UNDERSTANDING HADOOP HDFS & MAPREDUCE

CLASS-II

Instructor: Palash Gupta

Re-Cap

Hello Everyone!

Last Week, we went through fundamental of Big Data & Hadoop.

- Introduction
- What is Big Data?
- How big is Big Data?
- What are we trying to solve?
- Types of Data Structure
- Hadoop System Principle
- History of Hadoop
- Comparison with RDBMS
- Hadoop Eco System
- Hadoop Distribution
- Supported OS and HW

What we are going to Cover today?

- HDFS Concept
- HDFS Architecture
- Introduction to Map Reduce
- Working Procedure of Map Reduce

Introduction to Hadoop

Hadoop includes two main functions:

- ❑ HDFS
- ❑ MapReduce

❖ HDFS:

-Hadoop distributed file system

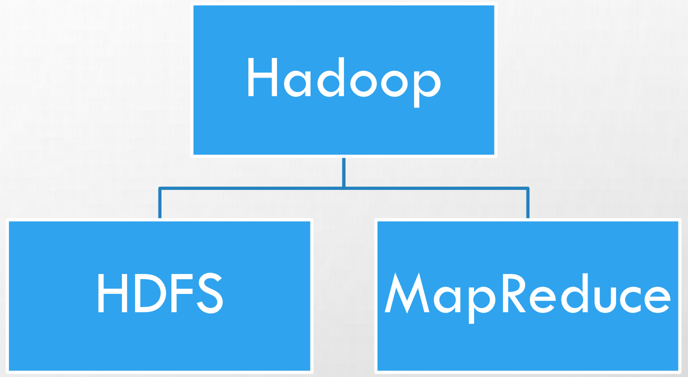
-It is used to store large data set in a distributed environment using commodity machine

-HDFS implementation is modeled after GFS, Google Distributed File system, thus you can read the first paper on this, to be found here: http://labs.google.com/papers/gfs.html.

❖ MapReduce:

-This is distributing computing framework

-It is used to compute large data set using a distributed framework to reduce the latency while using cost effective method



HDFS Concept

❑ **Assumptions and Goals**

-**Hardware Failure:** Detection of faults and quick, automatic recovery from them is a core architectural goal of HDFS.

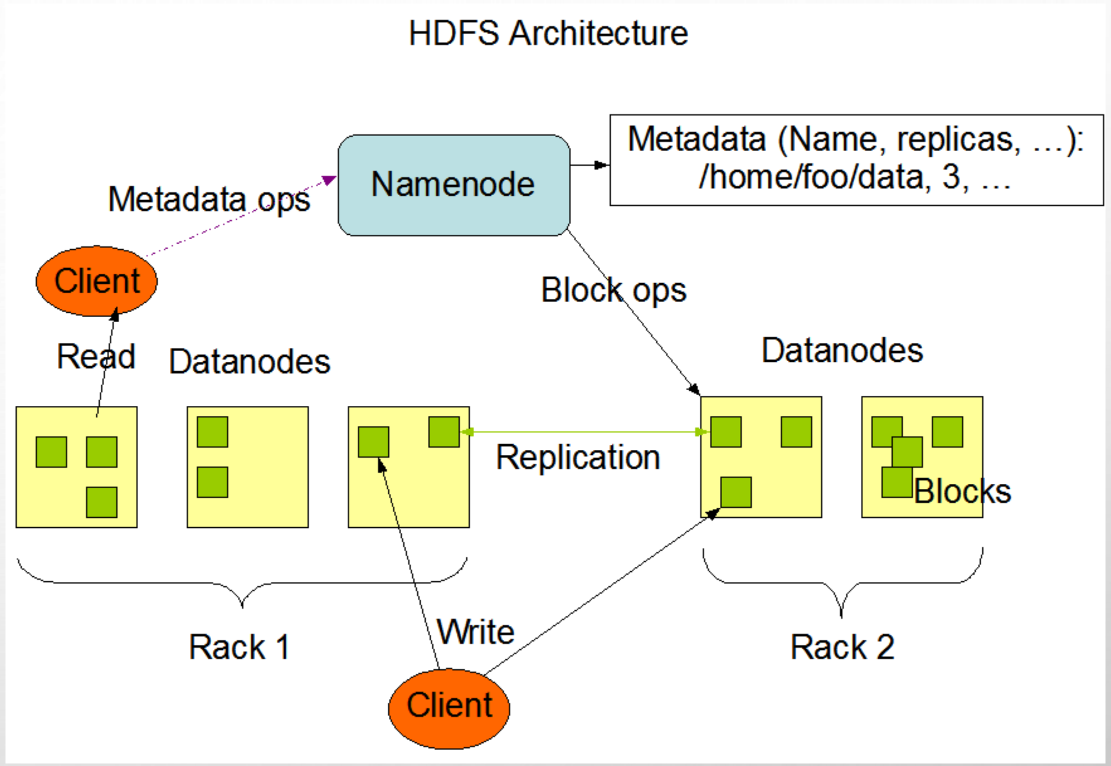
-**Streaming Data Access:** HDFS is designed more for batch processing rather than interactive use by users. The emphasis is on high throughput of data access rather than low latency of data access.

-**Large Data Sets:** HDFS is tuned to support large files. It should provide high aggregate data bandwidth and scale to hundreds of nodes in a single cluster.

-**Simple Coherency Model:** HDFS applications need a write-once-read-many access model for files. A file once created, written, and closed need not be changed.

-**Moving Computation is Cheaper than Moving Data:** A computation requested by an application is much more efficient if it is executed near the data it operates on.

-**Portability Across Heterogeneous Hardware and Software Platforms**



HDFS Architecture

- ❑ **Name Nodes**

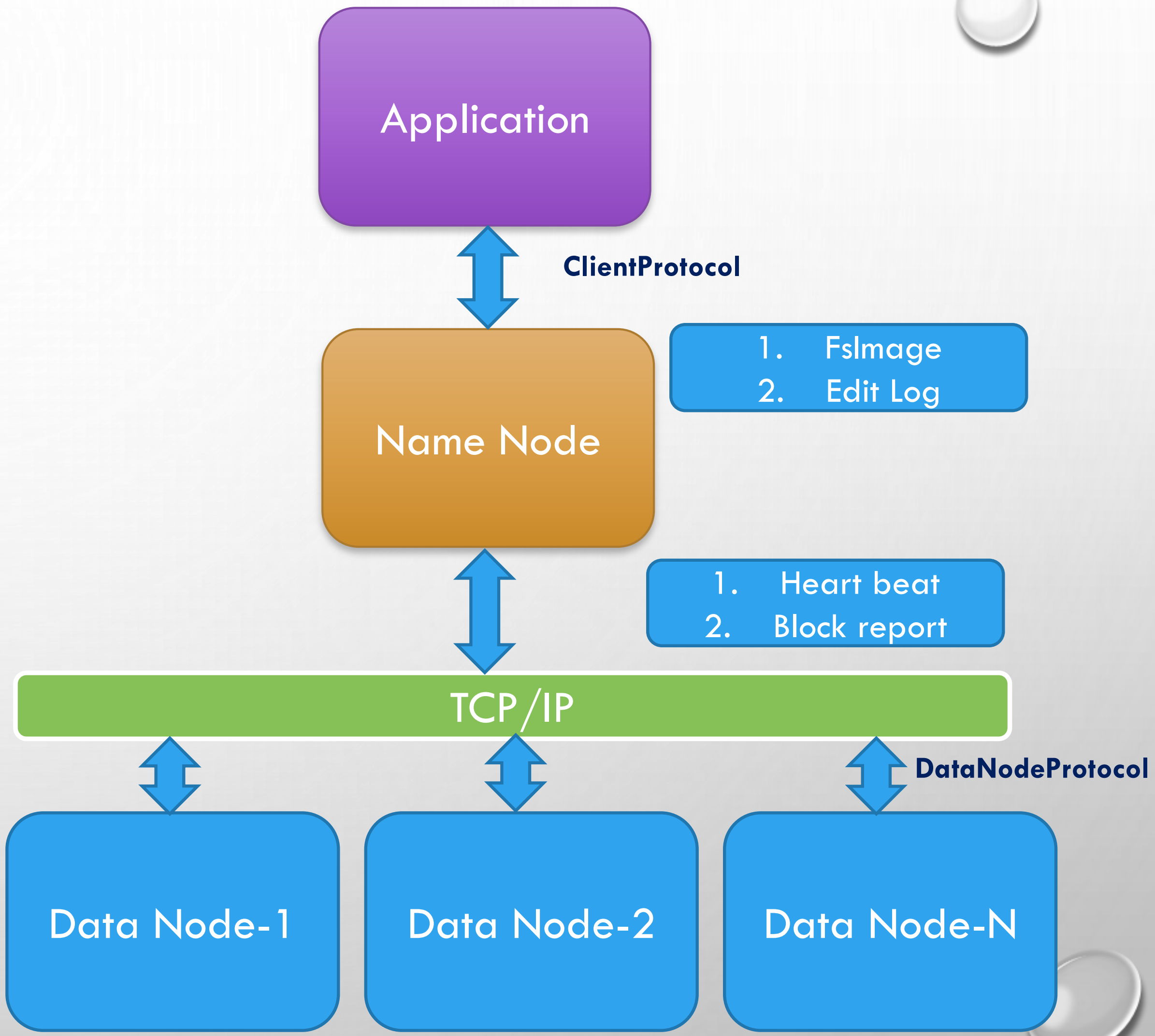
-HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients
-The NameNode executes file system namespace operations like opening, closing, and renaming files and directories.
-It also determines the mapping of blocks to DataNodes.
-NameNode stores the Metadata, this consists of fsimage and editlog.

Fsimage:
This contained serialized form of all directory and file in the file System.The FsImage is stored as a file in the NameNode's local file system.

Edit Log:
This is a transaction log, which logs every change in the file system.

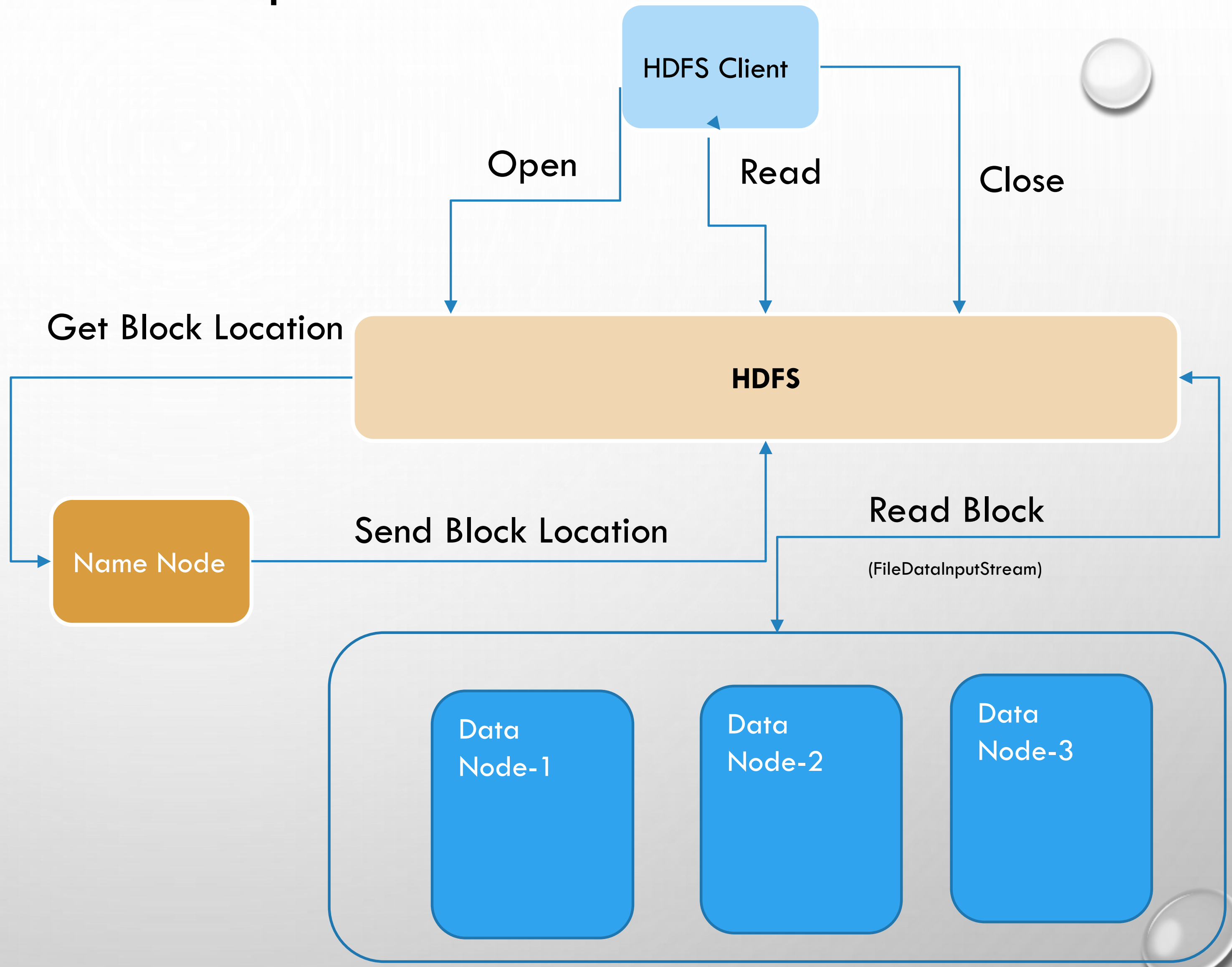
- ❑ **Data Nodes**

-There are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on
-A User file is split into one or more blocks and these blocks are stored in a set of DataNodes
-The DataNodes are responsible for serving read and write requests from the file system's clients.
-The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.



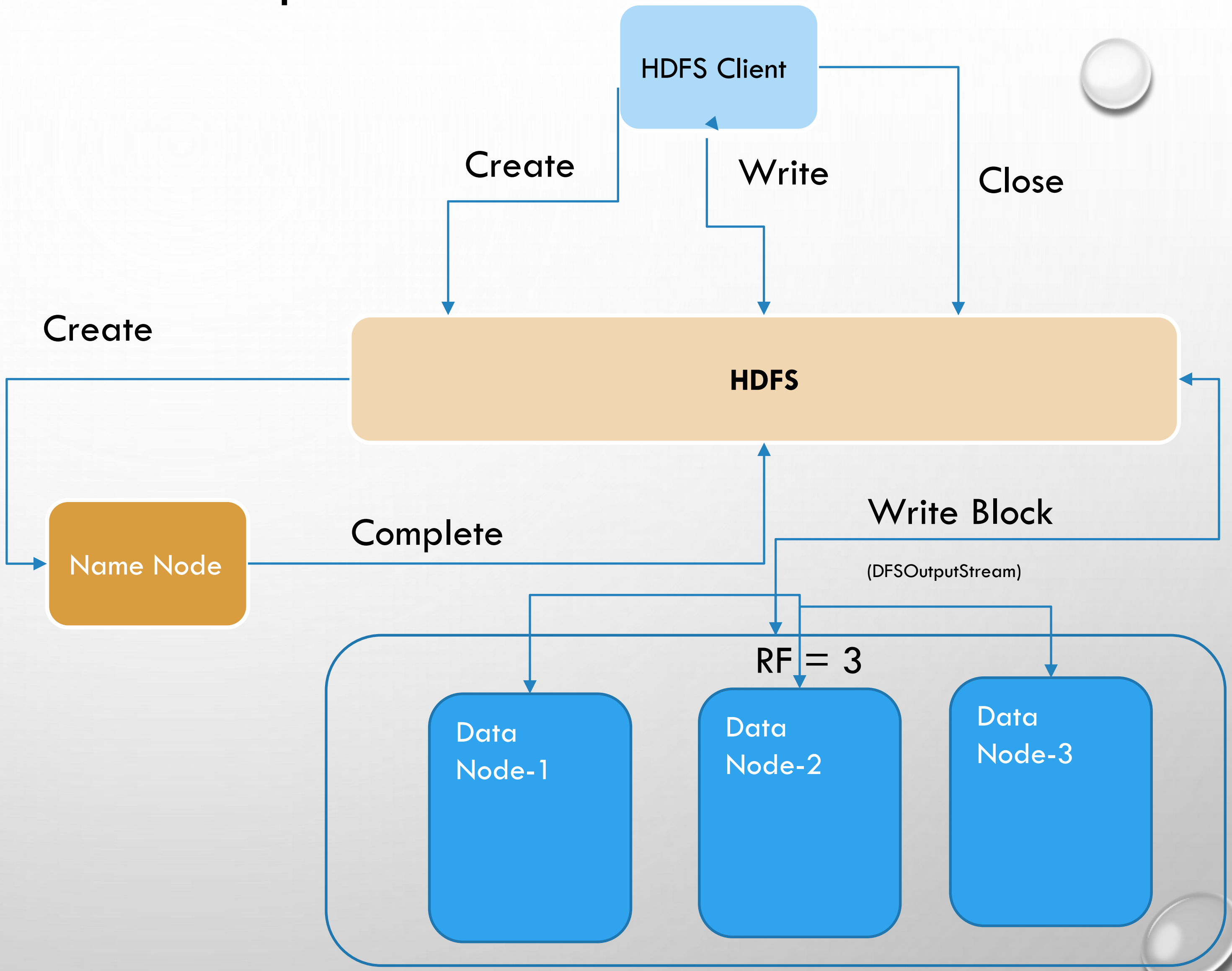
HDFS Architecture: Read Operation

- A read operation in general composition of many system call named Open, read and Close.
- The HDFS client start requesting Distributed File System to open the file and it started looking respective block information from Name Node.
- As soon as the information is retrieved, HDFS request block read operation to respective Data Nodes using FileDataInputStream.
- Finally it closes the steam of reding a file.



HDFS Architecture: Write Operation

- A client initiates write operation by calling 'create()' method of DistributedFileSystem object which creates a new file - Step no. 1 in the above diagram.
- DistributedFileSystem object connects to the NameNode using RPC call and initiates new file creation. However, this file creates operation does not associate any blocks with the file. It is the responsibility of NameNode to verify that the file (which is being created) does not exist already and a client has correct permissions to create a new file. If a file already exists or client does not have sufficient permission to create a new file, then IOException is thrown to the client. Otherwise, the operation succeeds and a new record for the file is created by the NameNode.
- Once a new record in NameNode is created, an object of type FSDataOutputStream is returned to the client. A client uses it to write data into the HDFS DataNode as per Replication Factor.



HDFS Architecture Contd.: HDFS Data Replication

- ❑ HDFS is designed to reliably store very large files across machines in a large cluster.
- ❑ **I have a 130MB file, Now how it can be distributed in a HDFS file system**

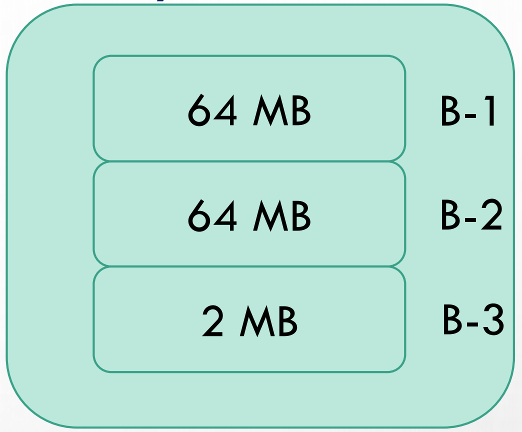
❖ **Name Node:**

-Will split the file based on block size
-Default block size is 64 MB
-All will be equal size block except the last one
-Distribute all data blocks to all Datanode
-Instruct for block replication based on replication factor configuration e.g. Consider the replication factor is 2.

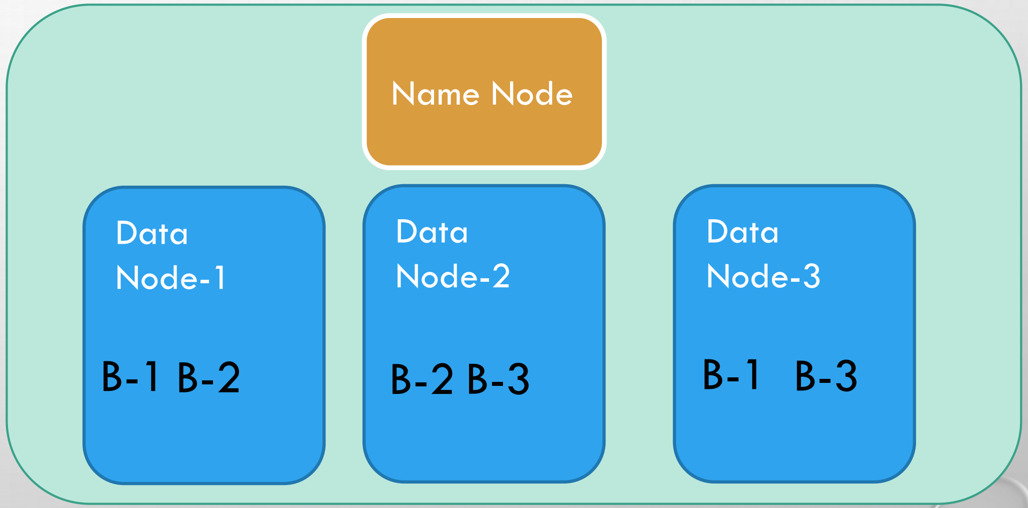
❖ **Data Node:**

-Write block to file system based on Name Node instruction

My TXT File



A Typical HDFS Cluster



HDFS Architecture Contd.: HDFS Sizing

Daily data input	**100 GB**	**Storage space used by daily data input = daily data input * replication factor = 300 GB**
HDFS replication factor	**3**	
Monthly growth	**5%**	Monthly volume = (300 * 30) + 5% = 9450 GB After one year = 9450 * (1 + 0.05)^12 = 16971 GB
Intermediate MapReduce data	**25%**	Dedicated space = HDD size * (1 - Non HDFS reserved space per disk / 100 + Intermediate MapReduce data / 100) = 4 * (1 - (0.25 + 0.30)) = 1.8 TB (which is the node capacity)
Non HDFS reserved space per disk	**30%**	
Size of a hard drive disk	**4 TB**	
Number of DataNodes needed to process: **Whole first month data = 9.450 / 1800 ~= 6 nodes** **The 12th month data = 16.971/ 1800 ~= 10 nodes** **Whole year data = 157.938 / 1800 ~= 88 nodes**		

HDFS Architecture Contd.: HDFS Sizing

NameNode memory	**2 GB - 4 GB**	**Memory amount = HDFS cluster management memory + NameNode memory + OS memory**
Secondary NameNode memory	2 GB - 4 GB	
OS memory	4 GB - 8 GB	
HDFS memory	2 GB - 8 GB	
At least NameNode (Secondary NameNode) memory = 2 + 2 + 4 = 8 GB		

DataNode process memory	**4 GB - 8 GB**	**Memory amount = Memory per CPU core * number of CPU's core + DataNode process memory + DataNode TaskTracker memory + OS memory**
DataNode TaskTracker memory	4 GB - 8 GB	
OS memory	4 GB - 8 GB	
CPU's core number	4+	
Memory per CPU core	4 GB - 8 GB	
At least DataNode memory = 4*4 + 4 + 4 + 4 = 28 GB		

- CPU: Multi-Core CPU with at lest four Cores per Physical CPU.
- Network: High throughput 10 GB ethernet intra Rack.

HDFS Architecture: HDFS Access & Limitations

- **HDFS Access**

Action	Command
Create a directory named `/foodir`	`bin/hadoop dfs -mkdir /foodir`
Remove a directory named `/foodir`	`bin/hadoop dfs -rmr /foodir`
View the contents of a file named `/foodir/myfile.txt`	`bin/hadoop dfs -cat /foodir/myfile.txt`

Action	Command
Put the cluster in Safemode	`bin/hadoop dfsadmin -safemode enter`
Generate a list of DataNodes	`bin/hadoop dfsadmin -report`
Recommission or decommission DataNode(s)	`bin/hadoop dfsadmin -refreshNodes`

- **HDFS Limitations**

-HDFS does not yet implement user quotas

-HDFS does not support hard links or soft links

-Write once Read multiple times

- Name node is still a single instance

MapReduce Architecture

- Hadoop **MapReduce** is the data processing layer of Hadoop. It processes large structured and unstructured data stored in HDFS. MapReduce also processes a huge amount of data in parallel. It does this by dividing the job (submitted job) into a set of independent tasks (sub-job). In Hadoop, MapReduce works by breaking the processing into phases: Map and Reduce.

  - **Map** – It is the first phase of processing, where we specify all the complex logic code.

  - **Reduce** – It is the second phase of processing. Here we specify light-weight processing like aggregation/summation.

MapReduce Architecture Contd.

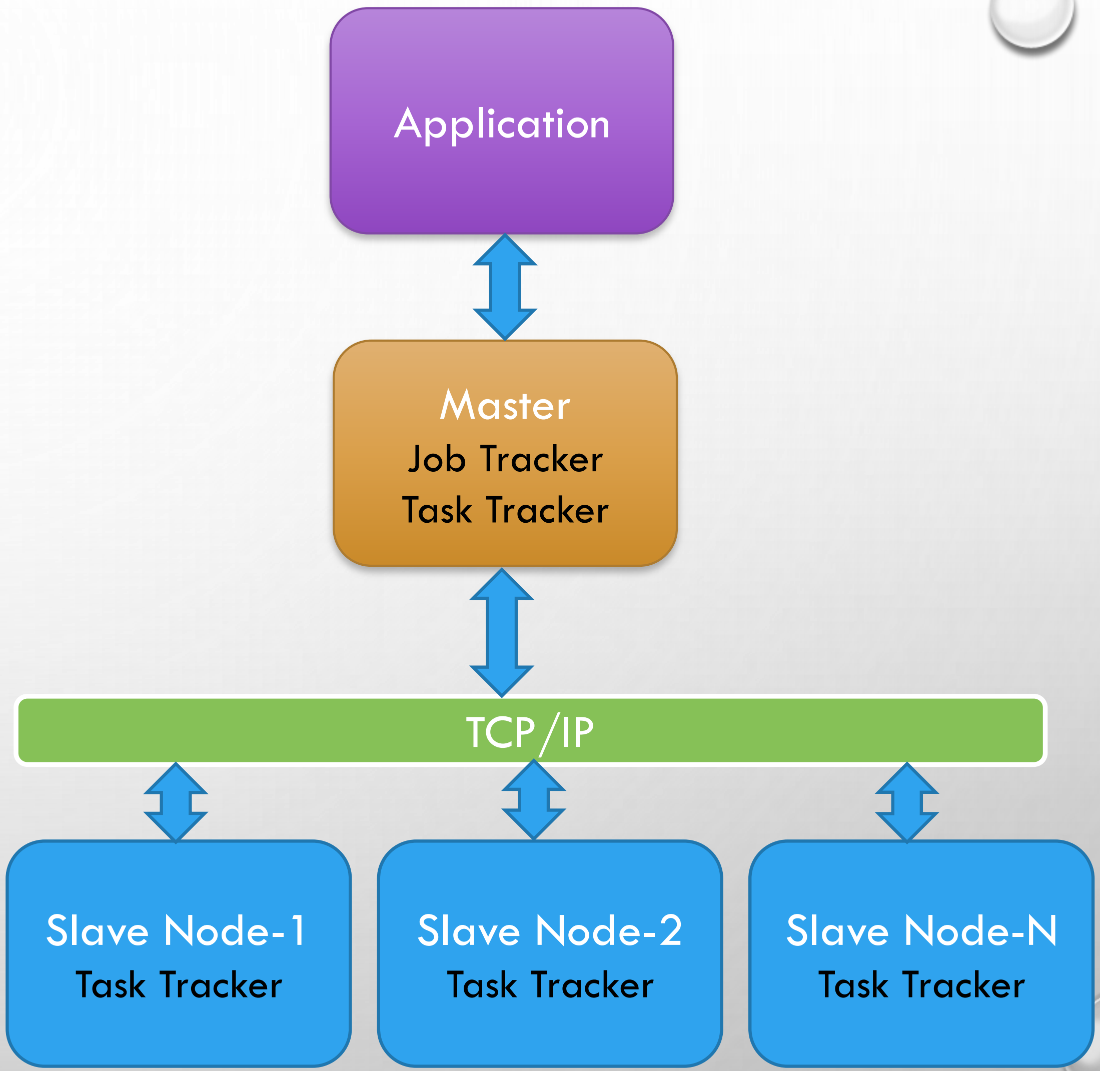
- **Master Node**

- The master node allows you to conduct parallel processing of data using Hadoop MapReduce.
-It assigns & manage the task to Slave node.

- **Slave Node**

-The slave nodes are the additional machines in the Hadoop cluster which allows you to store data to conduct complex calculations.

- All the slave node comes with Task Tracker and a DataNode. This allows you to synchronize the processes with the NameNode and Job Tracker respectively.



Working Procedure of MapReduce

a.txt

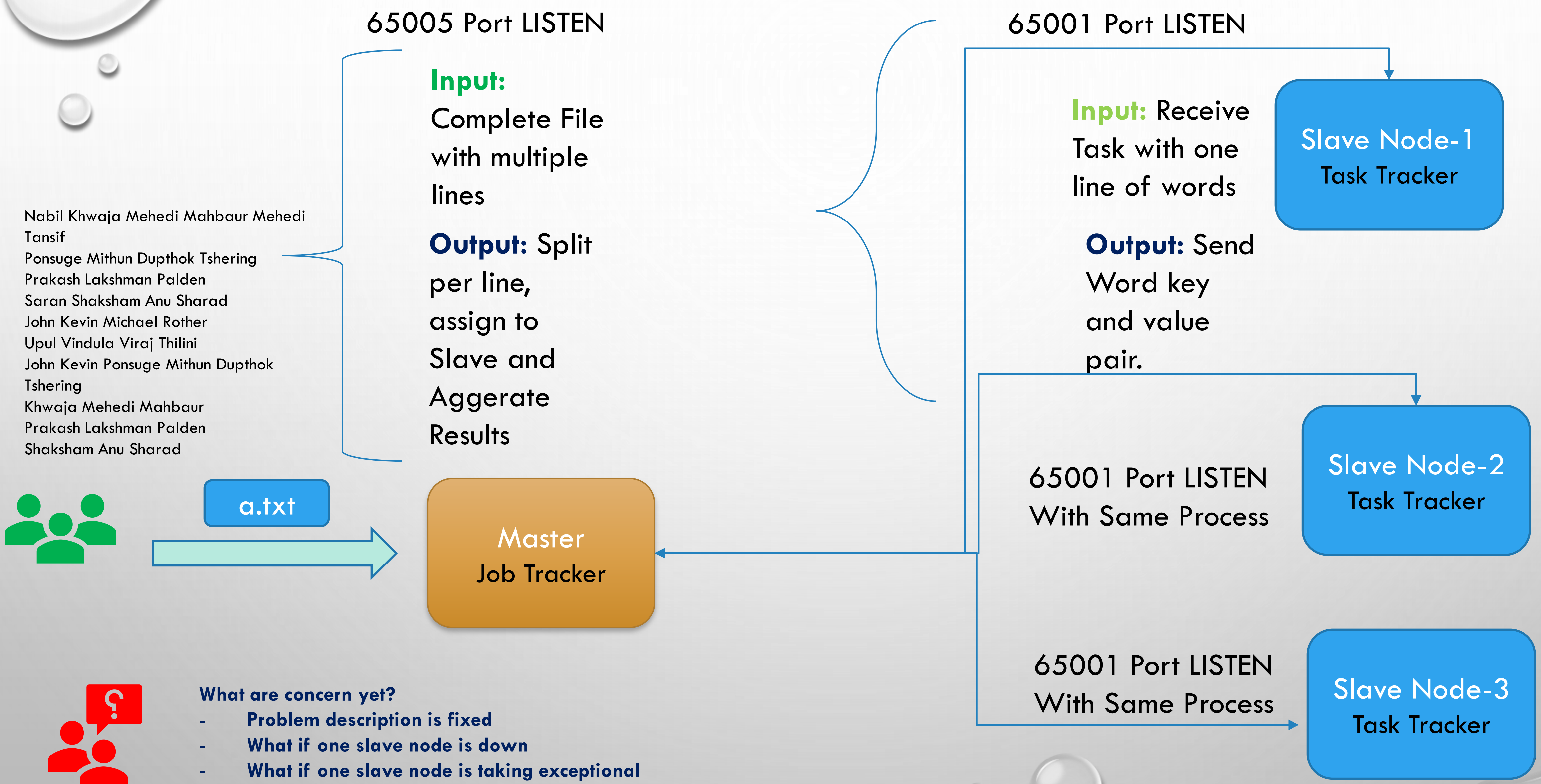
Nabil Khwaja Mehedi Mahbaur Mehedi Tansif
Ponsuge Mithun Dupthok Tshering
Prakash Lakshman Palden
Saran Shaksham Anu Sharad
John Kevin Michael Rother
Upul Vindula Viraj Thilini
John Kevin Ponsuge Mithun Dupthok Tshering
Khwaja Mehedi Mahbaur
Prakash Lakshman Palden
Shaksham Anu Sharad

Word Count Problem Traditional

1. (Nabil,1), (Khwaja,1) (Mehedi,1) (Mahbaur,1) (Mehedi,1) (Tansif,1)
2. (Ponsuge,1) (Mithun,1) (Dupthok,1) (Tshering,1)
3. (Prakash,1) (Lakshman,1) (Palden,1)
4. (Saran,1) (Shaksham,1) (Anu,1) (Sharad,1)
5. (John,1) (Kevin,1) (Michael,1) (Rother,1)
6. (Upul,1) (Vindula,1) (Viraj,1) (Thilini,1)
7. (John,2) (Kevin,2) (Ponsuge,2) (Mithun,2) (Dupthok,2) (Tshering,2)
8. (Khwaja,2) (Mehedi,2) (Mahbaur,2)
9. (Prakash,2) (Lakshman,2) (Palden,2)
10. (Shaksham,2) (Anu,2) (Sharad,2)

What happen if the file size is 100 GB?

Working Procedure of MapReduce Contd.


65005 Port LISTEN

Input: Complete File with multiple lines

Output: Split per line, assign to Slave and Aggerate Results

Nabil Khwaja Mehedi Mahbaur Mehedi Tansif
Ponsuge Mithun Dupthok Tshering
Prakash Lakshman Palden
Saran Shaksham Anu Sharad
John Kevin Michael Rother
Upul Vindula Viraj Thilini
John Kevin Ponsuge Mithun Dupthok Tshering
Khwaja Mehedi Mahbaur
Prakash Lakshman Palden
Shaksham Anu Sharad

65001 Port LISTEN

Input: Receive Task with one line of words

Output: Send Word key and value pair.

65001 Port LISTEN With Same Process

65001 Port LISTEN With Same Process

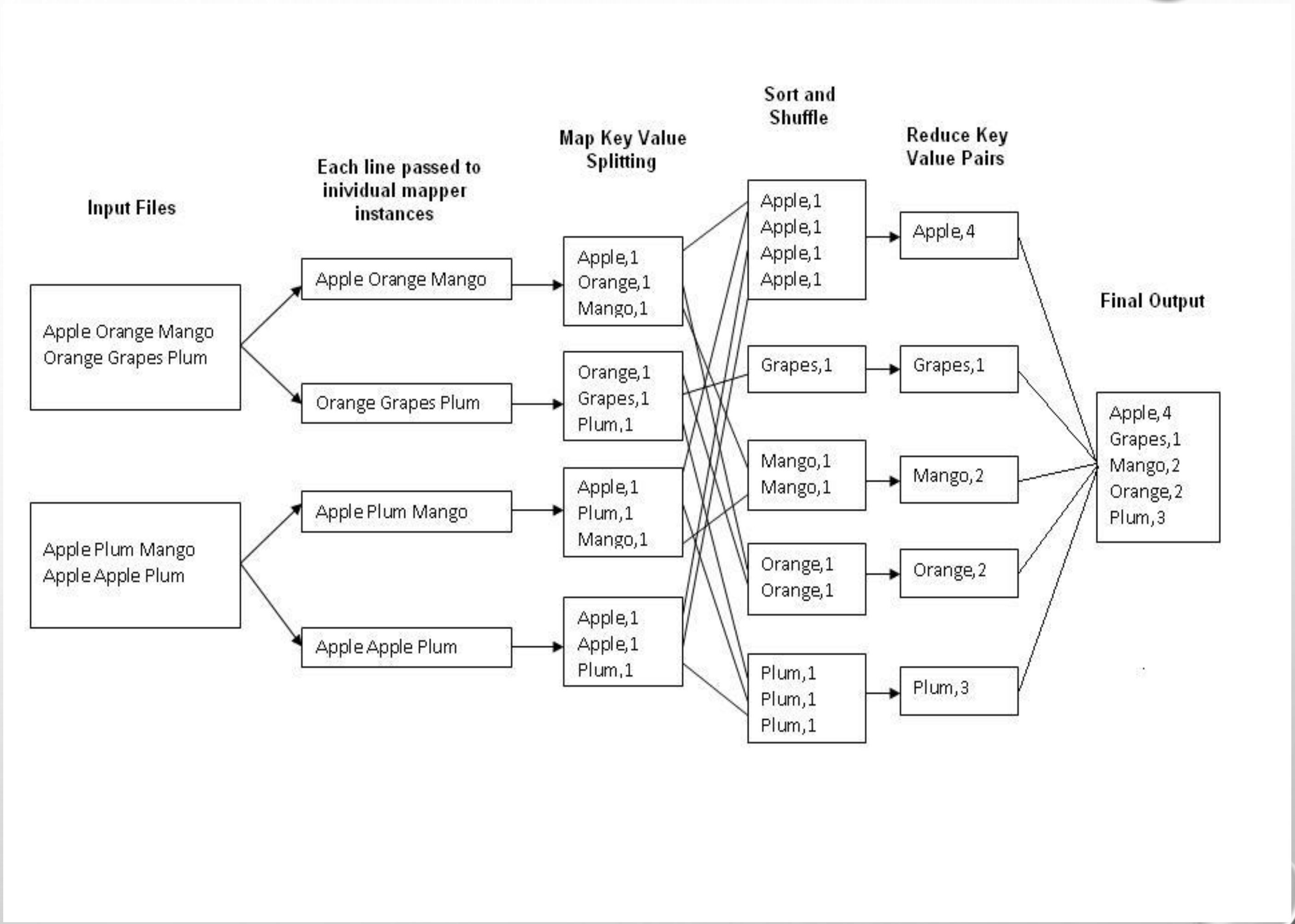
What are concern yet?

- **Problem description is fixed**
- **What if one slave node is down**
- **What if one slave node is taking exceptional time to process**

Working Procedure of MapReduce Contd.

- ❑ Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.
- ❑ A MapReduce *job* usually splits the input data-set into independent chunks which are processed by the *map tasks* in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the *reduce tasks*. Typically both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.
- ❑ Typically the compute nodes and the storage nodes are the same, that is, the MapReduce framework and the Hadoop Distributed File System are running on the same set of nodes. This configuration allows the framework to effectively schedule tasks on the nodes where data is already present, resulting in very high aggregate bandwidth across the cluster.

Understanding with an Example



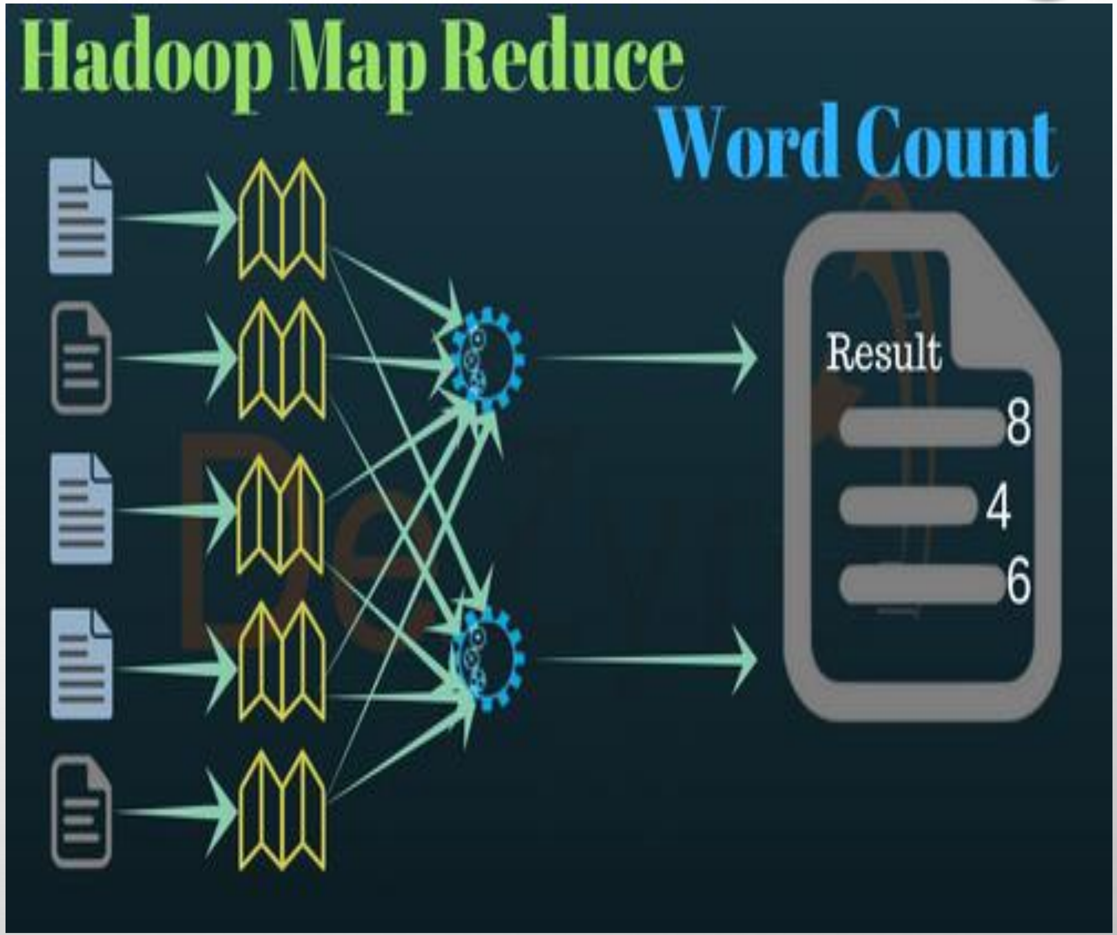
Working Procedure of MapReduce Contd.

- The MapReduce framework consists of a single master JobTracker and one slave TaskTracker per cluster-node. The master is responsible for scheduling the jobs' component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves execute the tasks as directed by the master.

- Minimally, applications specify the input/output locations and supply *map* and *reduce* functions via implementations of appropriate interfaces and/or abstract-classes. These, and other job parameters, comprise the *job configuration*. The Hadoop *job client* then submits the job (jar/executable etc.) and configuration to the JobTracker which then assumes the responsibility of distributing the software/configuration to the slaves, scheduling tasks and monitoring them, providing status and diagnostic information to the job-client.

- Although the Hadoop framework is implemented in Java^{TM}, MapReduce applications need not be written in Java always.

What if there is a Node Failure?



QUESTION & ANSWER

THANKS FOR ATTENDING THE CLASS & YOUR CO-OPERATION

References

- https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/
- http://elephantscale.com/
- https://www.hadoop.apache.org
- https://www.guru99.com/
- https://techvidvan.com/tutorials
- https://www.tutorialspoint.com/
- https://hadoop.apache.org/docs/r3.2.0/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html