Canopy, K-Means, and Spectral Clustering
  - **Dim Reduction:** SVD and PCA

5. Hadoop-based Technologies: Spark

- a scalable **compute** engine
- uses its own data processing framework instead of MapReduce
- can perform **in-memory computing** (10 to 100 times faster)
- can be adopted independently of Hadoop
- is usually used with Hadoop as an alternative to MapReduce
- *Spark Core: Module 4*
- *Spark GraphX: Module 4*
- *Spark Streaming: Module 5*
- *Spark Machine Learning: Module 6*

Hadoop-based Technologies: Summary

1. Which tool is better for creating a large-scale messenger? Why?
   - Hive?
   - HBase?

Hadoop-based Technologies: Summary

1. Which tool is better for creating a large-scale messenger? Why?
   - Hive?
   - HBase?

2. Which one supports a SQL-like language?
   - Hive?
   - Hbase?
   - Pig?

Hadoop-based Technologies: Summary

1. Which tool is better for creating a large-scale messenger? Why?
   - Hive?
   - HBase?

2. Which one supports a SQL-like language?
   - Hive?
   - Hbase?
   - Pig?

3. Why Spark is faster than Hadoop?

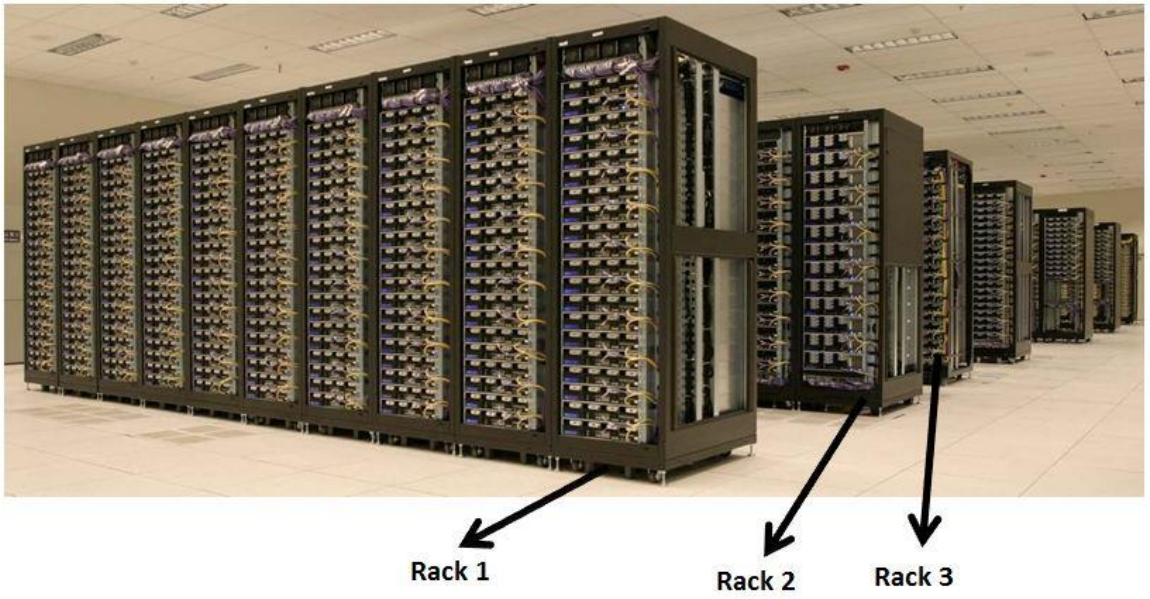
Hadoop Architecture

Hadoop Jargons

- Cluster:
  -
- Rack:
  -
- Node:
  -

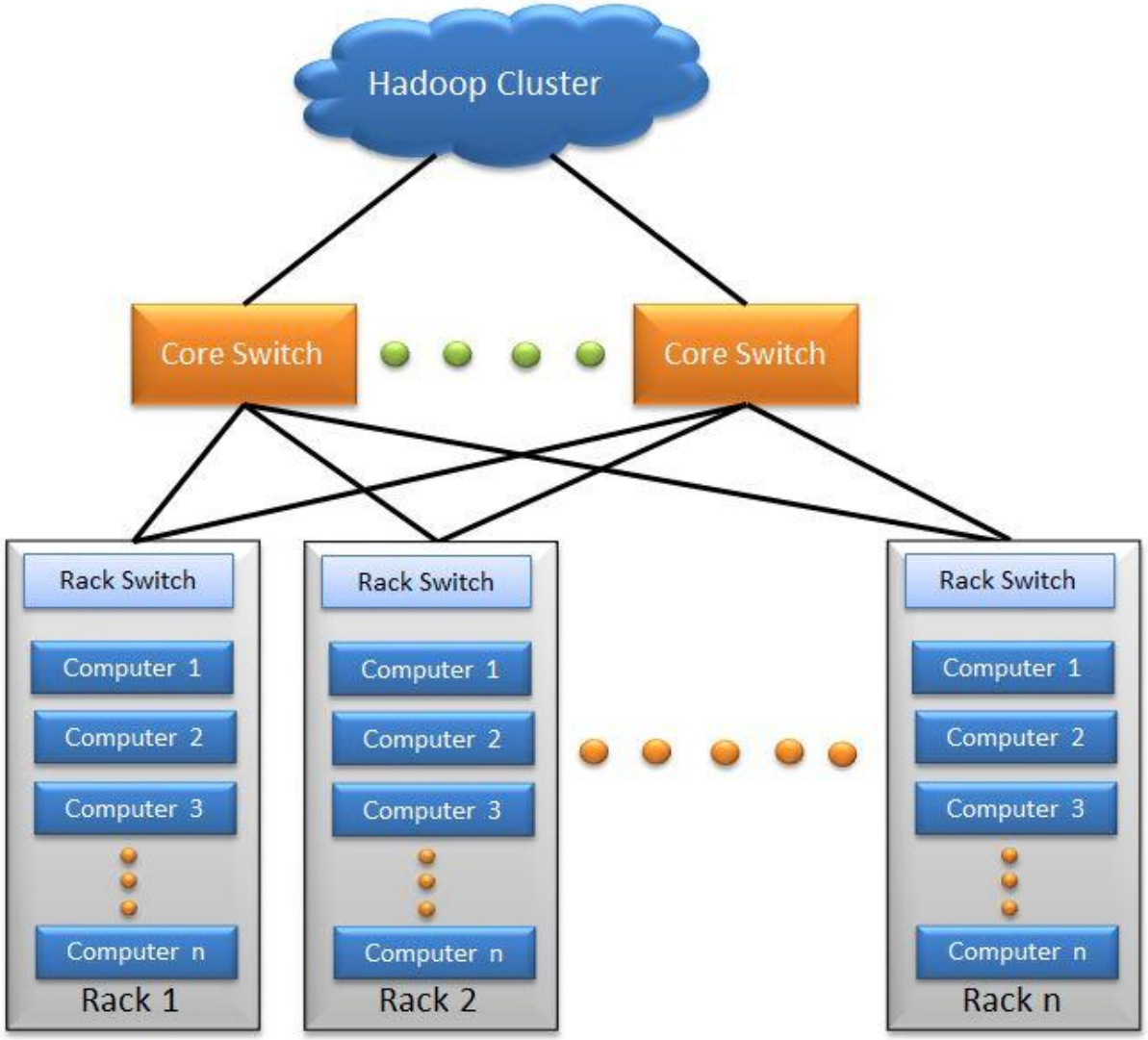
Hadoop Jargons

- Cluster:
  - A set of host machines
- Rack:
  - A subset of a cluster
- Node:
  - A machine!

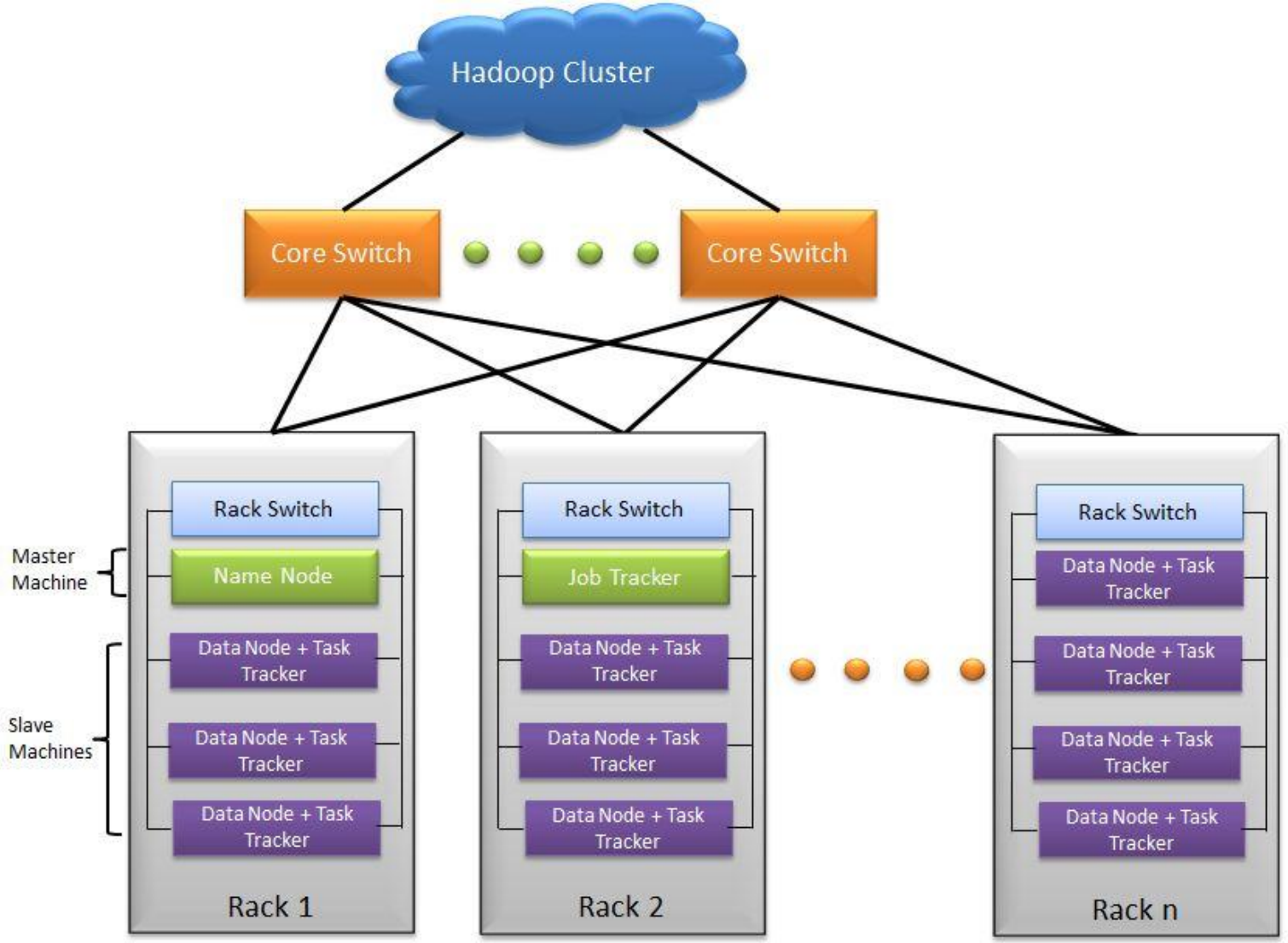


A Picture of Yahoo's Hadoop Cluster

Hadoop Cluster



Hadoop Cluster



Hadoop Cluster Components

- Client
  - loading the data into cluster (HDFS)
  - submit MapReduce jobs (MapReduce)
  - retrieve the data after job completion (HDFS)
- Master:
  - NameNode (HDFS)
  - Secondary NameNode (HDFS)
  - JobTracker (MapReduce)
- Slave
  - Store the data (HDFS)
  - Process the computation (MapReduce)

Hadoop Cluster Components

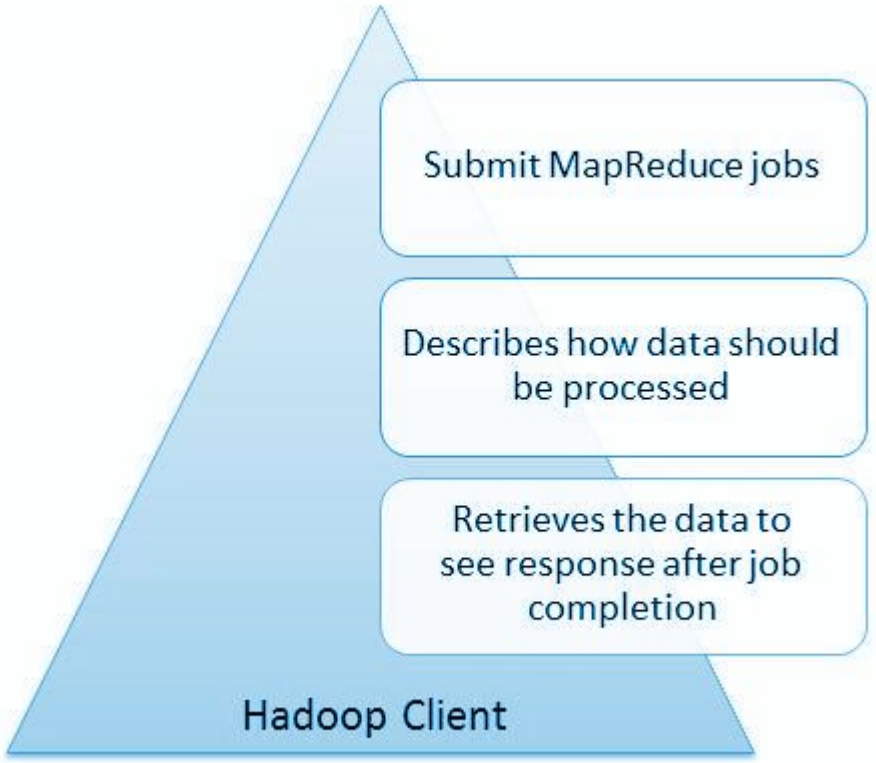
- Client
  - loading the data into cluster (HDFS)
  - submit MapReduce jobs (MapReduce)
  - retrieve the data after job completion (HDFS)
- Master:
  - NameNode (HDFS)
  - Secondary NameNode (HDFS)
  - JobTracker (MapReduce)
- Slave
  - Store the data (HDFS)
  - Process the computation (MapReduce)

Hadoop Cluster Components

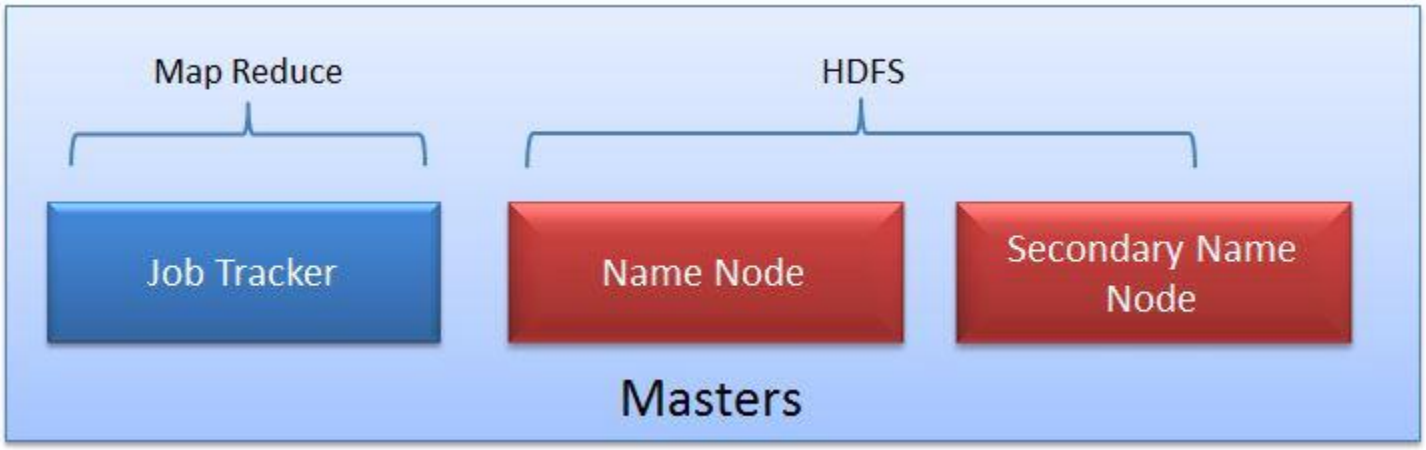
- Client
  - loading the data into cluster (HDFS)
  - submit MapReduce jobs (MapReduce)
  - retrieve the data after job completion (HDFS)
- Master:
  - NameNode (HDFS)
  - Secondary NameNode (HDFS)
  - JobTracker (MapReduce)
- Slave
  - Store the data (HDFS)
  - Process the computation (MapReduce)

Client Node

- Does **not**:
  - Store the data
  - Process the data

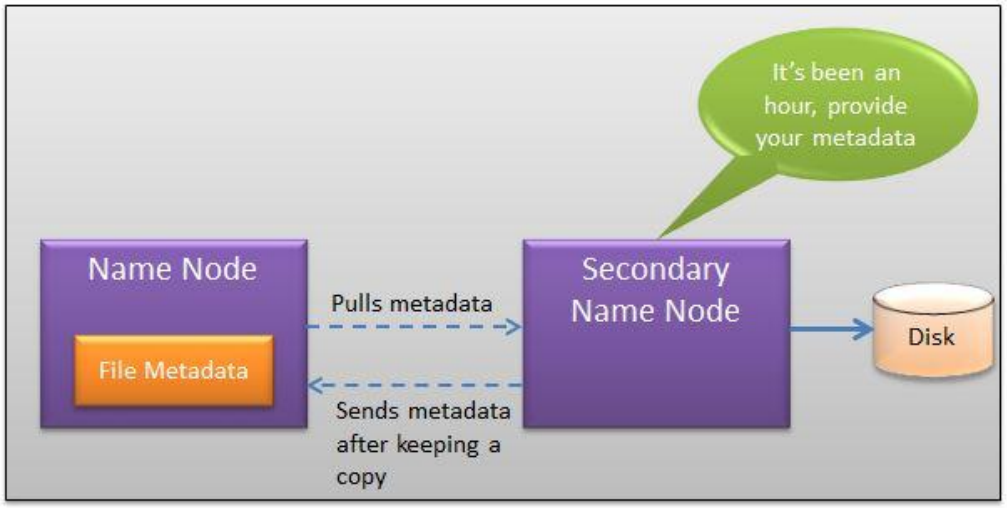


Master: NameNode (HDFS)



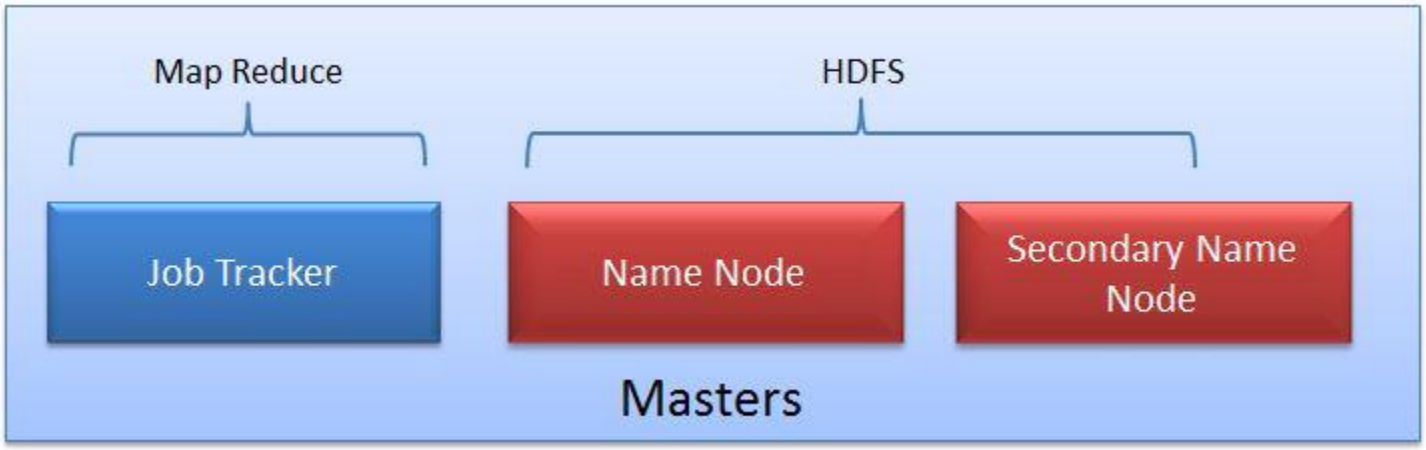
- store metadata (Not the real data).
- monitor the health of DataNode
- keeps track of all the file system related information
  - Which section of file is saved in which part of the cluster?
  - Last access time for the files?
  - User permissions?

Master: Secondary NameNode (HDFS)



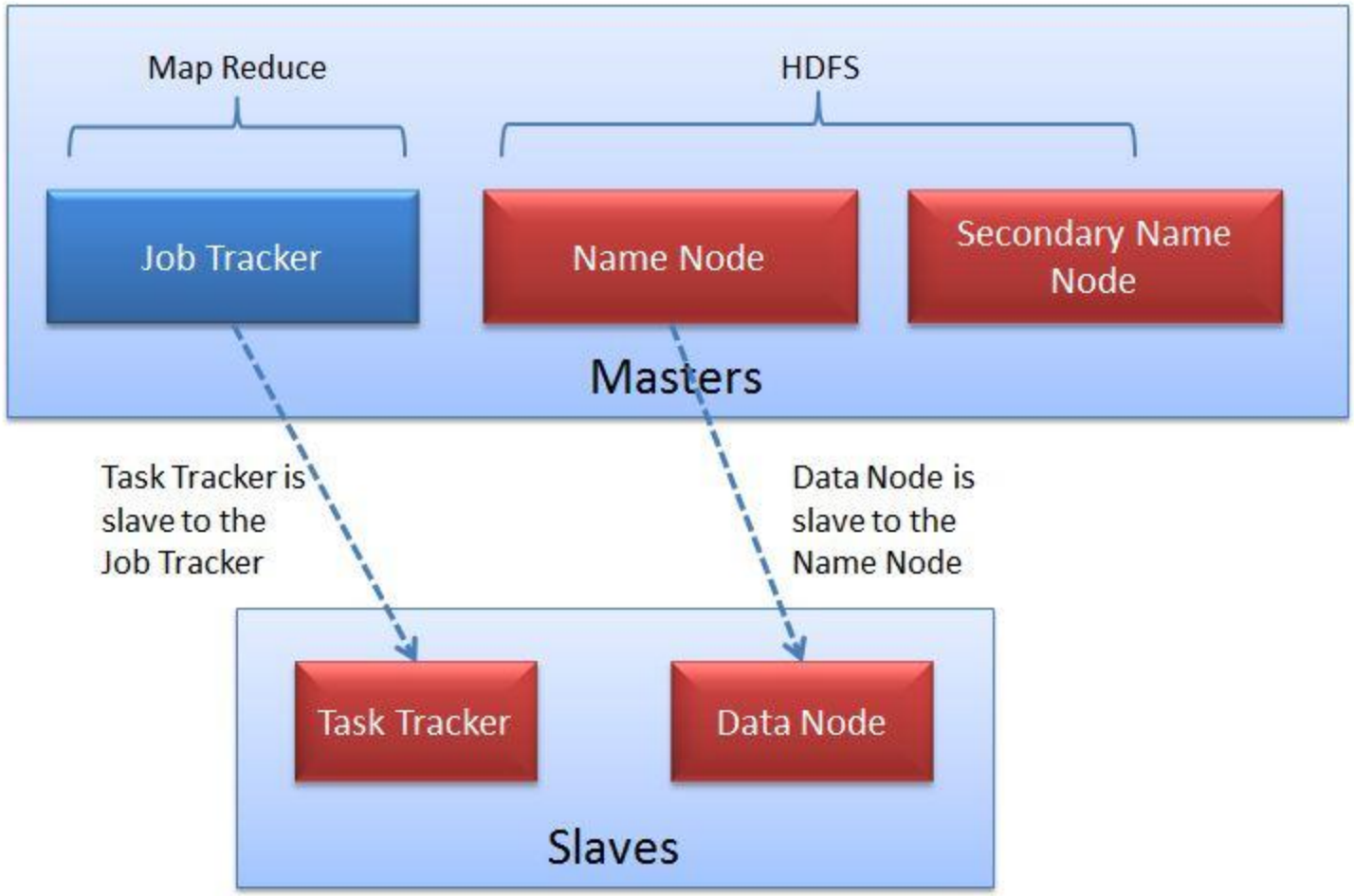
- **Not** a backup of the NameNode!
- NameNode keeps metadata in RAM, Secondary NameNode pulls these metadata and stores them in disk.
- Why?

Master: JobTracker (MR)

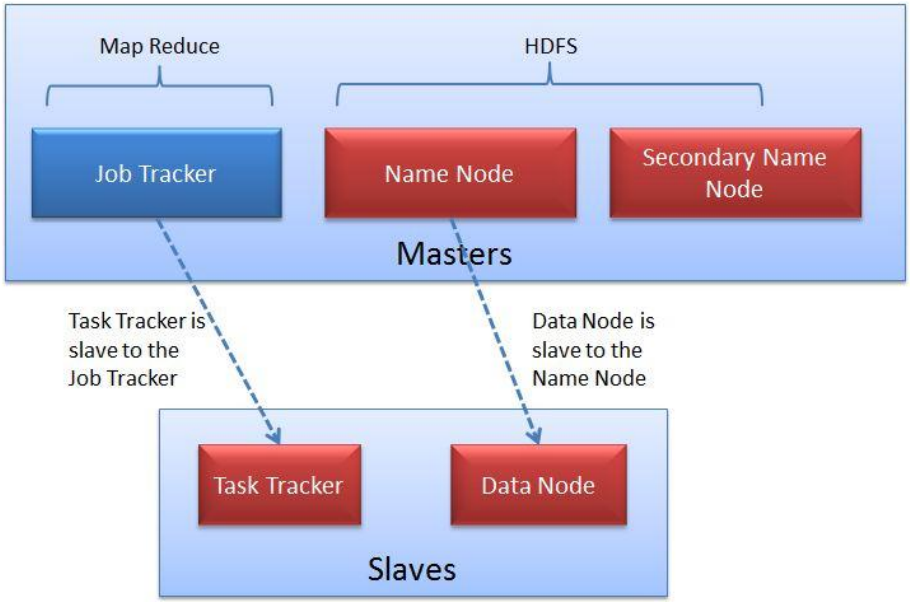


- coordinates the parallel processing of data
  - schedules clients jobs and allocates task to the task trackers
  - manages overall execution of MapReduce job
  - manages the resources of the cluster

Slaves


Map Reduce
HDFS
Job Tracker
Name Node
Secondary Name Node
Masters
Task Tracker is slave to the Job Tracker
Data Node is slave to the Name Node
Task Tracker
Data Node
Slaves

Slaves



- DataNode:
  - Store the data
- TaskTracker:
  - Process the computation

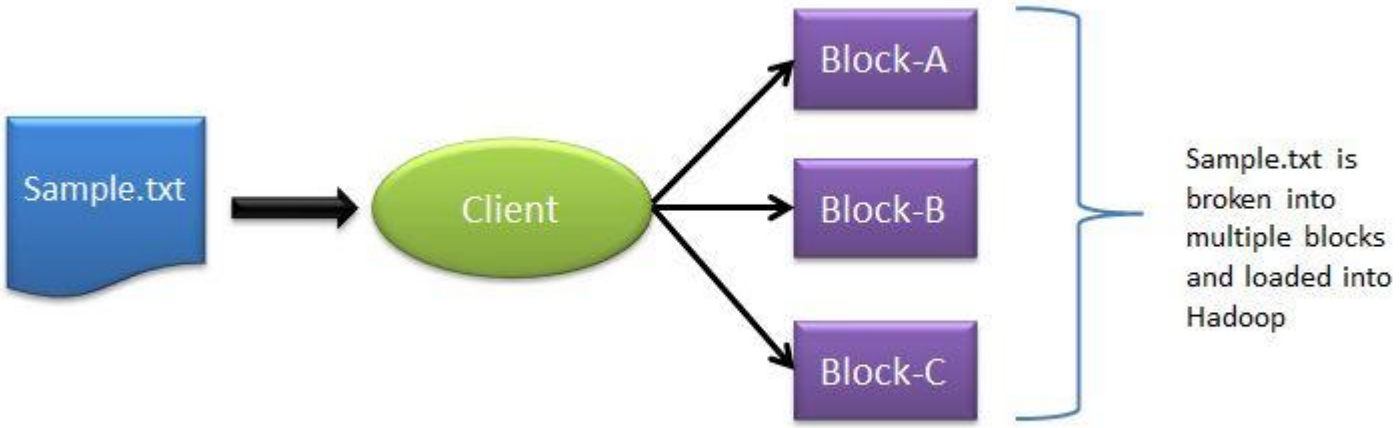
HDFS Workflow

HDFS Workflow

How a file gets loaded into the Hadoop Cluster?

HDFS Workflow

How a file gets loaded into the Hadoop Cluster?



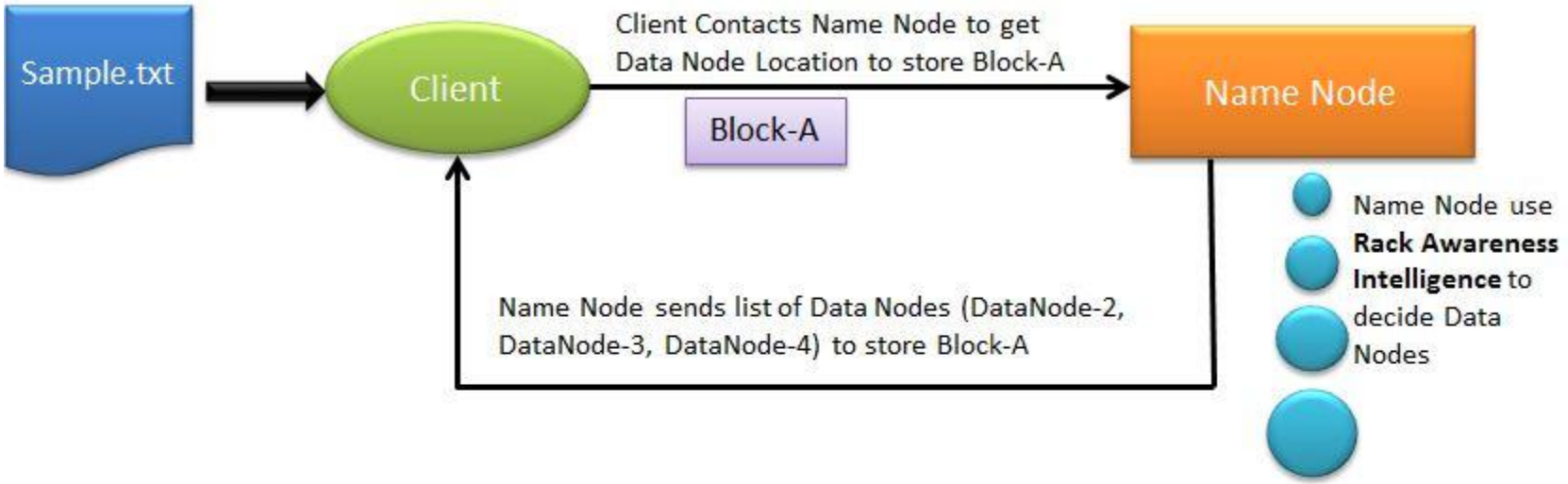
- Client machine loads files into cluster.
- Client breaks them into smaller chunks (Blocks)
- Client put these blocks on different machines (DataNodes)

HDFS Workflow

How does the Client knows that to which data nodes load the blocks?

HDFS Workflow

How does the Client knows that to which data nodes load the blocks?



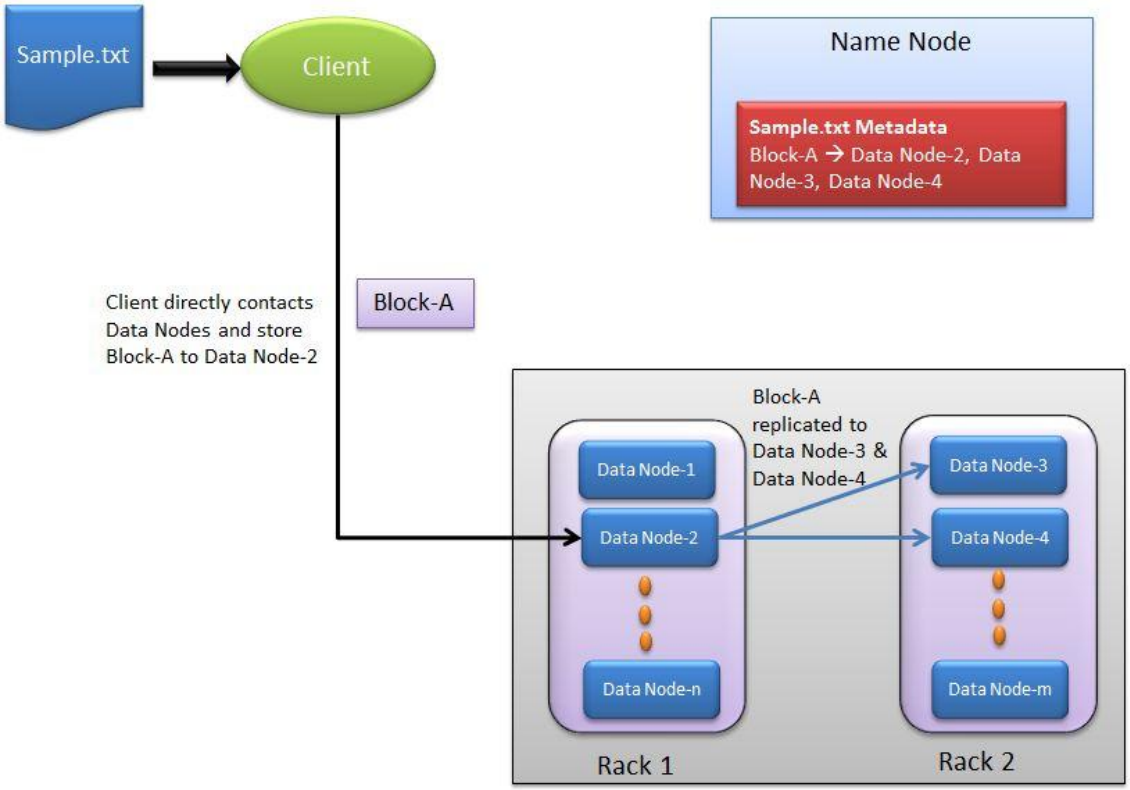
- For each of the data blocks Client contacts NameNode and in response NameNode sends an ordered list of the DataNodes.
- *Note: block is replicated to more than 1 data nodes. Why?*

HDFS Workflow

Who does the block replication?

HDFS Workflow

Who does the block replication?

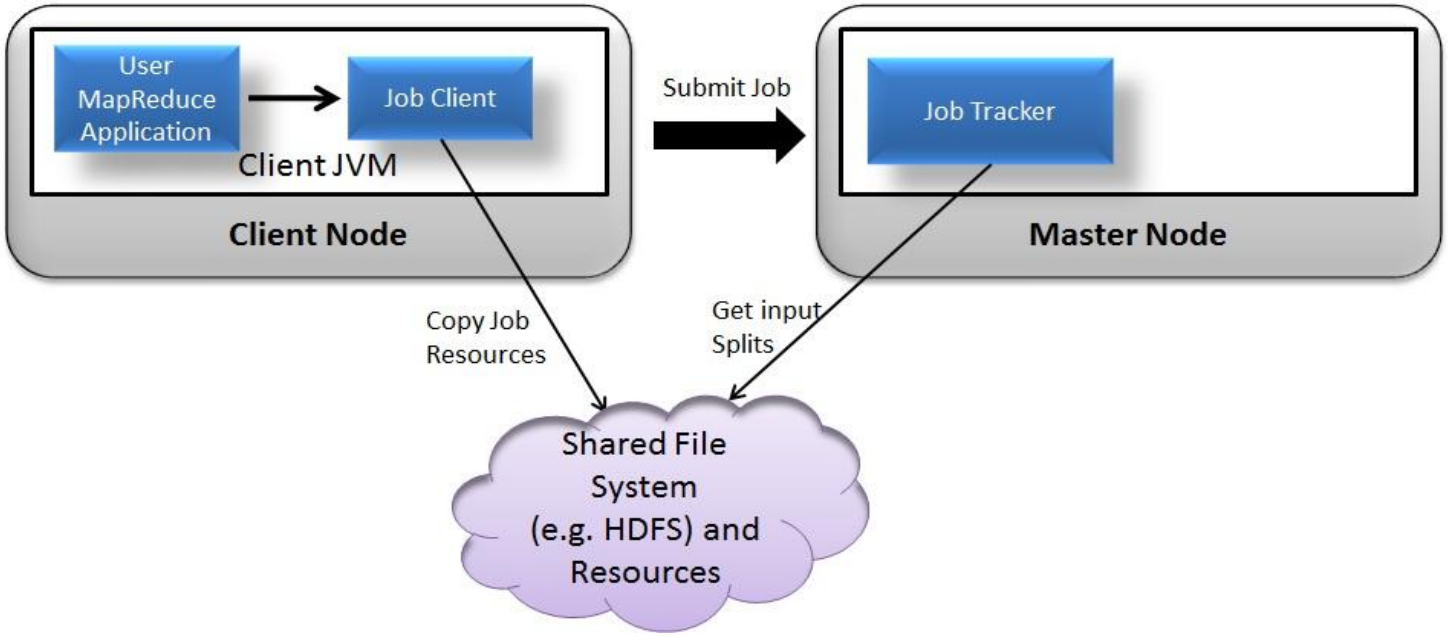


- Client write the data block directly to one DataNode.
- DataNodes then replicate the block to other Data nodes.

MapReduce Workflow

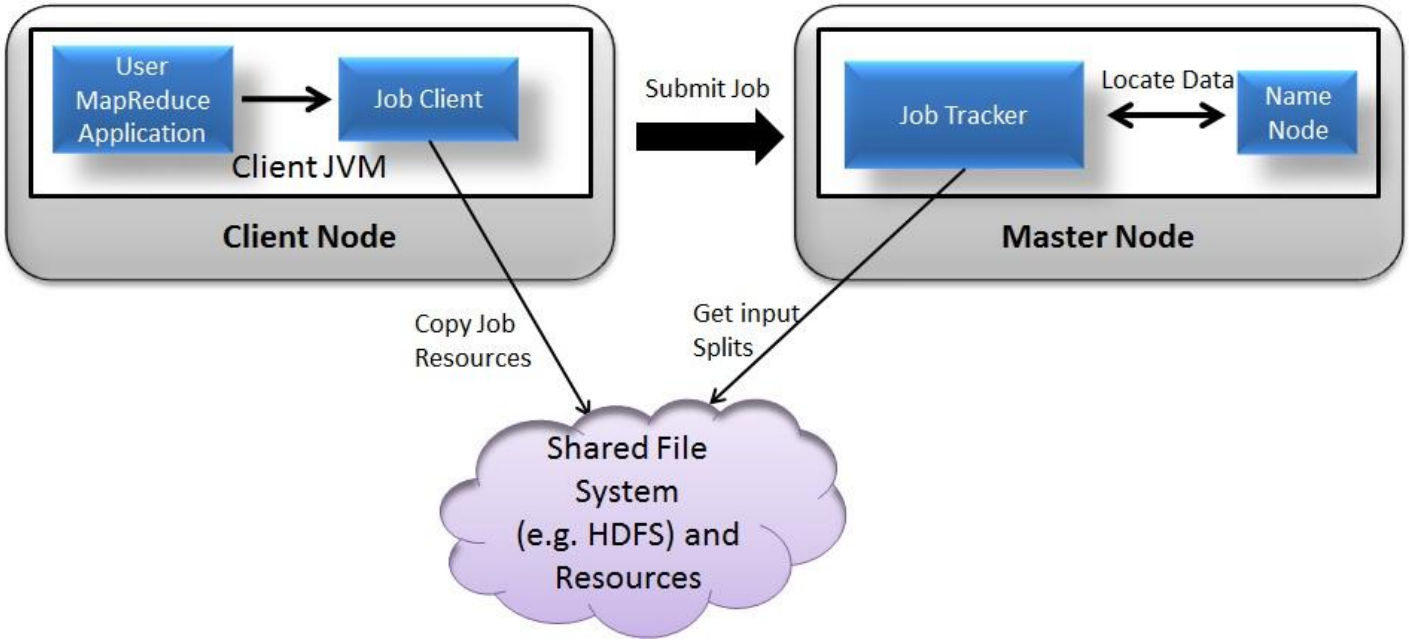
MapReduce Workflow

1- Client submits MapReduce job to Job Tracker



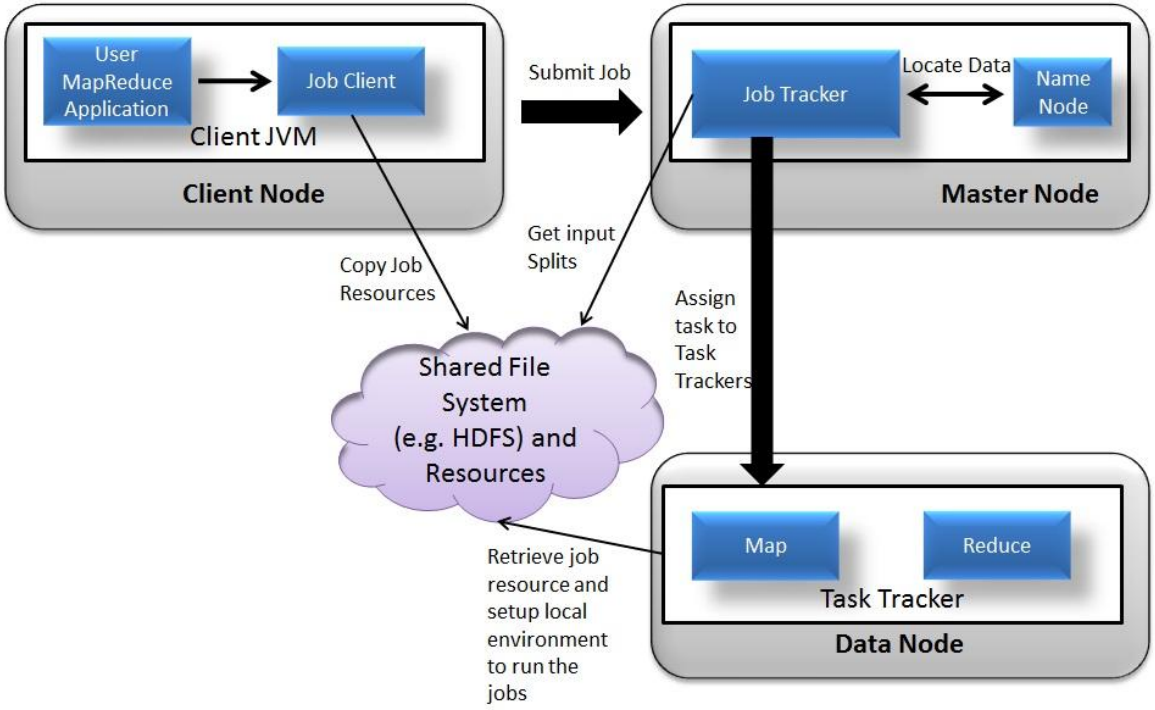
- When client/user submit a job, it goes to JobTracker.
- Client program contains all information like the map, combine and reduce function, input and output path of the data

2- JobTracker Manages/Controls Job



- It puts the job in a queue of pending jobs (order?)
- It first determines the number of splits, assign different map/reduce tasks to each TaskTracker.
- *How many map task we need?*

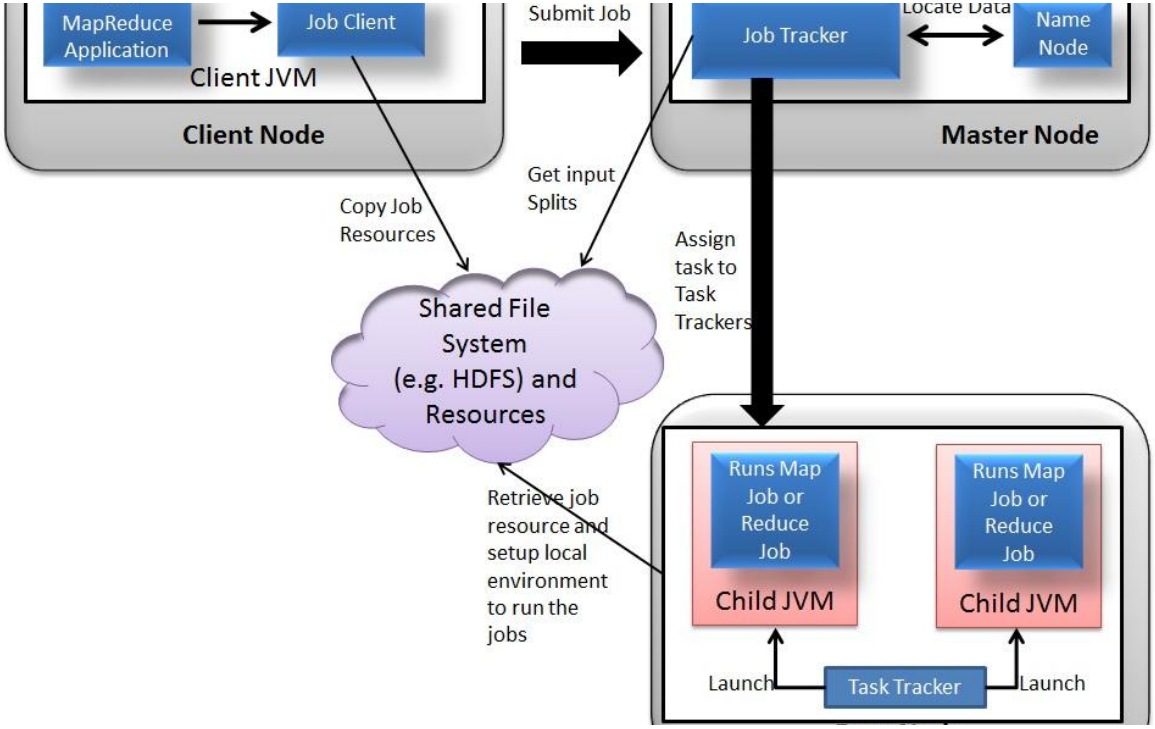
3- JobTracker assigns task to TaskTracker



- It prefers to run the task on a TaskTracker running on the same server as the DataNode.
- If not found, it looks for the machine in the same rack.
- *Any consideration of system load during this allocation?*

MapReduce Workflow

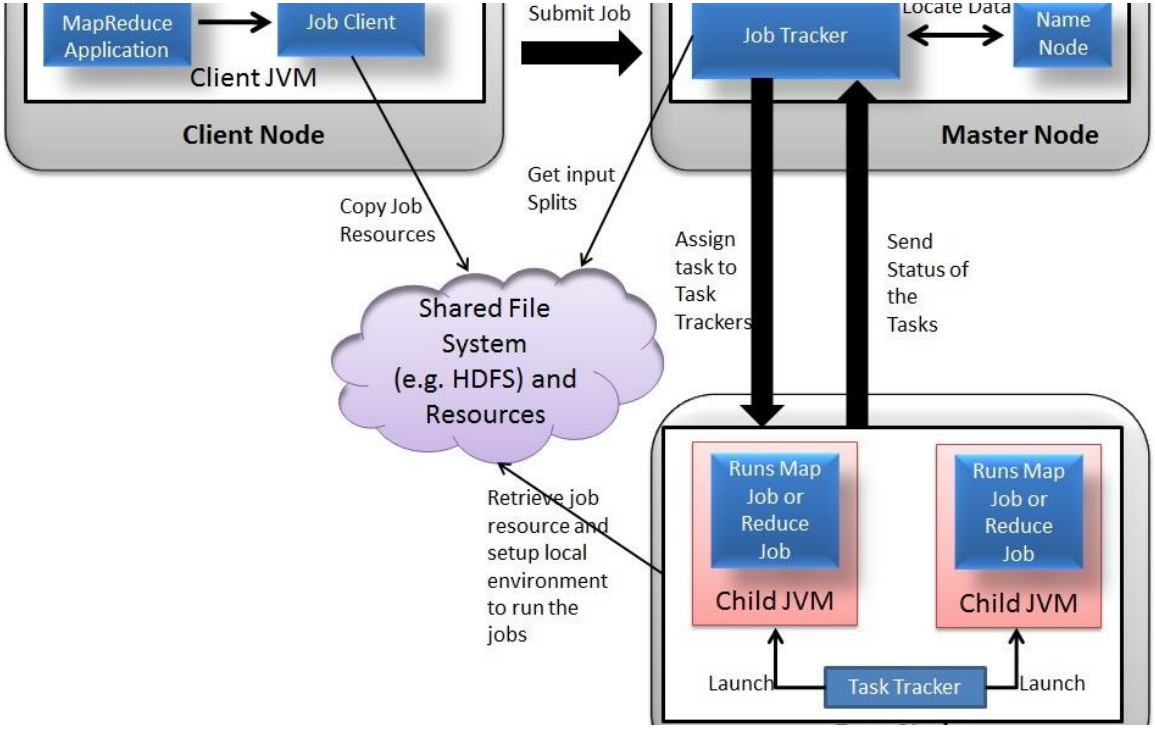
4- TaskTracker executes the task



- it copies files needed from the distributed cache to the local disk.
- it initiates tasks and reports progress to JobTracker.
- *Can a TaskTracker handle more than one tasks in parallel?*

MapReduce Workflow

5- TaskTracker sends notification to Job Tracker



- When all the map tasks are done by different task tracker they will notify the JobTracker.
- Job Tracker then ask the selected TaskTrackers to do the Reduce Phase.

MapReduce Workflow

6- Task recovery in failover situation

- Done at the TaskTracker level

7- Monitor TaskTracker

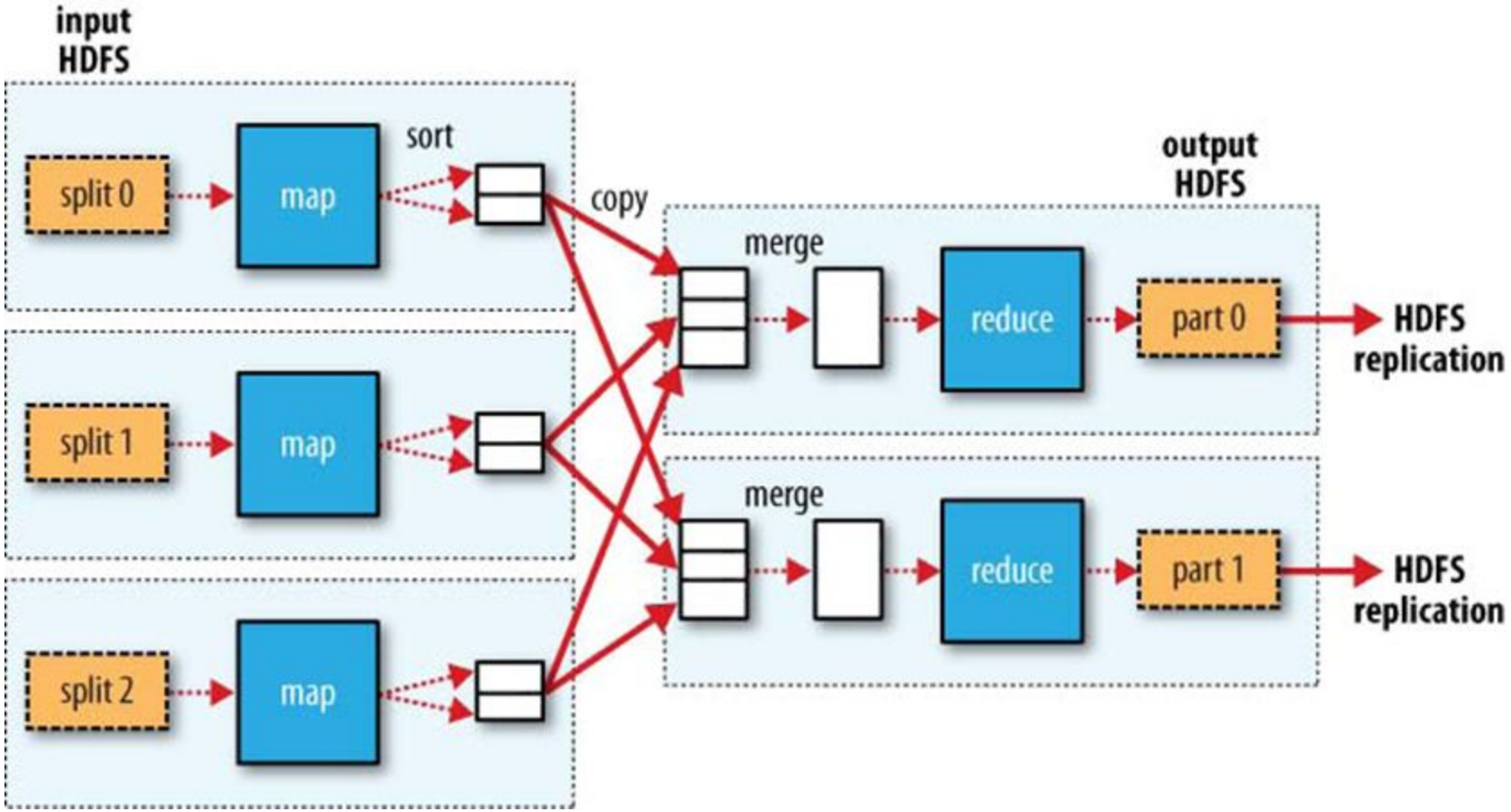
- A heartbeat is sent from the TaskTracker to the JobTracker every few minutes
- If Task Tracker do not submit heartbeat signals often enough, they are deemed to have failed and the work is scheduled on a different TaskTracker
- TaskTracker will notify the JobTracker when a task fails.
- The JobTracker decides what to do then:
  - resubmits the job elsewhere,
  - marks that specific record as something to avoid,
  - blacklists the TaskTracker as unreliable.

8- Job Completion!

- JobTracker updates its status

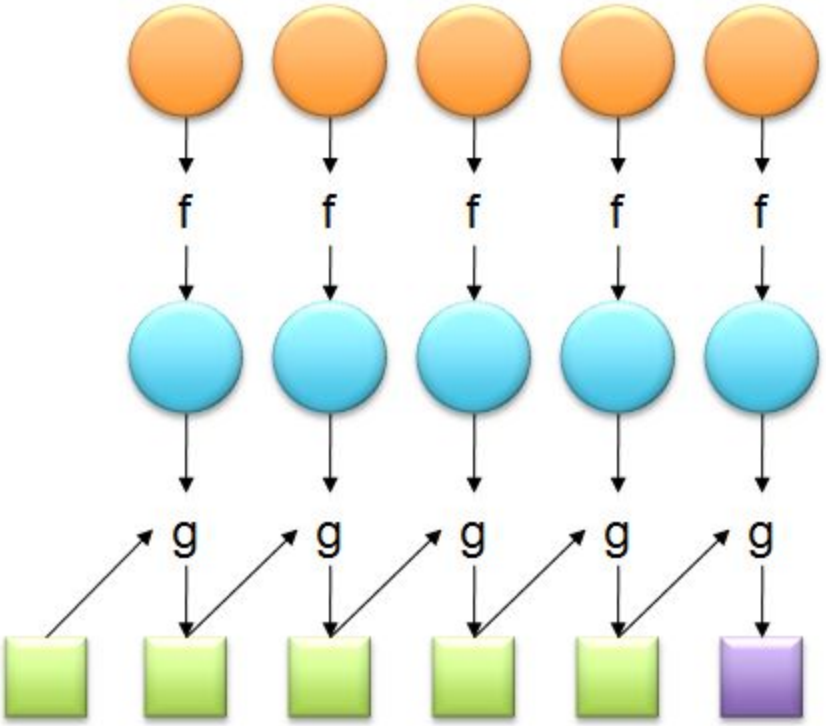
MapReduce Programming Model

MR Model

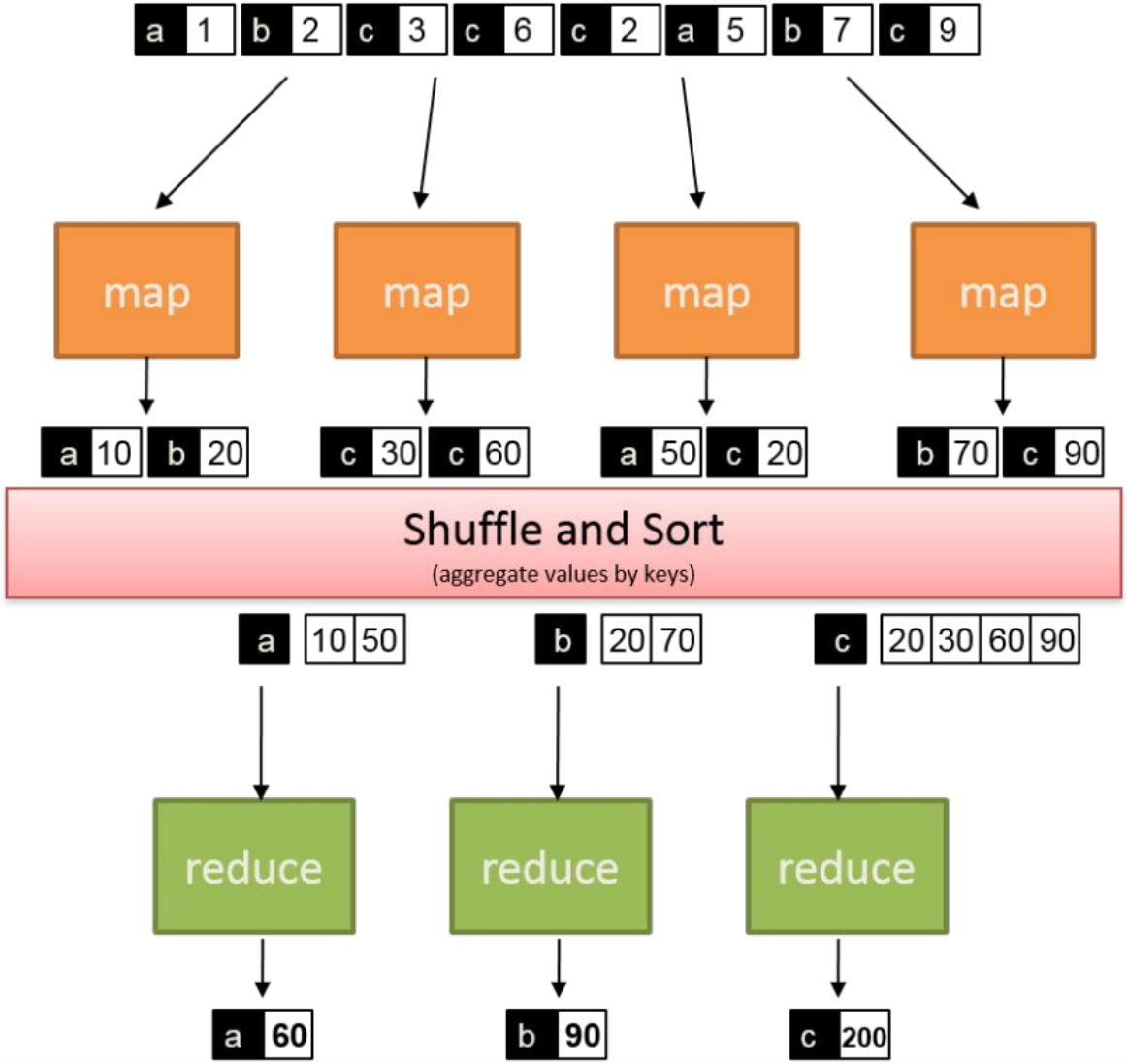

input
HDFS
split 0
map
sort
split 1
map
split 2
map
copy
merge
reduce
part 0
output
HDFS
HDFS
replication
merge
reduce
part 1
HDFS
replication

MR Model

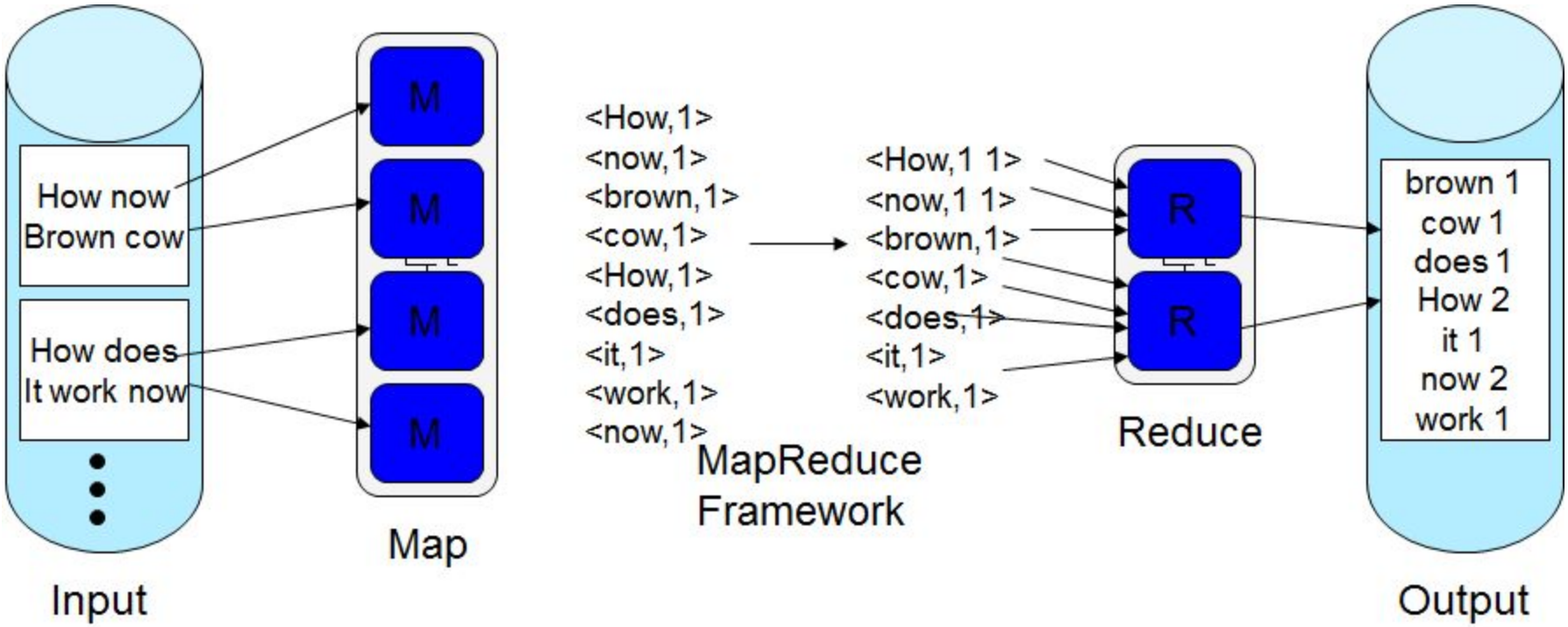
What are *f* and *g*?



MR Example 1: df.times(10).groupBy(key).sum()



MR Example 2: WordCount



MR Example 2: WordCount (Mapper)

```
#!/usr/bin/env scala
for (line <- io.Source.stdin.getLines) {
  line.split(" ").foreach(x=>println(s"$x\t1"))
}
```

MR Example 2: WordCount (Reducer)

```
#!/usr/bin/env scala
var wordCount = scala.collection.immutable.Map[String,Int]()
for (ln <- io.Source.stdin.getLines) {
    var   wordOne = ln.split("\t")
    if (wordCount.contains(wordOne(0))){
        wordCount += (wordOne(0) ->
(wordCount(wordOne(0))+wordOne(1).toInt))
    } else {
        wordCount += (wordOne(0) -> wordOne(1).toInt)
    }
}
```

Add comments to mapper and reducer

MR Challenge 2:

The example is very simple and has heaps of limitations. Identify the limitations and fix them.

Assessments

- Your submission will pass through plagiarism checking test
- You might be asked to present your work in a 5-10 minutes interview
- You only have one attempt with limited time (except the last project)

MONASH
University

Any (simple) Question?


MONASH
University
GROUP
OF EIGHT
AUSTRALIA