University Questions on Hadoop

1. Describe the operations of shuffle and sort in Map Reduce Framework. Explain with the help of one example. [May16] [05 Marks]

2. What is MapReduce? Explain how Map and Reduce work? What is shuffling in MapReduce? [May17] [10Marks]

3. How big data problems are handled by Hadoop system? [May19][05Marks]

4. Explain how Hadoop goals are covered in hadoop distributed file system. [May19][10]

5. Describe the structure of HDFS in a Hadoop Ecosystem using a diagram [10 Marks]

what is Big Data?

Big data is a term that describes the large volume of data – both structured and unstructured – that inundates a business on a day-to-day basis.

But it’s not the amount of data that’s important. It’s what organizations do with the data that matters.

Big data can be analyzed for insights that lead to better decisions and strategic business moves.

Why is Big Data so Important?

You can take data from any source and analyze it to find answers that enable

1) Cost reductions,
2) Time reductions,
3) New product development and optimized offerings, and
4) Smart decision making.

When you combine big data with high-powered analytics, you can accomplish business-related tasks such as:

Determining root causes of failures, issues and defects in near-real time.

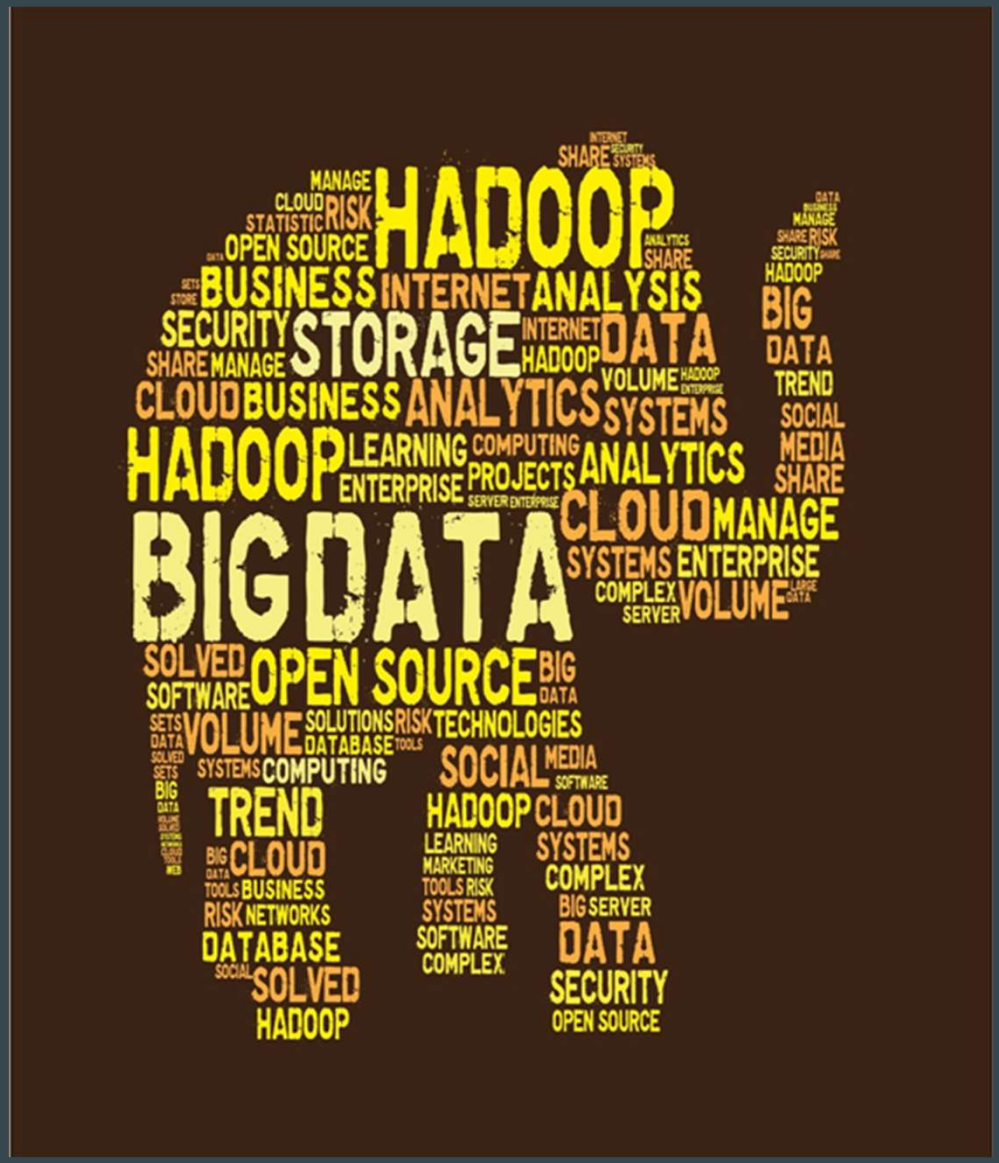
Generating coupons at the point of sale based on the customer's buying habits.

Recalculating entire risk portfolios in minutes.

Detecting fraudulent behavior before it affects your organization.

Hadoop

Hadoop is an open source, Java-based programming framework that supports the processing and storage of extremely large data sets in a distributed computing environment. It is part of the Apache project sponsored by the Apache Software Foundation.



Returns Results very fast

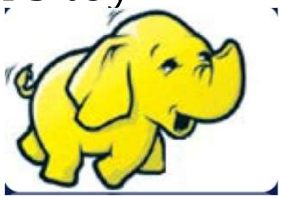
Distributes data across different machines
To process the tasks simultaneously.

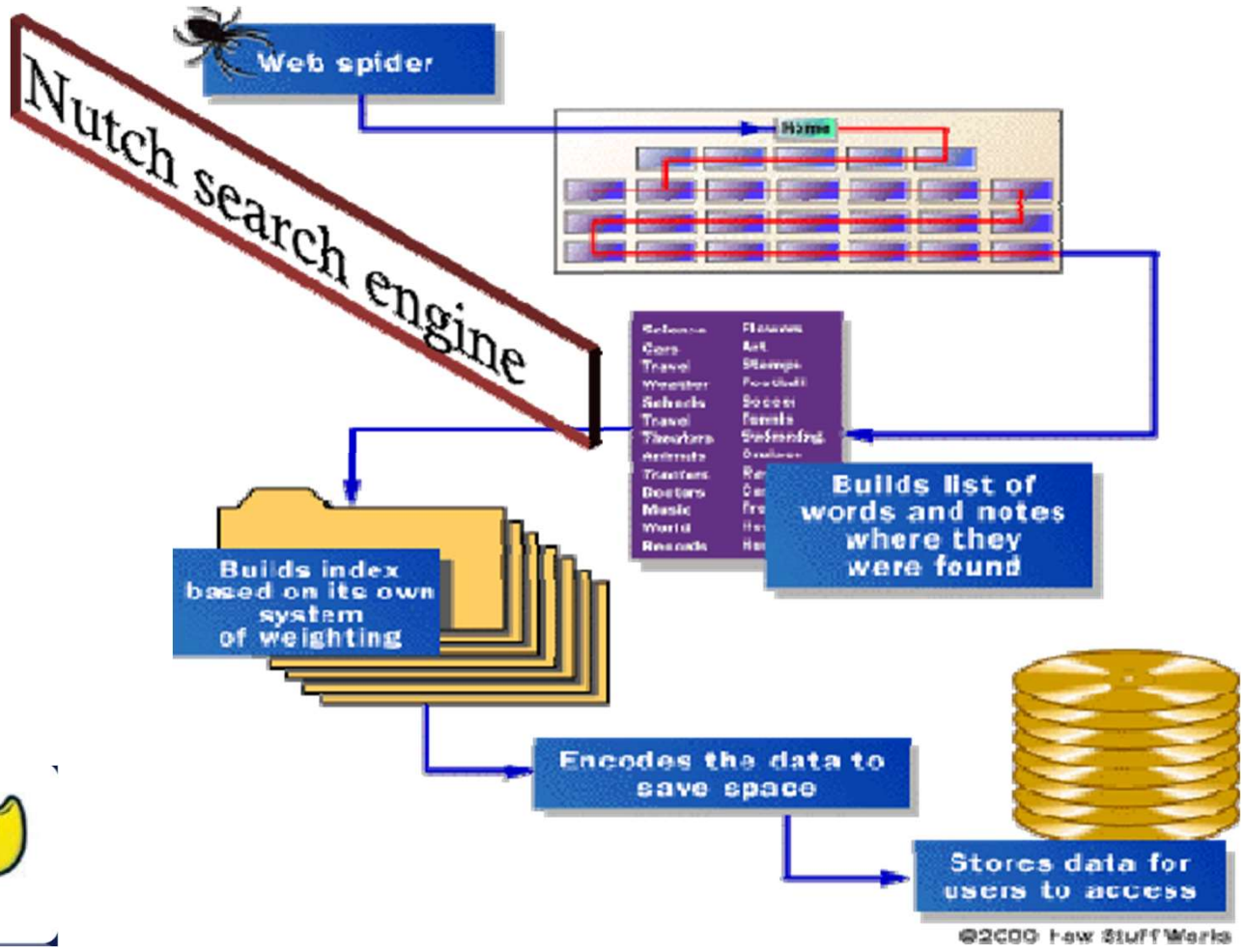
Doug Cutting and Mike Cafarella

Doug Joined Yahoo in 2006.

Crawler part : Nutch

Storage and Distributed Processing part : Hadoop (name of Doug's son's toy elephant.)





In 2008 **Yahoo** released Hadoop as an open source project.

Today it is a framework by non-profit organization (**ASF**)
Apache Software Foundation.

Hadoop is a framework that allows **storing** and **distributed processing** of ***big data*** across clusters of Commodity hardware using a single programming model.

Goals..

Scalable: It can scale up to 1 to thousands of machines.
Fault Tolerance It is designed to detect and handle failures at the application layer.
Economical: Based on commodity hardware.
Handle hardware failures: It is delivering a highly available service on the top of a cluster of computers.

Hadoop does the following tasks.

1. **Massive Data storage** on clusters of commodity hardware
2. **Fast processing**

Assumptions..

- **Huge data-** Large Data Set,
- **Hardware may fail,**
- **Portability Across Heterogeneous Hardware and Software Platforms**
- **Streaming Data Access-(Batch Processing)** so high throughput and low latency,
- **Simple Coherency Model-**data is write once and read multiple.
- **Moving Computation is Cheaper than Moving Data**
- **Supports tens of millions of files in a single instance.**

Hadoop framework includes following four main modules:

Hadoop Common: The necessary Java files and scripts required to start Hadoop.
Libraries and utilities needed by other hadoop modules.

Hadoop uses the Hadoop Common as a kernel to provide the framework's essential libraries.

Hadoop Yet Another Resource Negotiator YARN: *Resource Management Platform.*
This is a framework for job scheduling and cluster resource management.

Hadoop Distributed File System (HDFS™): A distributed file system that provides high-throughput access to application data. It has a limited interface for managing file system.

Hadoop MapReduce: This is YARN-based system for parallel processing of large data sets.

MapReduce is the key algorithm that the Hadoop MapReduce Engine uses to distribute work around a cluster. Software Framework that supports parallel processing

It provides the programming model used to tackle large distributed data processing -- mapping data and reducing it to a result.

Since 2012, the term "Hadoop" often refers not just to the base modules mentioned above but also to the collection of additional software packages that can be installed on top of or alongside Hadoop, such as **Apache Pig, Apache Hive, Apache HBase, Apache Spark etc.**

1. Hadoop Common Package :

Provides filesystem and OS level abstractions

.JAR files + Scripts Needed to start Hadoop.

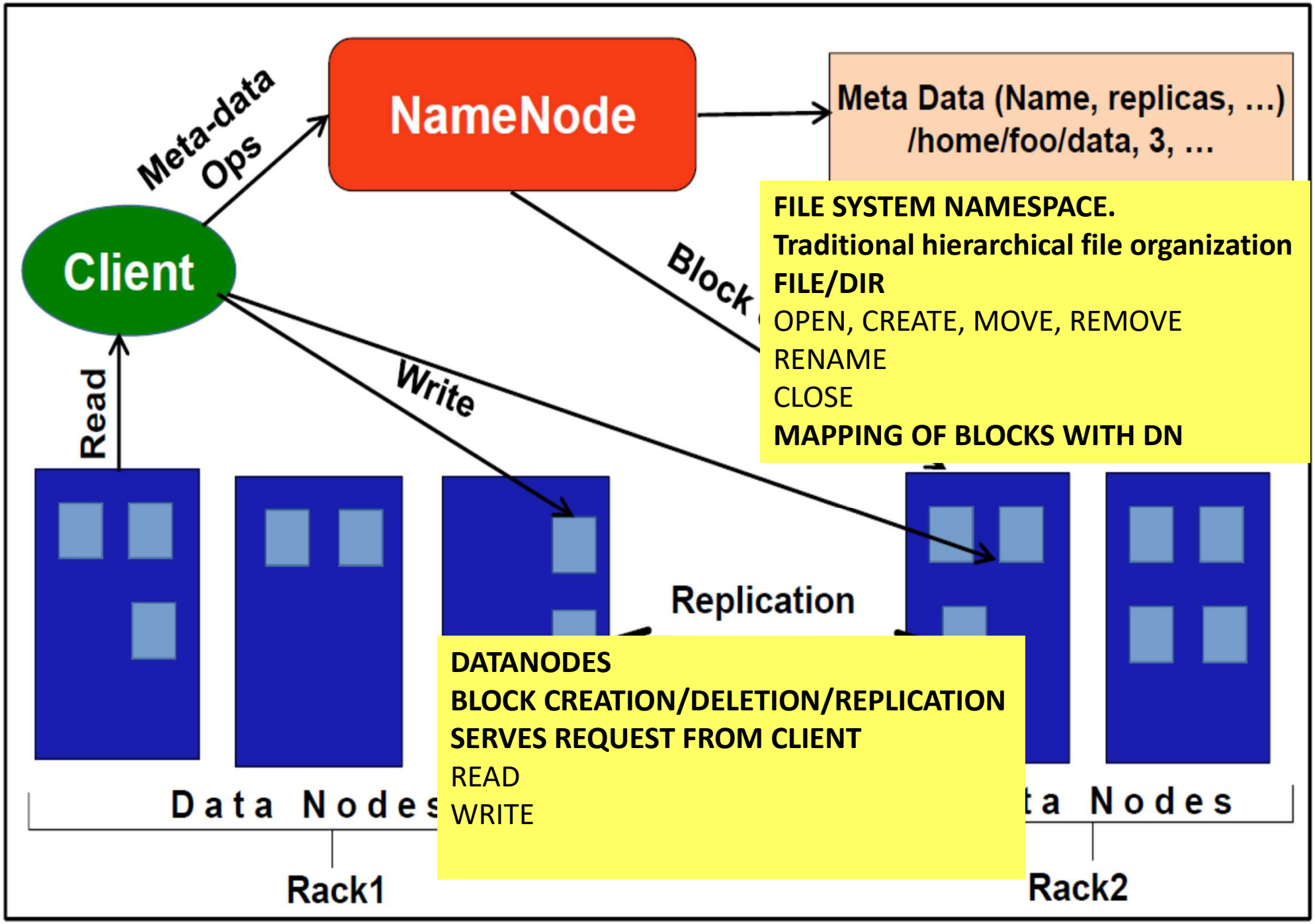
JRE 1.6 and Higher version in the machine : to start Hadoop.

SSH (Secure Shell) : Standard startup and shutdown **scripts** uses SSH to setup between the nodes in the cluster.

Java Libraries and Utilities required by other Hadoop modules like Hbase/Hive etc.

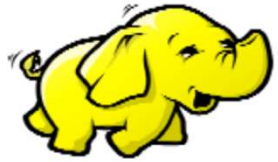
MASTER SLAVE ARCHITECTURE

HDFS ARCHITECTURE



FILE SYSTEM NAMESPACE.
Traditional hierarchical file organization
FILE/DIR
OPEN, CREATE, MOVE, REMOVE
RENAME
CLOSE
MAPPING OF BLOCKS WITH DN

DATANODES
BLOCK CREATION/DELETION/REPLICATION
SERVES REQUEST FROM CLIENT
READ
WRITE



HDFS ARCHITECTURE

NameNode

Meta Data (Name, replicas, …)
/home/foo/data, 3, …

Meta-data Ops

Client

Block ops

Read

Write

Replication

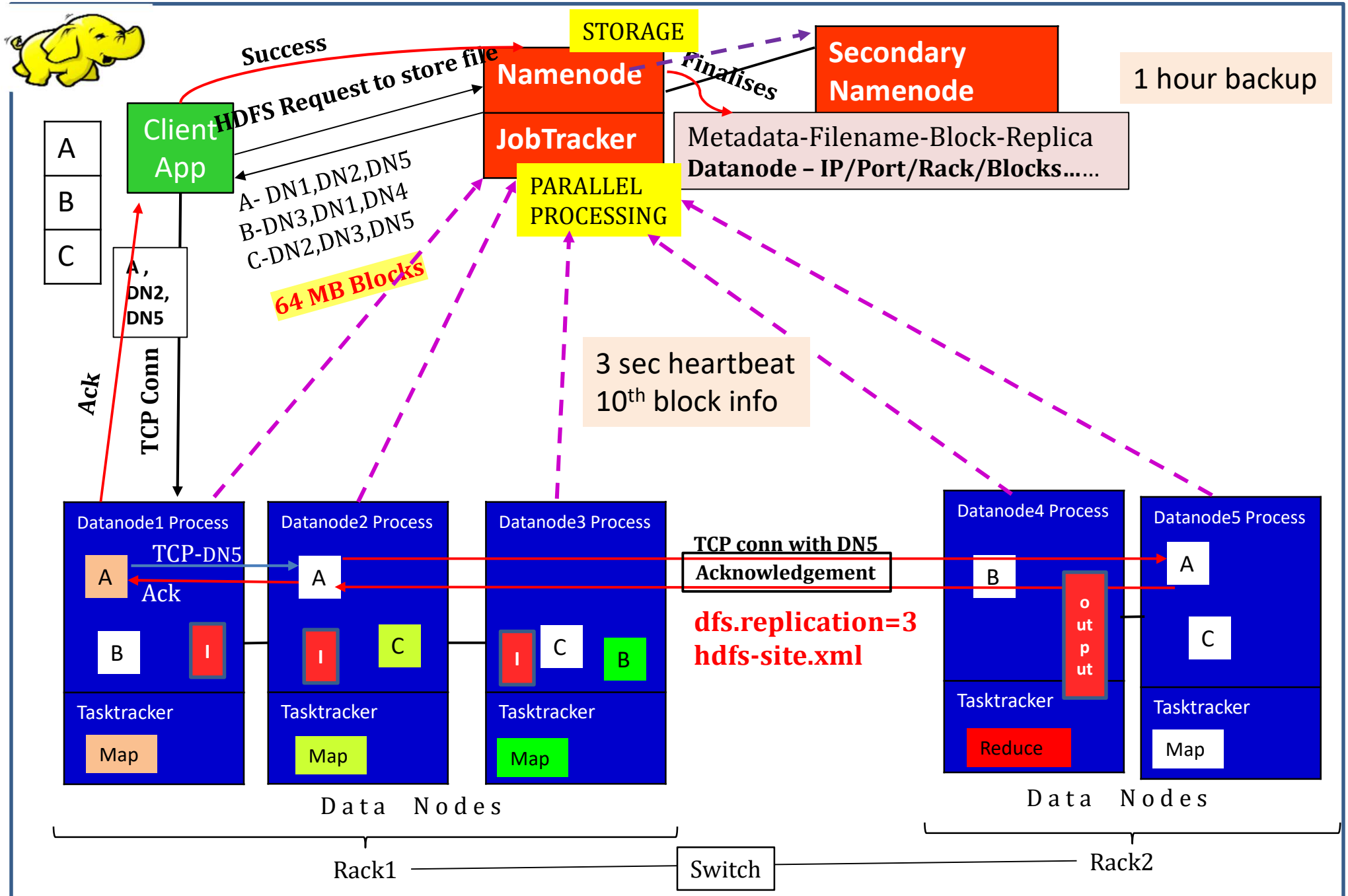
Blocks

Data Nodes

Rack1

Data Nodes

Rack2

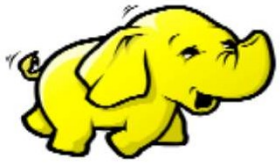


This process shows that Hadoop uses a lot of network bandwidth and storage.

Replication policy helps in proper traffic management and improves write performance.

Checksum is used to ensure data integrity.

Hadoop Target is cluster of 10000 nodes



HDFS ARCHITECTURE

Distributed File Design

File permissions and authentication

Node

Meta Data (Name, replicas, ...)
/home/foo/data, 3, ...

Highly Fault Tolerant

Client

Provides Command Line Interface

Block ops

Read

Write

Makes apps available for parallel processing

High Scalability-(Horizontal) and Throughput

64 MB Block

A

A

A

Distributed File Storage
Redundant storage

Replication

Blocks

A

Streaming Access to file data

Data Nodes

Data Nodes

16-64 DN in 1 Rack

Rack1

Runs on Commodity hardware

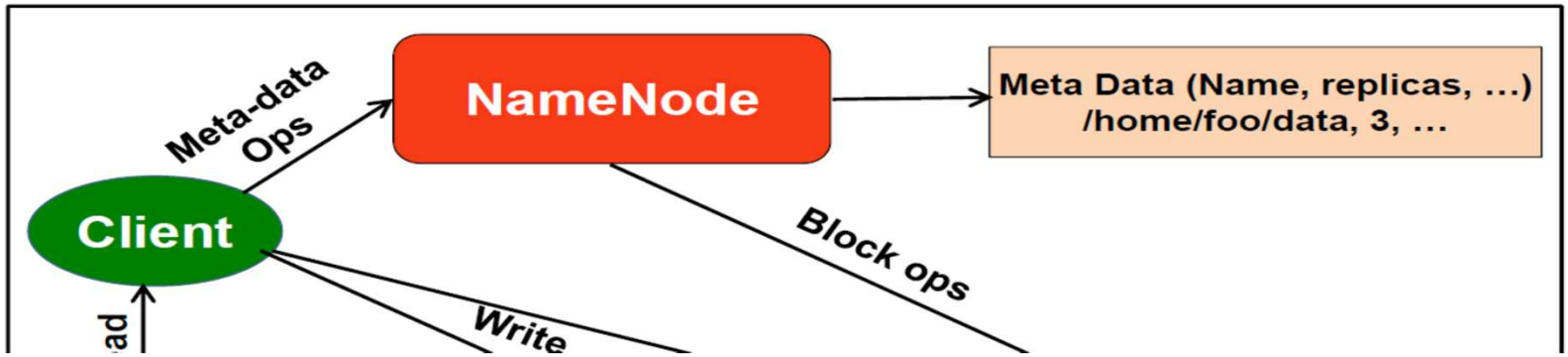
Rack2

2. Hadoop Distributed File System (HDFS)

- Distributed file system (**DFS)** design.
- Hadoop provides a **command interface** to interact with HDFS.
- It runs on **commodity hardware**. It is designed using low-cost hardware.
- HDFS is highly fault tolerant
- HDFS stores huge data across multiple machines. (**distributed storage**)
- These files are stored in **redundant fashion** to rescue the system from possible data losses in case of failure. Thus provides reliable and fast acccess.
- HDFS also makes applications available to **parallel processing.**
- The built-in servers of **namenode** and **datanode** help users to easily check the status of cluster.
- Streaming access to file system data.
- Provides Scalability and provides high throughput.
- HDFS provides file permissions and authentication.

HDFS ARCHITECTURE

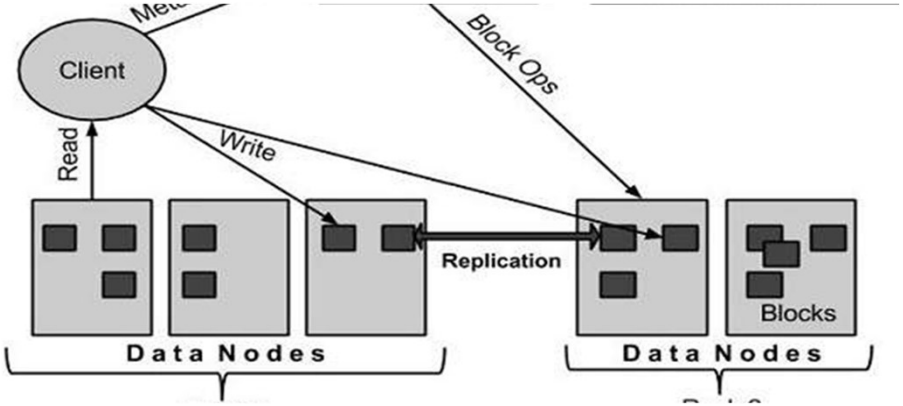
Master-slave architecture



Namenode (Master server) **it is also replicated.**

- Commodity hardware that contains the GNU/Linux OS and the namenode software.
- Manages the file system namespace.
- Maintains "inode" information
- Maps inode to block and locations
- Regulates client's access to files. (Authorization/Authentication)
- It also executes file system operations such as renaming, closing, and opening files and directories.
- Creates checkpoints and logs namespace changes.

Namenode maps datanode to the list of blocks, monitors the health of datablock and replicates missing blocks.



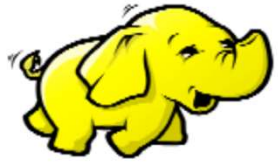
Datanodes (Slave nodes)

- is a commodity hardware having the **GNU/Linux OS and datanode software**.
  - For every node (Commodity hardware/System) in a cluster, there will be a datanode.

- Provide and manage **actual the data storage** of their system.
- Performs **read-write operations** on the file systems, as per client request.
- They also perform operations such as block creation, deletion, and replication according to the instructions of the namenode.
- Periodically send **heartbeats to the NameNode**.
- The default block size is 64MB, but it can be increased as per the need to change ..(128MB)
- HDFS provides a shell like any other file system and a list of commands are available to interact with the file system.

Goals of HDFS

Fault detection and recovery
Huge datasets
Computation at data



HDFS ARCHITECTURE

NameNode

Meta Data (Name, replicas, …)
/home/foo/data, 3, …

Meta-data Ops

Client

Block ops

Read

Write

Replication

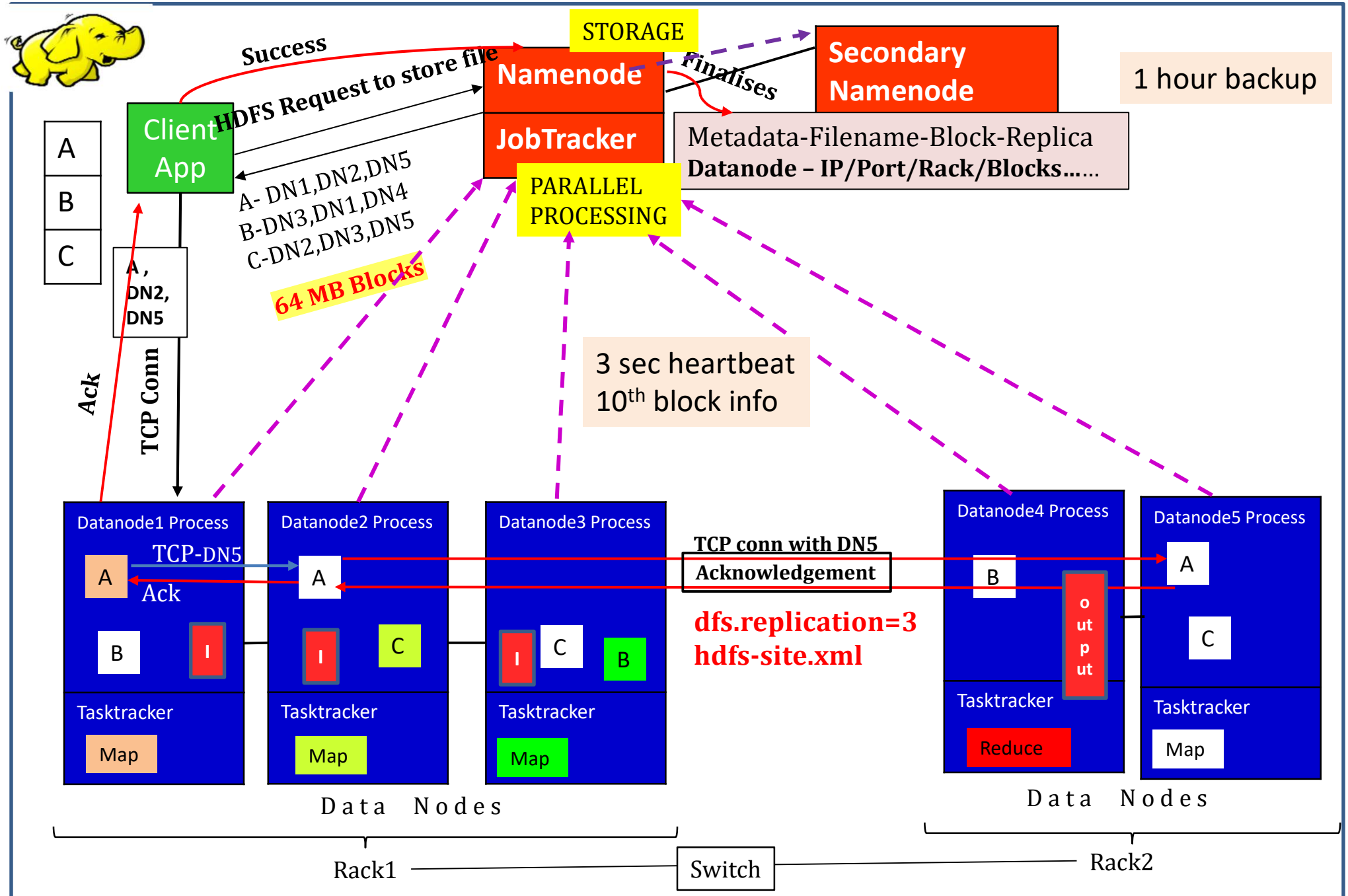
Blocks

Data Nodes

Rack1

Data Nodes

Rack2

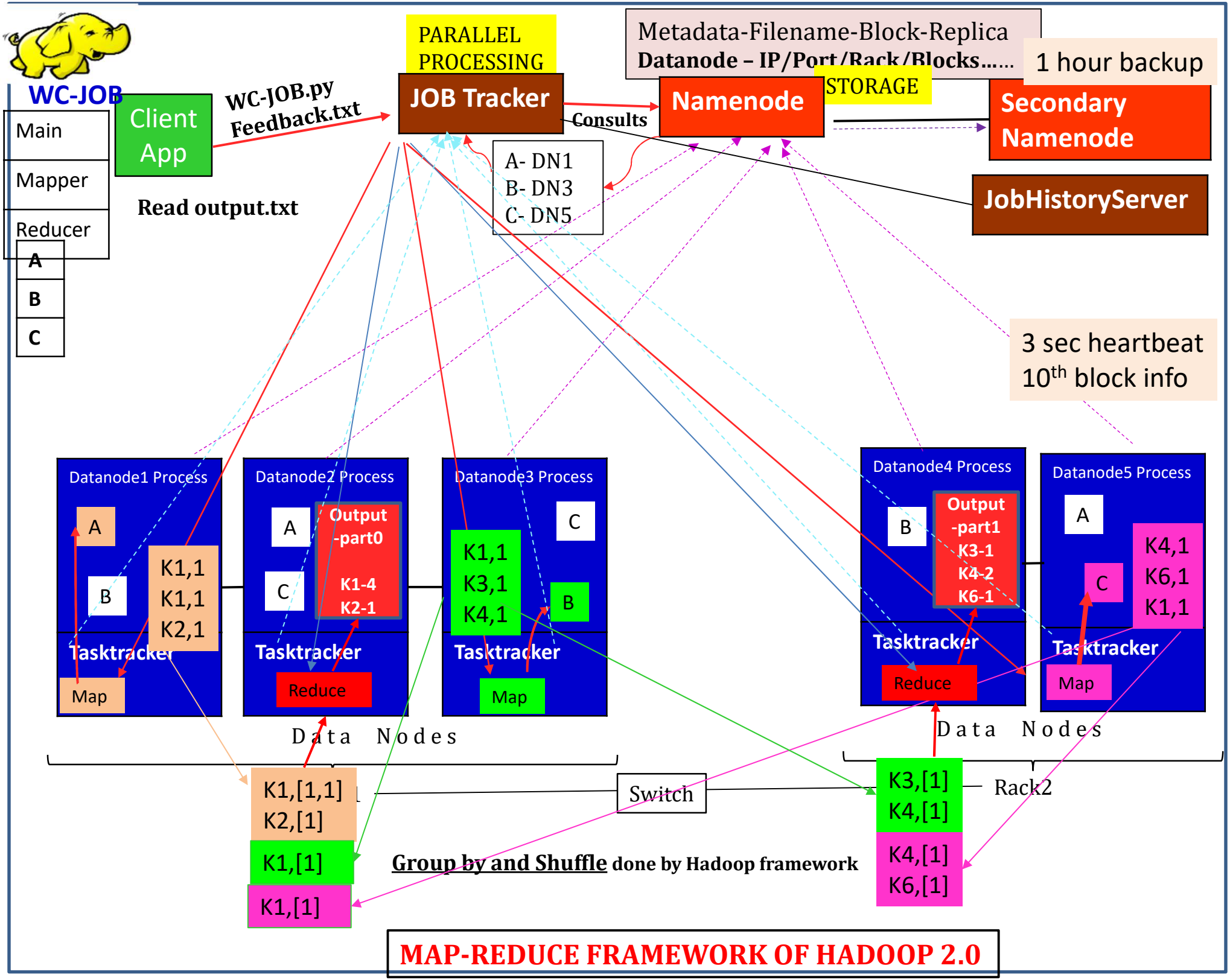


This process shows that Hadoop uses a lot of network bandwidth and storage.

Replication policy helps in proper traffic management and improves write performance.

Checksum is used to ensure data integrity.

Hadoop Target is cluster of 10000 nodes


WC-JOB
Main
Mapper
Reducer
A
B
C
Client App
WC-JOB.py
Feedback.txt
Read output.txt
PARALLEL PROCESSING
JOB Tracker
Consults
A- DN1
B- DN3
C- DN5
Metadata-Filename-Block-Replica
Datanode – IP/Port/Rack/Blocks……
STORAGE
Namenode
Secondary Namenode
JobHistoryServer
1 hour backup
3 sec heartbeat
10th block info
Datanode1 Process
A
B
K1,1
K1,1
K2,1
Tasktracker
Map
Datanode2 Process
A
C
Output -part0
K1-4
K2-1
Tasktracker
Reduce
Datanode3 Process
K1,1
K3,1
K4,1
C
B
Tasktracker
Map
Data Nodes
Datanode4 Process
B
Output -part1
K3-1
K4-2
K6-1
Tasktracker
Reduce
Datanode5 Process
A
C
K4,1
K6,1
K1,1
Tasktracker
Map
Data Nodes
K1,[1,1]
K2,[1]
K1,[1]
K1,[1]
Switch
K3,[1]
K4,[1]
Rack2
K4,[1]
K6,[1]
Group by and Shuffle done by Hadoop framework
MAP-REDUCE FRAMEWORK OF HADOOP 2.0

3. MapReduce

Hadoop **MapReduce** is a software framework.

Helps in writing applications.

Helps in parallel processing of Big Data on large clusters.

Basically comprises of three sequential processes/tasks- Map, Shuffle, Reduce

The MapReduce framework takes care of scheduling and monitoring of tasks.
It also re-executes the failed tasks with help of JobTracker.

The MapReduce framework consists of a single master **JobTracker** and one or many slave **TaskTracker** per cluster-node.

- **The Map Task:** This is the first task, which takes input data on which desired function is executed to convert it into a set of data, (intermediate key/value pairs). It is a parallel and share nothing processing of input.
- **Shuffle Task:** Shuffle phase is introduced and managed internally by the framework
- **The Reduce Task:** Input to Reduce is output of Map Task Grouped by key and Values **combined /aggregated** together as per Reduce algorithm provided by user..

Typically both the input and the output are stored in a file-system.

MapReduce

- At Cluster Level two processes **JobTracker** and **TaskTracker**
- **JobTracker** Process (runs on any one of the nodes in the cluster) is responsible for
  - scheduling the jobs component tasks on the slaves
  - resource management across clusters,
  - tracking resource consumption/availability and,
  - monitoring them and
  - re-executing the failed tasks.

- Each MapReduce job is split into a number of tasks that are assigned to various tasktrackers depending upon which data is stored on the node by JobTracker.

- The slaves **TaskTracker**(deployed on each machine in the cluster) execute the (map/reduce) tasks as directed by the master and provide task-status information to the master periodically.

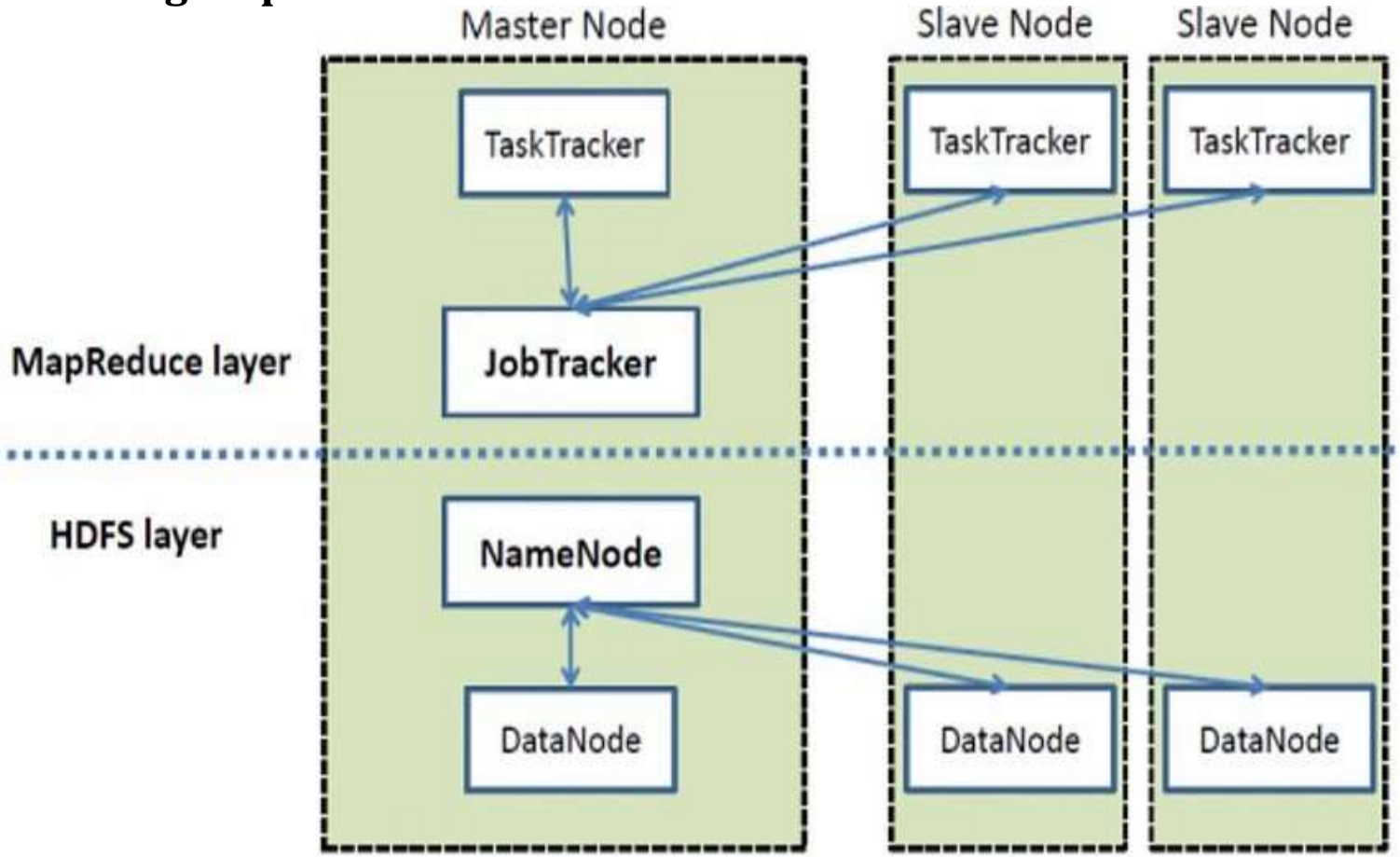
- The JobTracker is a single point of failure for the Hadoop which means if JobTracker goes down, all running jobs are halted.
- **JobHistoryServer :** It is a process that saves historical information about completed tasks/apps

If a task fails, the JobTracker will automatically relaunch the task, on a different node if necessary, up to a predefined limit of retries

Client Machine loads data and MapReduce program into the cluster and retrieves the results once the program is executed.

JobTracker coordinates and parallel processing of data using MapReduce.

If **reduce** task is executed on separate DN then intermediate results are sent over network for final computation.



NameNode coordinates and monitors HDFS-Data Storage functions.

HIGH LEVEL ARCHITECTURE OF HADOOP

Client communicates with NN to get the DN where results file. Blocks are read sequentially.

4. Hadoop YARN - YET ANOTHER RESOURCE NEGOTIATOR

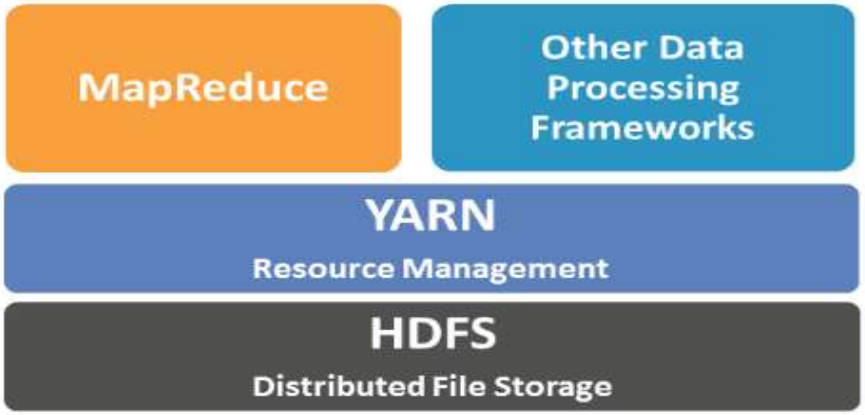
Hadoop v1.0

Ref: https://www.edureka.co/blog/hadoop-yarn-tutorial/

- JobTracker - Resource Allocation, Task scheduling and Monitoring
- Scalability Issues.
- Support - Custer of 5000 nodes and 40,000 tasks running concurrently.
- Computational resources utilization is inefficient in MRV1.

Hadoop v2.0

Optimised Cluster utilisation, Compatible with v1.0



- **Resource Management,**
- **Job scheduling and Monitoring**

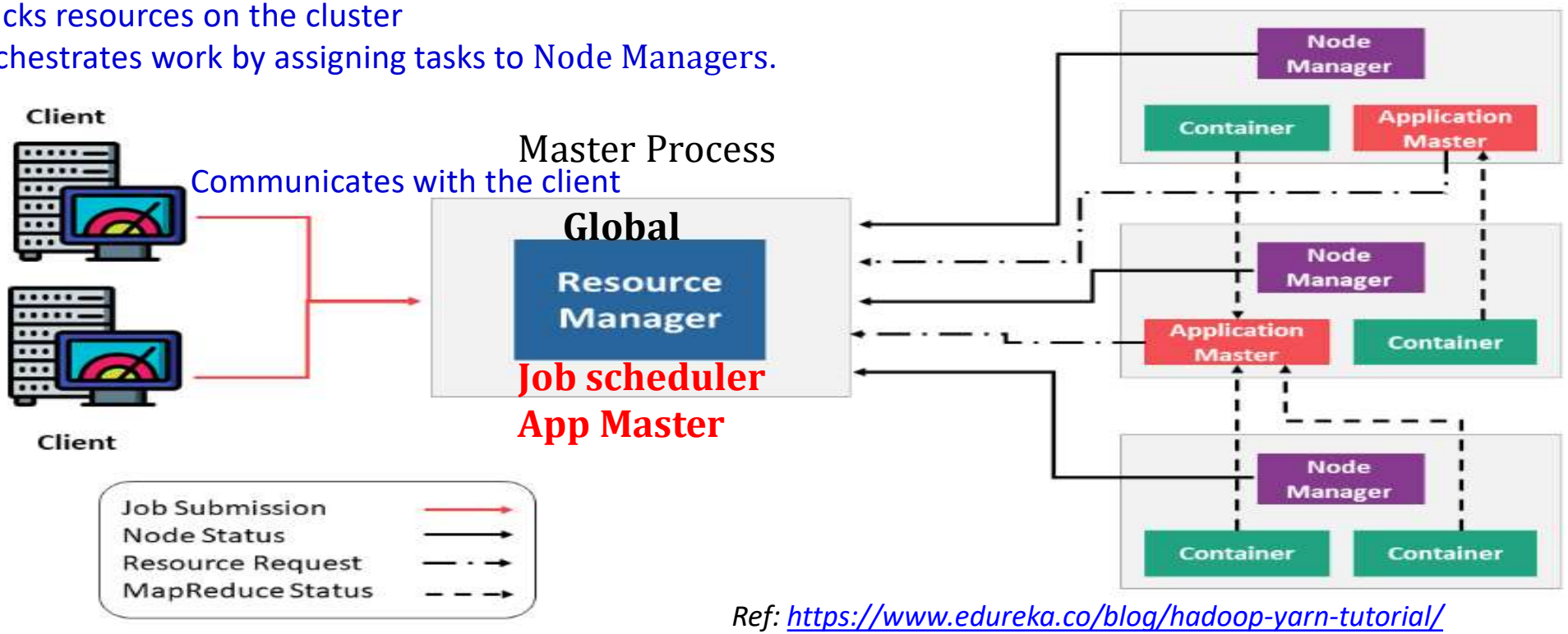
- different data processing methods (Multi-tenancy)
  - graph processing
  - interactive processing
  - stream processing

  as well as
  - batch processing to run and process data stored in HDFS.
- variety of tools like ***Spark*** for real-time processing, ***Hive*** for SQL, ***HBase*** for NoSQL and others.

4. Hadoop YARN - YET ANOTHER RESOURCE NEGOTIATOR

Global Resource Manager

1. Tracks resources on the cluster
2. Orchestrates work by assigning tasks to Node Managers.



Ref: https://www.edureka.co/blog/hadoop-yarn-tutorial/

Application Master per Application

- Negotiate appropriate resources from scheduler
- Track their status
- Monitor progress
- Works with node manager(s) to execute and monitor the component tasks

Node Manager

- is responsible for launching apps container and execution of task at datanode.
- Monitors apps resource usage (CPU/ RAM/ disk /network) at datanode, LOG mgt.
- Reports to RM

yarn-site.xml Present in each host of cluster at well known location.
It is used to configure the ResourceManager and NodeManager.
Two resources. **Vcore** : Usage share of CPU core. (For cpu intensive jobs > 1) and **RAM**

Steps of Workflow of Application in Hadoop YARN

An application is submitted by the client.

1.Application Manager is started by the allocation of the Container by the Resource Manager.

2.Resource Manager and **Application Manager** register with each other.

3.The **Application Manager** does the negotiation of the Container to the Resource Manager.

4.The **Node Manager** launches the **Container** after being notified by the Application Manager.

5.Execution of Application code is done in the Container.

6.The **Application Manager** or Resource Manager monitors the status of the application after being contacted by the client.

7.Un-Registration of Application Manager is done with Resource Manager after the process is complete.

Advantages of Hadoop

Hadoop framework allows the user to quickly write and test distributed systems. It is efficient, and it automatic distributes the data and work across the machines and in turn, utilizes the underlying **parallelism** of the CPU cores.

Hadoop does not rely on hardware to provide **fault-tolerance** and **high availability** (FTHA), rather Hadoop library itself has been designed to detect and handle failures at the application layer.

Scalable: Servers can be **added** or **removed** from the cluster dynamically and Hadoop continues to operate without interruption.

Economical: Another big advantage of Hadoop is that apart from being **open source**, it is compatible on all the platforms since it is Java based.

- **Huge data-** Large Data Set, high volume, variety and velocity
- **Portability Across Heterogeneous Hardware and Software Platforms**
- **Streaming Data Access-(Batch Processing)** so high throughput and low latency,
- **Moving Computation is Cheaper than Moving Data**
- **Supports tens of millions of files in a single instance.**
- **Reliable – checksum for a block, authentication, authorization**
- **Tools available**

Limitations

1. HDFS can not be directly mounted by an existing OS. Getting data in and out from existing OS is inconvenient.
2. **File Access** can be achieved through
    1. Native Java API through C++/Ruby/Python etc
    2. Command line interface
    3. HDFS-UI web app over HTTP

3. **Security Concerns:** Hadoop security model is disabled by default. It does not provide encryption at storage and network level. So Govt. agencies and others do not prefer to keep their data in hadoop framework.
4. **Vulnerable by nature:** Framework is entirely written in java. A language most widely used by cyber criminals.
5. **Not fit for small data:** Due to high capacity design, HDFS lacks the ability to efficiently support the random reading of small files. It is not recommended for organizations with small data.
6. **Potential stability issues:** Open-source platform that is created by many developers. While improvements are constantly made, it has stability issues. Need to make sure that organizations are running the stable version.
7. **General limitations:** Google mentioned that Hadoop may not be the single solution for big data.
8. Cloud Dataflow .. Google's solution, Apache Flume has the ability to improve the efficiency and reliability of data, Collection, Integration and aggregation of data