Introduction to Hadoop

CSE 569

Content

- Distributed system, DFS
- Hadoop
- Architecture of Hadoo
- Hadoop usage at Facebook

Operating systems

- Operating system - Software that supervises and controls tasks on a computer. Individual OS:
  - Batch processing ◊ jobs are collected, placed in a queue, no interaction with job during processing
  - Time shared ◊ computing resources are provided to different users, interaction with program during execution
  - RT systems ◊ fast response, can be interrupted

Distributed Systems

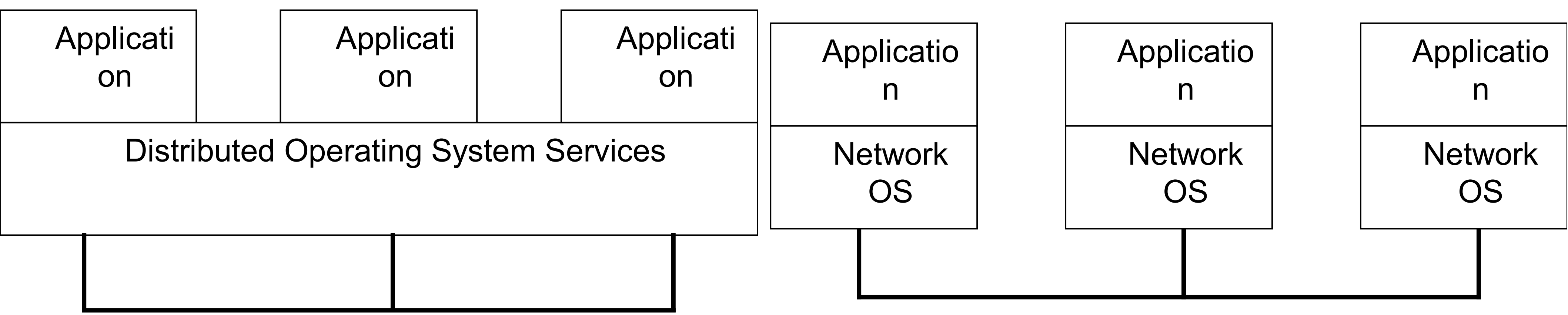
- Consists of a number of computers that are connected and managed so that they automatically share the job processing load among the constituent computers.
- A distributed operating system is one that appears to its users as a traditional uniprocessor system, even though it is actually composed of multiple processors.
- It gives a single system view to its users and provides a single service.
- Users are transparent to location of files. It provides a virtual computing environment.

Eg The Internet, ATM banking networks, mobile computing networks, Global Positioning Systems and Air Traffic Control

DISTRIBUTED SYSTEM IS A COLLECTION OF INDEPENDENT COMPUTERS THAT APPEARS TO IS USERS AS A SINGLE COHERENT SYSTEM

Network Operating System

- In a **network operating system** the users are aware of the existence of multiple computers.
- The operating system of individual computers must have facilities to have communication and functionality.
- Each machine runs its own OS and has its own user.
- Remote login and file access
- Less transparent but more independency



Distributed OS

Networked OS

DFS

- **Resource sharing** is the motivation behind distributed Systems. To share files ◊ file system
- File System is responsible for the organization, storage, retrieval, naming, sharing, and protection of files.
- The file system is responsible for **controlling** access to the data and for performing **low-level** operations such as buffering frequently used data and issuing disk I/O requests
- The goal is to allow users of physically distributed computers to **share data and storage resources** by using a **common file system**.

What is Hadoop

- Open source software platform for scalable, distributed computing
- Hadoop provides fast and reliable analysis of both structured data and unstructured data
- Apache Hadoop software library is essentially a framework that allows for the distributed processing of large datasets across clusters of computers using a simple programming model.
- Hadoop can scale up from single servers to thousands of machines, each offering local computation and storage.

What is Hadoop?..Contd

- **Software platform that lets one easily write and run applications that process vast amounts of data. It includes:**
  - **MapReduce – offline computing engine**
  - **HDFS – Hadoop distributed file system**

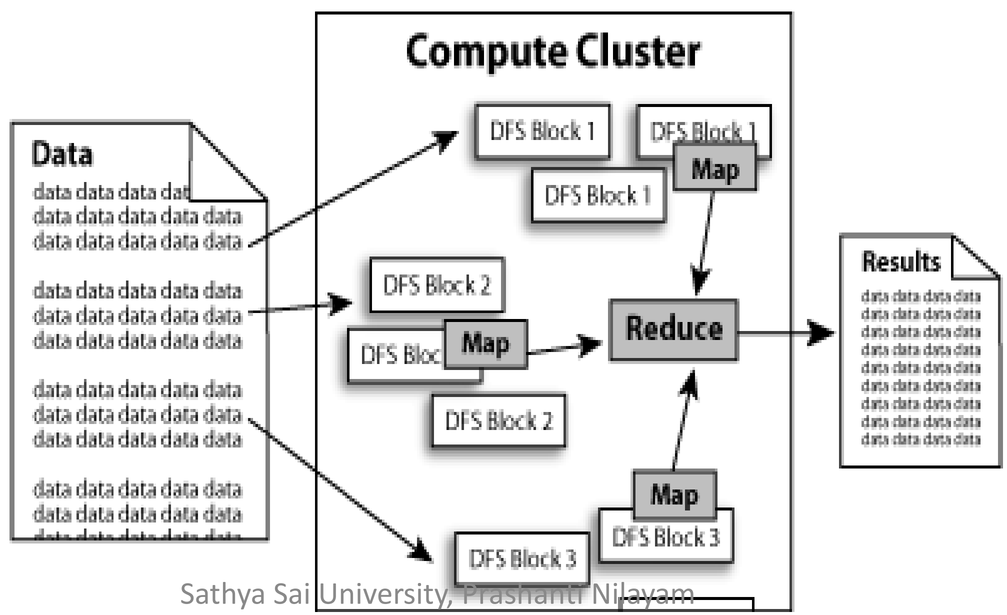
- **Here's what makes it especially useful:**
  - **Scalable:** It can reliably store and process petabytes.
  - **Economical:** It distributes the data and processing across clusters of commonly available computers (in thousands).
  - **Efficient:** By distributing the data, it can process it in parallel **on the nodes where the data is located.**
  - **Reliable:** It automatically maintains multiple copies of data and automatically redeploys computing tasks based on failures.

Who uses Hadoop?

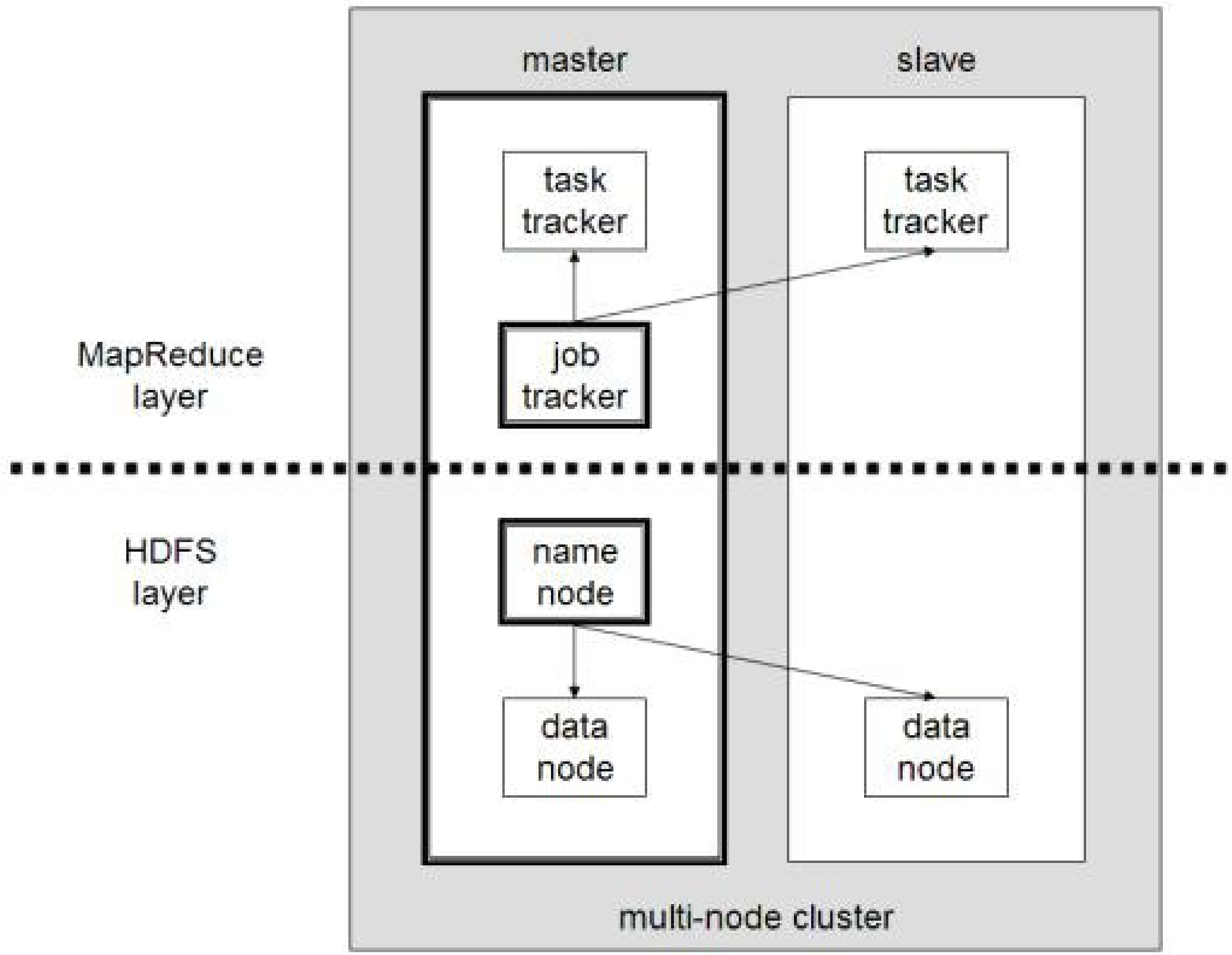
- Amazon/A9
- Facebook
- Google
- IBM
- Joost
- Last.fm
- New York Times
- PowerSet
- Veoh
- Yahoo!

What does it do?

- Hadoop implements Google's MapReduce, using HDFS
- MapReduce divides applications into many small blocks of work.
- HDFS creates multiple replicas of data blocks for reliability, placing them on compute nodes around the cluster.
- MapReduce can then process the data where it is located.
- Hadoop 's target is to run on clusters of the order of 10,000-nodes.



Hadoop Cluster Architecture:



HDFS

- HDFS uses a master/slave architecture where master consists of a single NameNode that manages the file system metadata and one or more slave DataNodes that store the actual data.
- A file in an HDFS namespace is split into several blocks and those blocks are stored in a set of DataNodes. The NameNode determines the mapping of blocks to the DataNodes.
- The DataNodes takes care of read and write operation with the file system. They also take care of block creation, deletion and replication based on instruction given by NameNode.

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

NameNode

- **DFS Master “Namenode”**
  - Manages the **file system namespace**
  - **Controls** read/write access to files
  - Manages **block replication**
  - **Checkpoints** namespace and journals namespace changes for **reliability**

Metadata of Name node in Memory

- The entire **metadata is in main memory**
- No demand paging of FS metadata

Types of Metadata:

List of files, file and chunk namespaces; list of blocks, location of replicas; file attributes etc.

DataNode

- Stores the actual data in HDFS
- Can run on any underlying filesystem (ext3/4, NTFS, etc)
- Notifies NameNode of what blocks it has
- NameNode replicates blocks 2x in local rack, 1x elsewhere

DATA NODES

- Serve **read/write requests** from clients
- Perform **replication** tasks upon instruction by namenode

Data nodes act as:

1) A Block Server
   - Stores **data** in the local file system
   - Stores **metadata** of a block (e.g. CRC)
   - Serves data and metadata to Clients
2) Block Report: Periodically sends a **report of all existing blocks** to the NameNode
3) Periodically sends heartbeat to NameNode (detect node failures)
4) Facilitates Pipelining of Data (to other specified DataNodes)

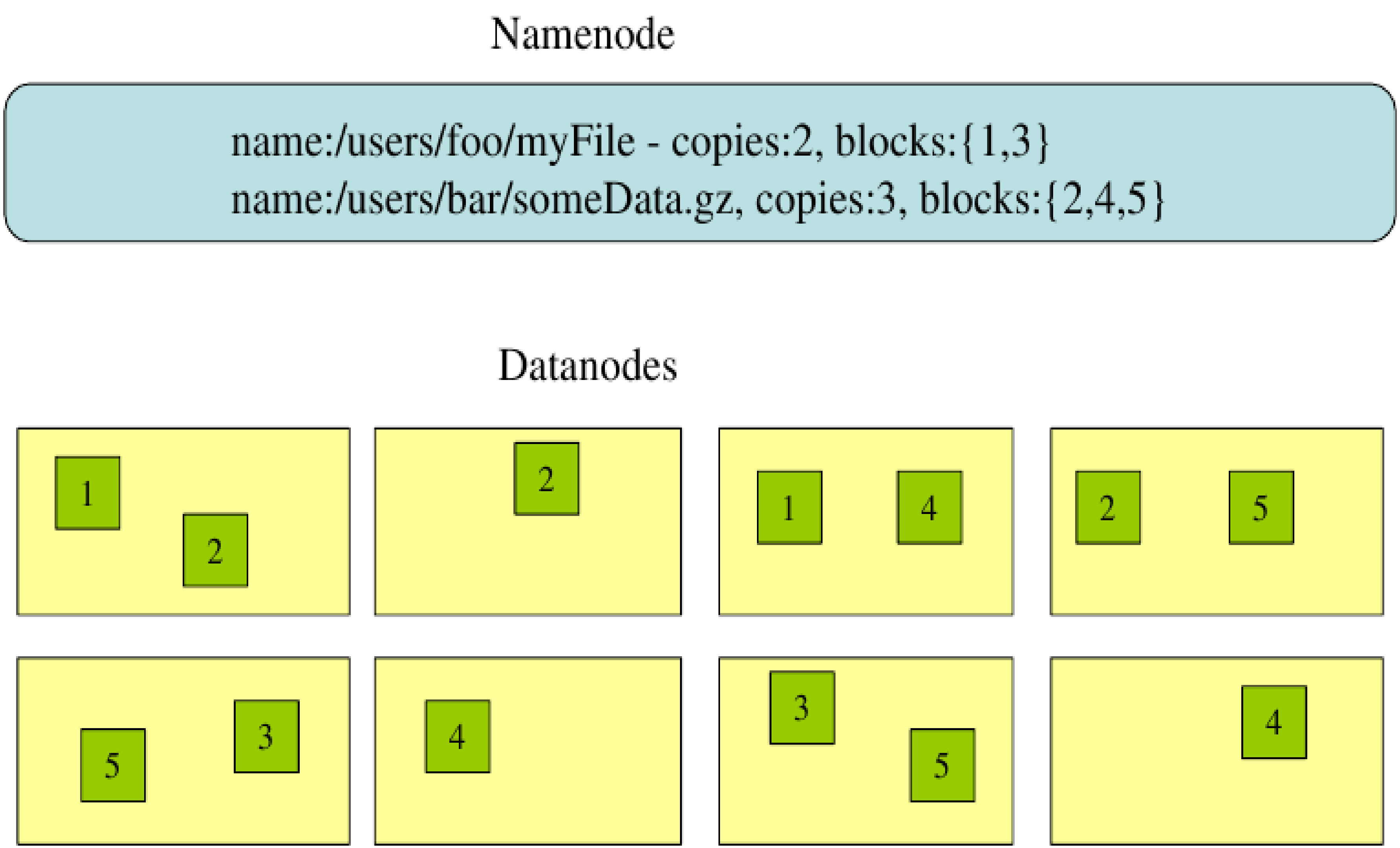
- Data Model
  - Data is **organized** into files and directories
  - Files are divided into uniform sized **blocks** and distributed across cluster nodes
  - **Replicate blocks** to handle hardware failure
  - **Checksums** of data for corruption detection and recovery

- **Assumes commodity hardware that fails**
  - Files are **replicated** to handle hardware failure
  - **Checksums** for corruption detection and recovery
  - **Continues operation** as nodes / racks added / removed

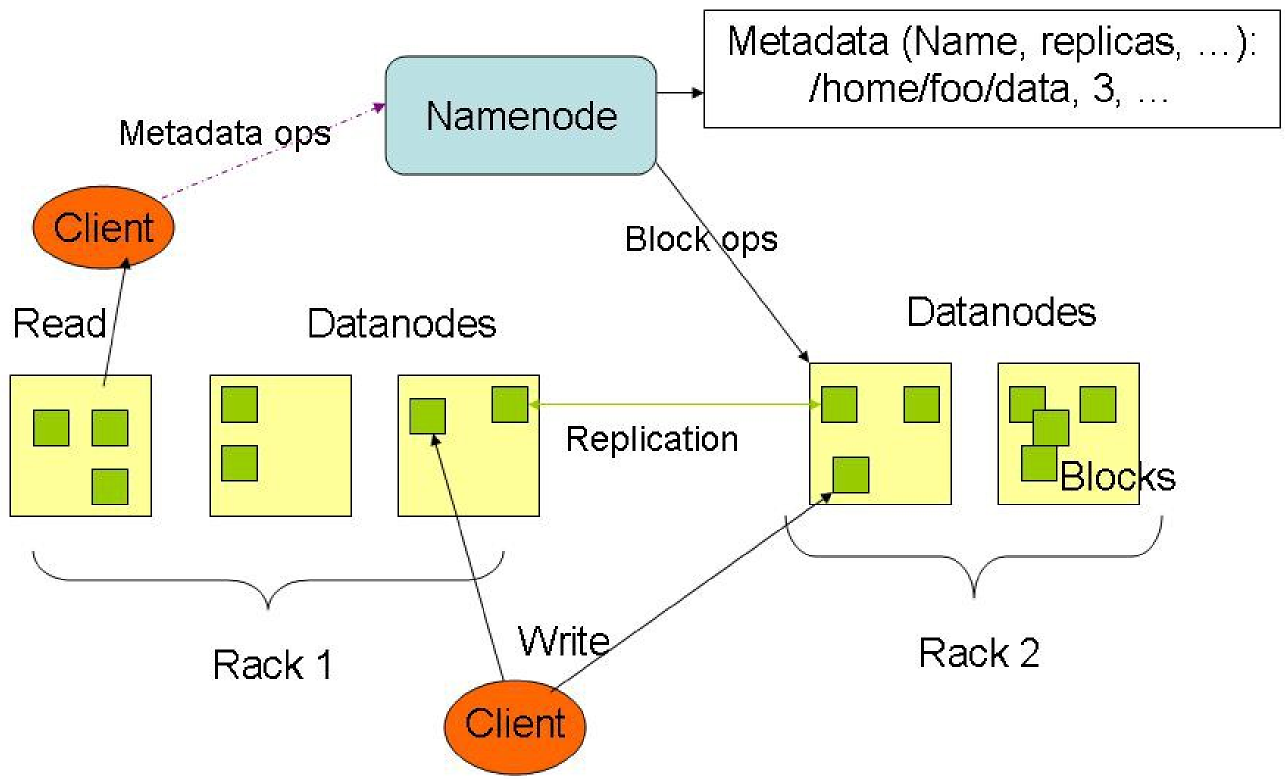
- **Optimized for fast batch processing**
  - Data location exposed to allow **computes to move to data**
  - Stores data in **chunks/blocks** on every node in the cluster
  - Provides **VERY high aggregate bandwidth**

- **Files are broken in to large blocks.**
  - Typically **128 MB** block size
  - Blocks are **replicated** for reliability
  - One replica on **local node,**
    another replica on a **remote rack,**
    Third replica on **local rack,**
    Additional replicas are **randomly placed**
- **Understands rack locality**
  - Data placement exposed so that **computation** can be **migrated** to data
- **Client talks to both NameNode and DataNodes**
  - **Data is not sent through the namenode**, clients access data directly from DataNode
  - **Throughput** of file system scales nearly linearly with the number of nodes.

Block Placement



HDFS Architecture



MapReduce

The term MapReduce actually refers to the following two different tasks that Hadoop programs perform:

The Map Task: This is the first task, which takes input data and converts it into a set of data, where individual elements are broken down into tuples key/value pairs.

The Reduce Task: This task takes the output from a map task as input and combines those data tuples into a smaller set of tuples. The reduce task is always performed after the map task.

How does MapReduce work?

- The run time partitions the input and provides it to different Map instances;
- Map (key, value) ◊ (key', value')
- The run time collects the (key', value') pairs and distributes them to several Reduce functions so that each Reduce function gets the pairs with the same key'.
- Each Reduce produces a single (or zero) file output.
- Map and Reduce are user written functions

Example MapReduce: To count the occurrences of words in the given set of documents

map(String key, String value):

// key: document name; value: document contents; map (k1,v1) ◊ list(k2,v2)

for each word w in value: EmitIntermediate(w, "1");

(Example: If input string is (“Saibaba is God. I am I”), Map produces {<“Saibaba”,1”>, <“is”, 1>, <“God”, 1>, <“I”,1>, <“am”,1>,<“I”,1>}

reduce(String key, Iterator values):

// key: a word; values: a list of counts; reduce (k2,list(v2)) ◊ list(v2)

int result = 0;

for each v in values:

result += ParseInt(v);

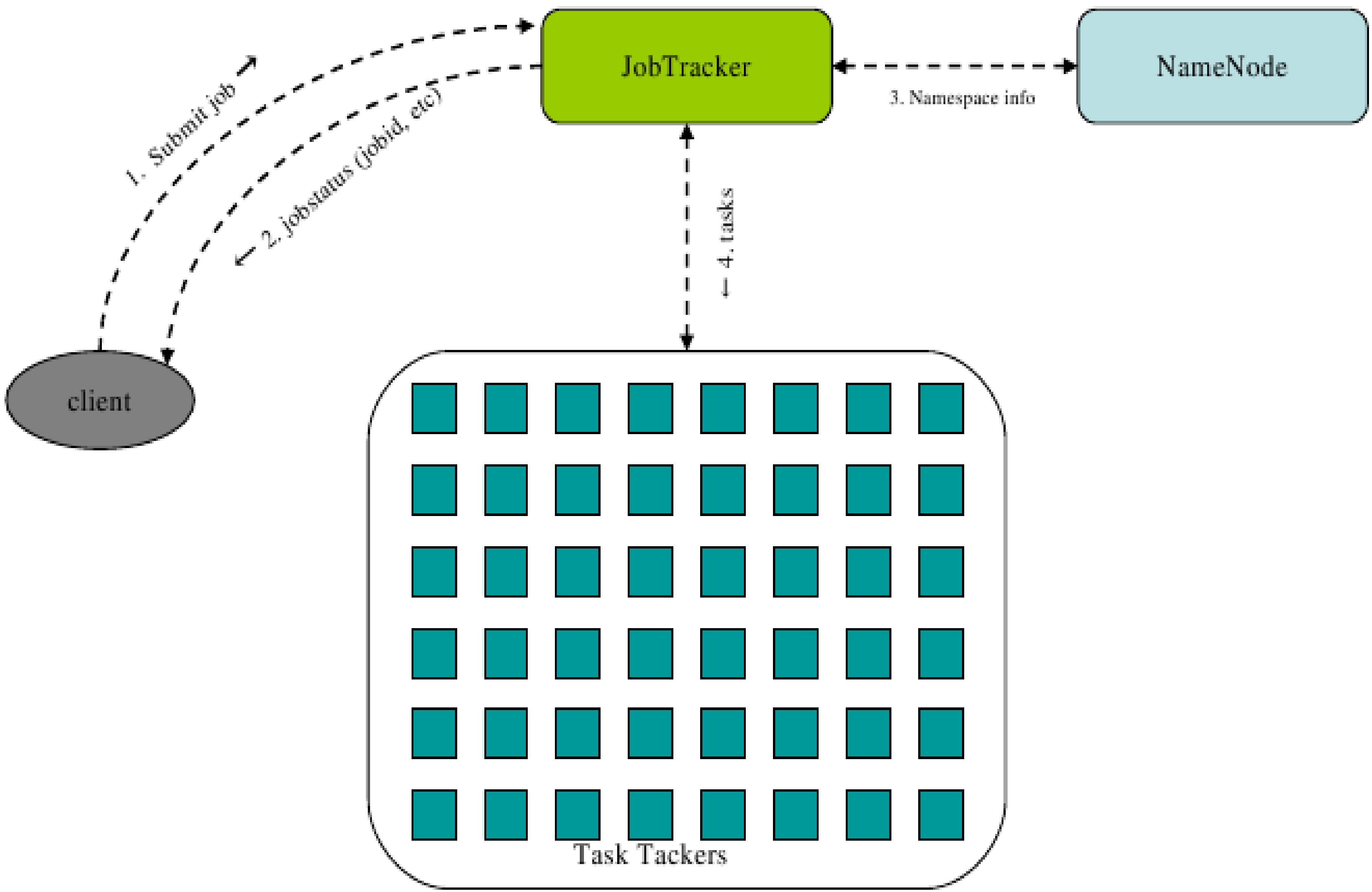
Emit(AsString(result));

(Example: reduce(“I”, <1,1>) ◊ 2)

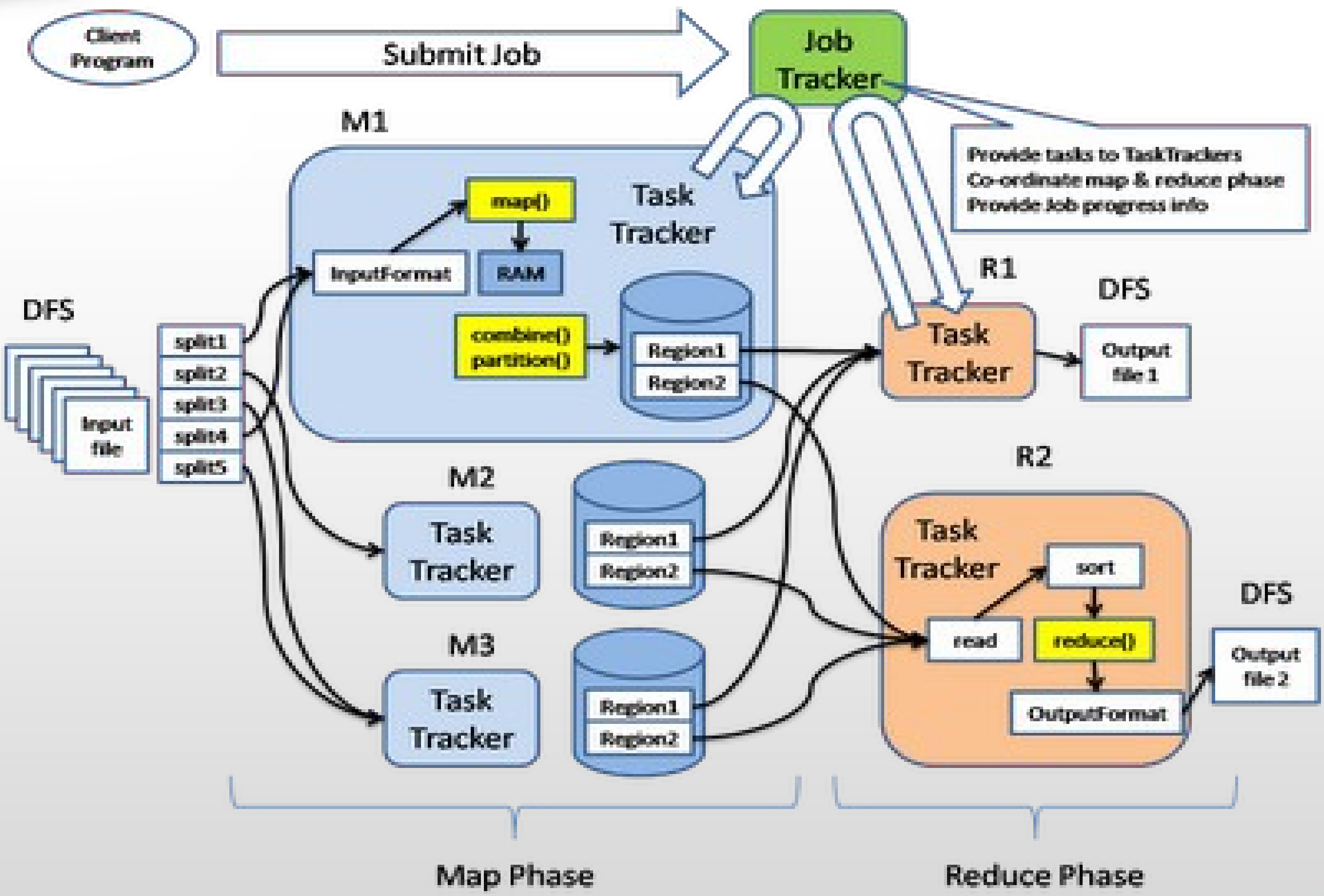
MapReduce Engine

- **Map/Reduce Master "Jobtracker"**
  - **Accepts** MR jobs submitted by users
  - **Assigns** Map and Reduce tasks to Tasktrackers
  - **Monitors** task and tasktracker status, re-executes tasks upon failure
- **Map/Reduce Slaves "Tasktrackers"**
  - **Run** Map and Reduce tasks upon instruction from the Jobtracker
  - **Manage** storage and transmission of intermediate output.

JOBTRACKER, TASKTACKER AND JOBCLIENT



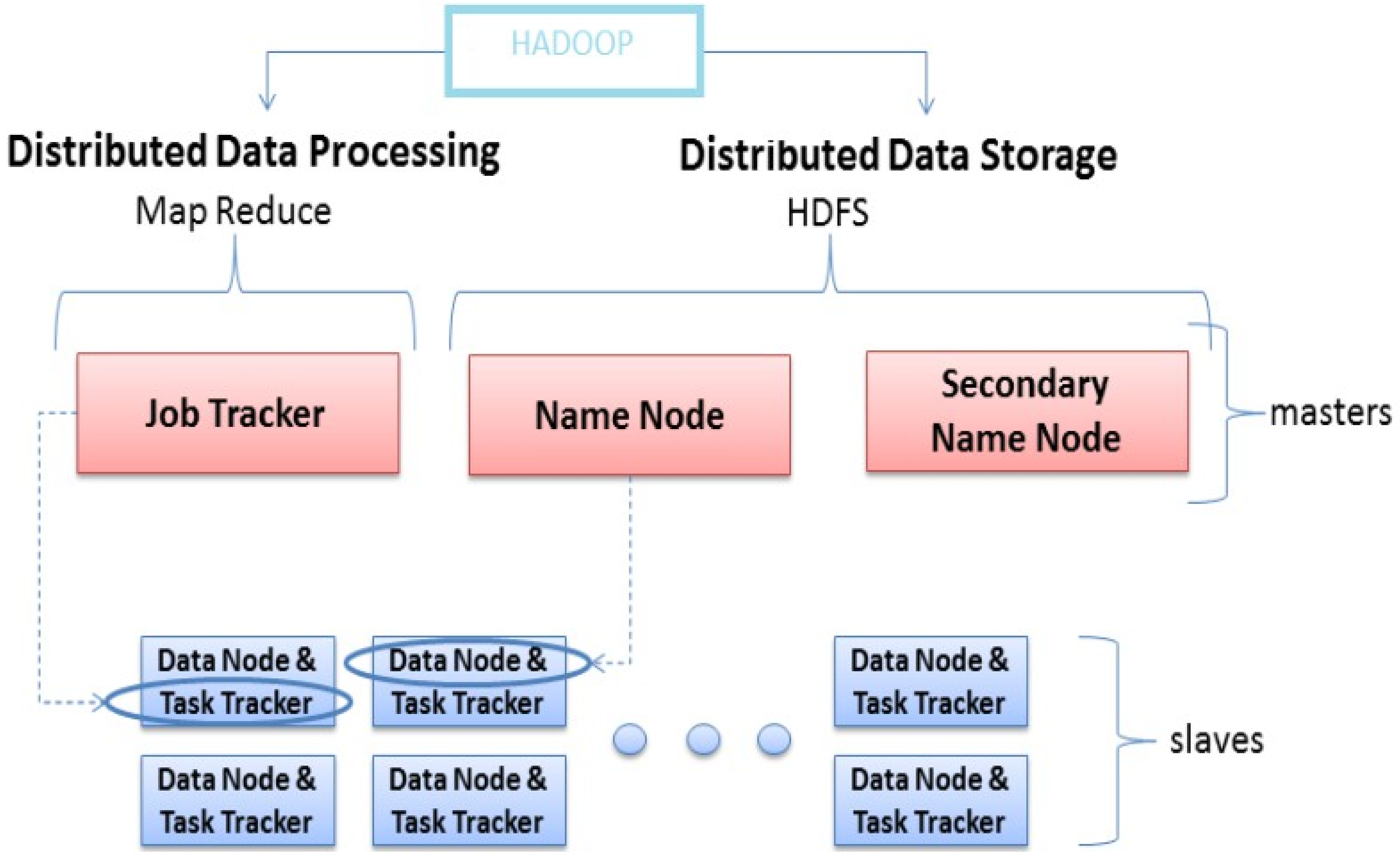
Hadoop's Architecture: MapReduce Engine



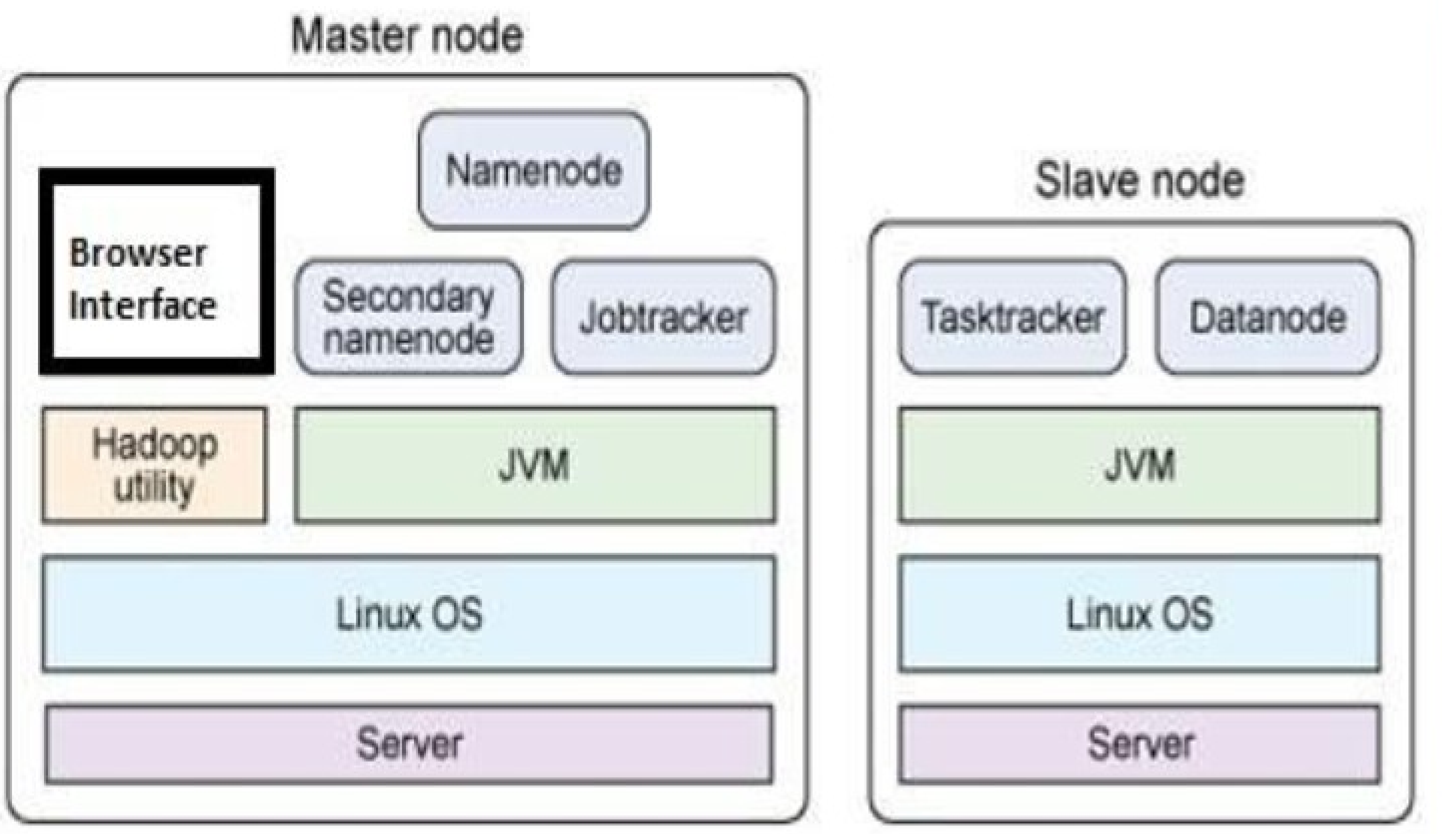
Hadoop at Facebook

- **Production cluster**
  - 4800 cores, 600 machines, 16GB per machine – April 2009
  - 8000 cores, 1000 machines, 32 GB per machine – July 2009
  - 4 SATA disks of 1 TB each per machine
  - 2 level network hierarchy, 40 machines per rack
  - Total cluster size is 2 PB, projected to be 12 PB in Q3 2009

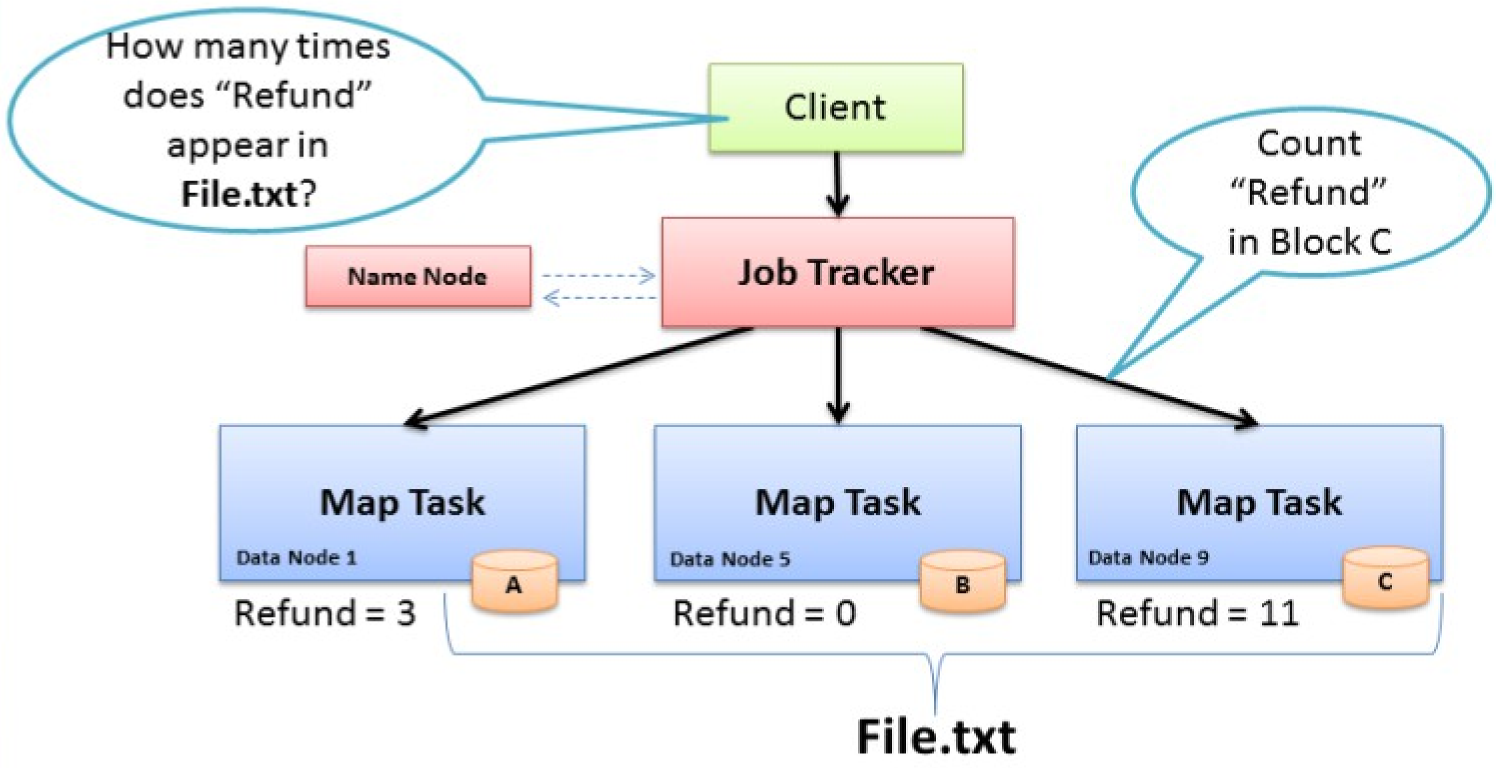
- **Test cluster**
  - 800 cores, 16GB each


HADOOP
Distributed Data Processing
Map Reduce
Distributed Data Storage
HDFS
Job Tracker
Name Node
Secondary Name Node
masters
Data Node & Task Tracker
Data Node & Task Tracker
Data Node & Task Tracker
Data Node & Task Tracker
Data Node & Task Tracker
Data Node & Task Tracker
slaves

HDFS Cluster Architecture

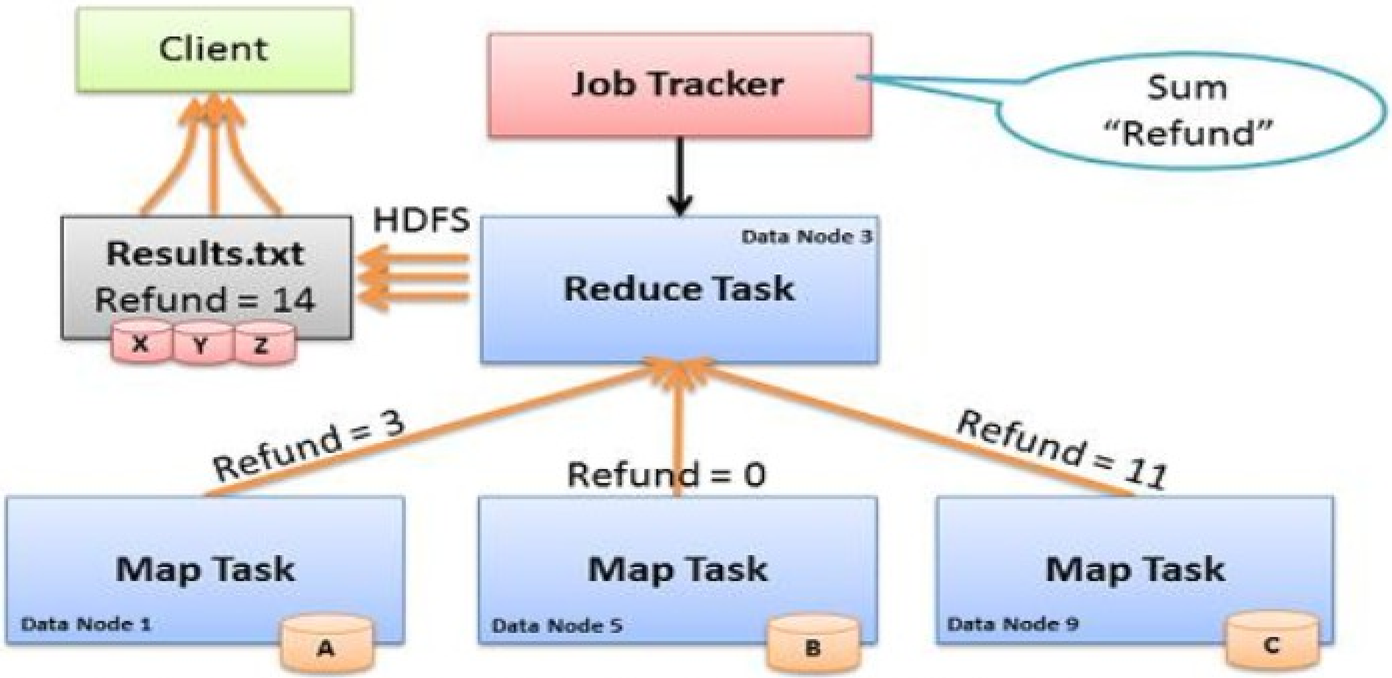


MAP Task



- **Map:** "Run this computation on your local data"
- Job Tracker delivers Java code to Nodes with local data

Reduce Task



- **Reduce:** "Run this computation across Map results"
- Map Tasks send output data to Reducer over the network
- Reduce Task data output written to and read from HDFS