

# Apache Hadoop

# Historical Review of Big Data

1997, The problem of Big Data, NASA researchers, Michael Cox et and David Ellsworth's paper



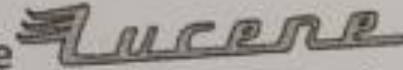
1998, Google was founded



1999, Apache Software Foundation (ASF) was established



2000, Doug Cutting launched his indexing search project: Lucene



2000, L Page and S. Brin wrote paper "the Anatomy of a Large-Scale Hyertextual Web search engine"

2001, The 3Vs, Doug Laney's paper "3D data management: controlling data Volume, Velocity & Variety" **Gartner.**

2002, Doug Cutting and Mike Caffarella started Nutch, a subproject of Lucene for crawling websites



2003, Sanjay Ghemawat et al. published "The Google File System" (GFS)

2003, Cutting and Caffarella adopted GFS idea and create Nutch Distribute File System (NDFS) later, it became HDFS

2004, Google Began to develop Big Table



2004, Yonik Seeley created Solr for Text-centric, read-dominant, document-oriented & flexible schema search engine

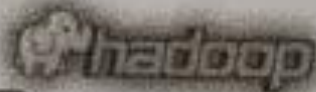


2004, Jeffrey Dean and Sanjay Ghemawat published "Simplified Data Processing on Large Cluster" or MapReduce

2005 Nutch established Nutch MapReduce

2005, Damien Katz created Apache CouchDB (Cluster Of Unreliable Commodity Hardware), former Lotus Notes

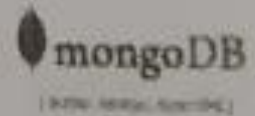
2006, Cutting and Cafarella started Hadoop or a subproject of Nutch



2006, Yahoo Research developed Apache Pig run on Hadoop



2007, 10gen, a start-up company worked on Platform as a Service (PaaS). Later, it became MongoDB



2008, Apache Hive (extend SQL), HBase (Manage data) and Cassandra (Schema free) to support Hadoop



2008 Hadoop became top level ASF project

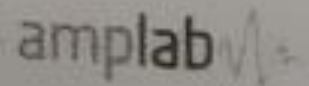
2008 TUB and HPI initiated Stratosphere Project and later become Apache Flink



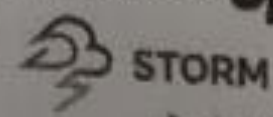
2010, Google licenced to ASF Hadoop



2010, Apache Spark, a cluster computing platform extends from MapReduce for in-memory primitives



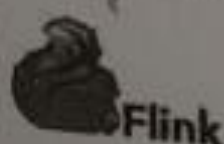
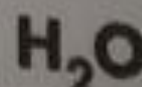
2011, Apache Storm was launched for a distributed computation framework for data stream



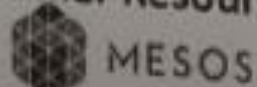
2012, Apache Drill for Schema-Free SQL Query Engine for Hadoop, NoSQL and cloud Storage



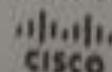
2012, Phase 3 of Hadoop - Emergence of "Yet Another Resource Negotiator" (YARN) or Hadoop 2



2013 Mesos became a top level Apache project



2014, Spark has > 465 contributors in 2014, the most active ASF project



2015, Enter Zeta Byte Era

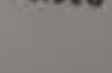
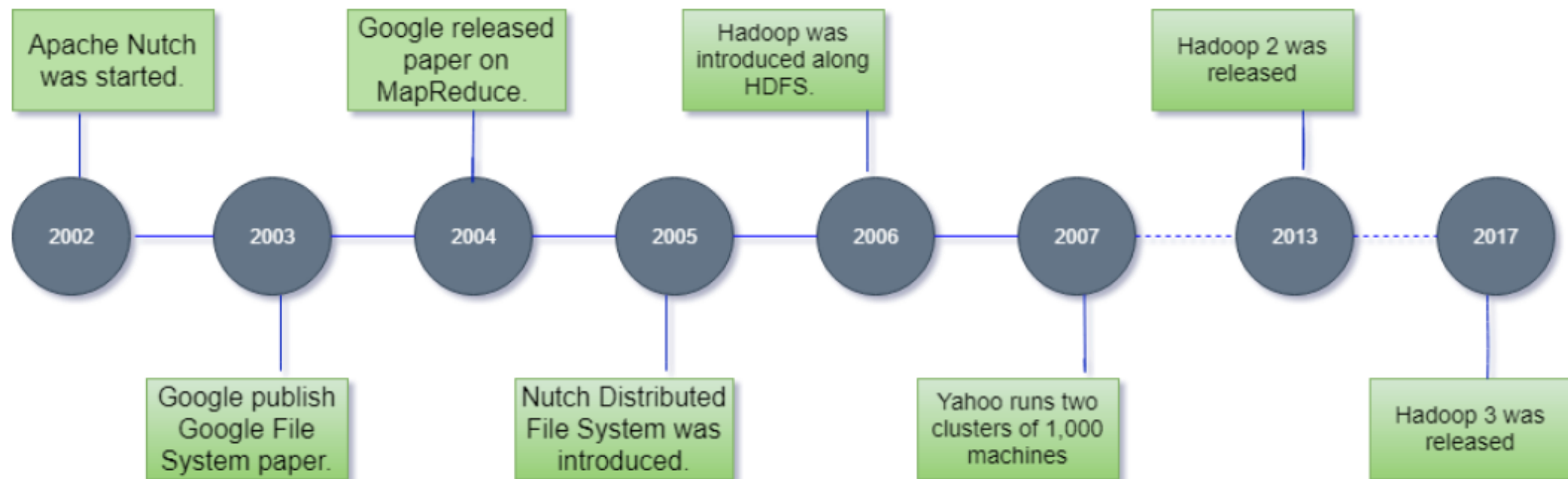


FIG. 1



Year	Event
2003	Google released the paper, Google File System (GFS).
2004	Google released a white paper on Map Reduce.
2006	<ul style="list-style-type: none"> <li>•Hadoop introduced.</li> <li>•Hadoop 0.1.0 released.</li> <li>•Yahoo deploys 300 machines and within this year reaches 600 machines.</li> </ul>
2007	<ul style="list-style-type: none"> <li>•Yahoo runs 2 clusters of 1000 machines.</li> <li>•Hadoop includes HBase.</li> </ul>
2008	<ul style="list-style-type: none"> <li>•YARN JIRA opened</li> <li>•Hadoop becomes the fastest system to sort 1 terabyte of data on a 900 node cluster within 209 seconds.</li> <li>•Yahoo clusters loaded with 10 terabytes per day.</li> <li>•Cloudera was founded as a Hadoop distributor.</li> </ul>
2009	<ul style="list-style-type: none"> <li>•Yahoo runs 17 clusters of 24,000 machines.</li> <li>•Hadoop becomes capable enough to sort a petabyte.</li> <li>•MapReduce and HDFS become separate subproject.</li> </ul>
2010	<ul style="list-style-type: none"> <li>•Hadoop added the support for Kerberos.</li> <li>•Hadoop operates 4,000 nodes with 40 petabytes.</li> <li>•Apache Hive and Pig released.</li> </ul>
2011	<ul style="list-style-type: none"> <li>•Apache Zookeeper released.</li> <li>•Yahoo has 42,000 Hadoop nodes and hundreds of petabytes of storage.</li> </ul>
2012	Apache Hadoop 1.0 version released.
2013	Apache Hadoop 2.2 version released.
2014	Apache Hadoop 2.6 version released.
2015	Apache Hadoop 2.7 version released.
2017	Apache Hadoop 3.0 version released.
2018	Apache Hadoop 3.1 version released.

# GFS

- Google File System (GFS) is a scalable distributed file system (DFS) created by Google Inc. and developed to accommodate Google's expanding data processing requirements. GFS provides fault tolerance, reliability, scalability, availability and performance to large networks and connected nodes. GFS is made up of several storage systems built from low-cost commodity hardware components. It is optimized to accommodate Google's different data use and storage needs, such as its search engine, which generates huge amounts of data that must be stored.

# Apache Hadoop

- Apache Hadoop is an open source software framework for storage and large scale processing of data-sets on clusters of commodity hardware.
- Hadoop was created by Doug Cutting and Mike Cafarella in 2005.

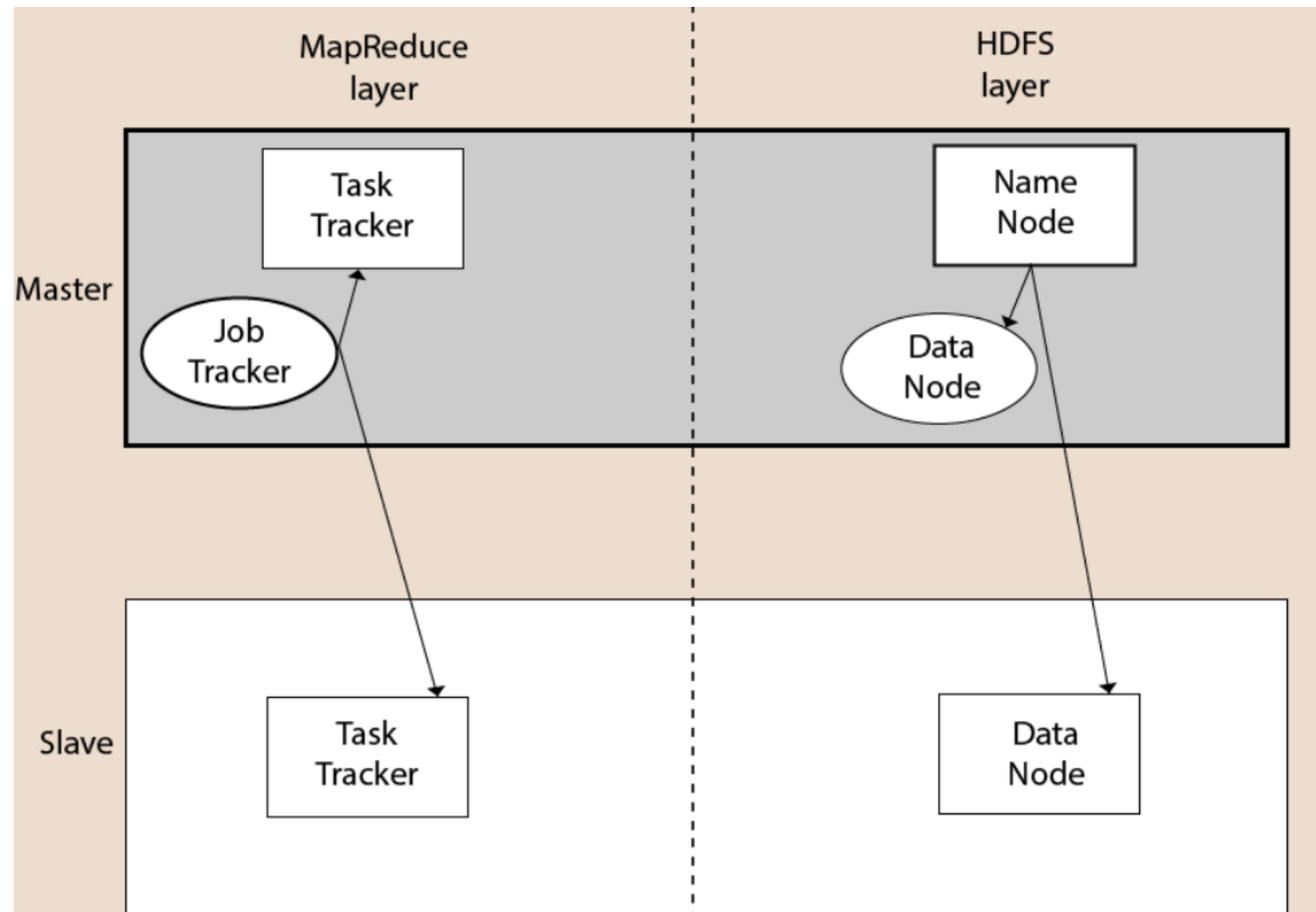




- Doug Cutting , who was working at Yahoo! at the time
- He named the project after his son's toy elephant.
- Cutting's son was 2 years old at the time and just beginning to talk. He called his beloved stuffed yellow elephant "Hadoop"



# Hadoop Architecture



The Hadoop architecture is a package of the file system, MapReduce engine and the HDFS (Hadoop Distributed File System). The MapReduce engine can be MapReduce/MR1 or YARN /MR2.

A Hadoop cluster consists of a single master and multiple slave nodes. The master node includes Job Tracker, Task Tracker, NameNode, and DataNode whereas the slave node includes DataNode and TaskTracker.

# The Apache Hadoop framework modules

- Hadoop Common: contains libraries and utilities needed by other Hadoop modules
- Hadoop Distributed File System (HDFS): a distributed file-system that stores data on the commodity machines, providing very high aggregate bandwidth across the cluster
- Hadoop YARN: a resource-management platform responsible for managing compute resources in clusters and using them for scheduling of users' applications
- Hadoop MapReduce: a programming model for large scale data processing

# Hadoop Distributed file system

- HDFS is a filesystem designed for storing very large files with streaming data access patterns, running on clusters of commodity hardware.
- The Hadoop Distributed File System (HDFS) is a distributed file system for Hadoop. It contains a master/slave architecture. This architecture consist of a single NameNode performs the role of master, and multiple DataNodes performs the role of a slave.
- Both NameNode and DataNode are capable enough to run on commodity machines. The Java language is used to develop HDFS. So any machine that supports Java language can easily run the NameNode and DataNode software.

- HDFS is not good fit for following
  - Low-latency data access
  - Lots of small files
  - Multiple writers and arbitrary file modifications

# NameNode

- It is a single master server exist in the HDFS cluster.
- As it is a single node, it may become the reason of single point failure.
- It manages the file system namespace by executing an operation like the opening, renaming and closing the files.
- It simplifies the architecture of the system.

# DataNode

- The HDFS cluster contains multiple DataNodes.
- Each DataNode contains multiple data blocks.
- These data blocks are used to store data.
- It is the responsibility of DataNode to read and write requests from the file system's clients.
- It performs block creation, deletion, and replication upon instruction from the NameNode.

# Job Tracker

- The role of Job Tracker is to accept the MapReduce jobs from client and process the data by using NameNode.
- In response, NameNode provides metadata to Job Tracker.

## Task Tracker

- It works as a slave node for Job Tracker.
- It receives task and code from Job Tracker and applies that code on the file.  
This process can also be called as a Mapper.



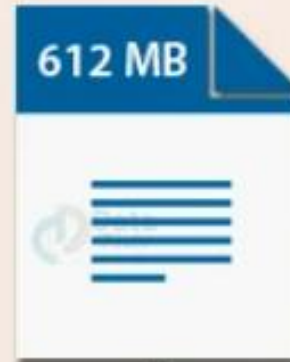
# HDFS concepts

- Block size
- HDFS block size
- Difference between HDFS and other file system
- Why is a block size in HDFS so large?

# Block size

- Hadoop is known for its reliable storage. Hadoop HDFS can store data of any size and format.
- HDFS in Hadoop divides the file into small size blocks called **data blocks**. These data blocks serve many advantages to the Hadoop HDFS.
- Files in HDFS are broken into **block-sized chunks** called **data blocks**. These blocks are stored as independent units.
- The size of these HDFS data blocks is **128 MB** by default. We can configure the block size as per our requirement by changing the **dfs.block.size** property in **hdfs-site.xml**
- Hadoop distributes these blocks on different slave machines, and the master machine stores the metadata about blocks location.
- All the blocks of a file are of the same size except the last one (if the file size is not a multiple of 128)

# Data Blocks in Hadoop HDFS



*Example.txt*



- Suppose we have a file of size **612 MB**, and we are using the default block configuration (128 MB). Therefore **five** blocks are created, the first four blocks are 128 MB in size, and the fifth block is 100 MB in size ( $128 \times 4 + 100 = 612$ ).

# Why are blocks in HDFS huge?

- **The default size of the HDFS data block is 128 MB.** The reasons for the large size of blocks are:
  1. To minimize the cost of seek: For the large size blocks, time taken to transfer the data from disk can be longer as compared to the time taken to start the block. This results in the transfer of multiple blocks at the disk transfer rate.
  2. If blocks are small, there will be too many blocks in [Hadoop](#) HDFS and thus too much metadata to store. Managing such a huge number of blocks and metadata will create overhead and lead to traffic in a network.

# Advantages of Hadoop Data Blocks

- No limitation on the file size: A file can be larger than any single disk in the network.
- Simplicity of storage subsystem
- Fit well with replication for providing Fault Tolerance and High Availability
- Eliminating metadata concerns: Since blocks are just chunks of data to be stored, we don't need to store file metadata (such as permission information) with the blocks, another system can handle metadata separately.

## Rack Awareness

**Rack** : is the collection of machines which are physically located in a single data-center connected through traditional network design and top of rack switching mechanism.

In Hadoop, **Rack** is a physical collection of **Nodes** (slave machines) put together at a single location for data storage.



## Rack Awareness

**Rack Awareness** : the concept that chooses closer DataNodes while storing the data blocks based on the rack information, maintained by NameNode in the form of Rack-Id's, is called Rack Awareness in Hadoop

**Rack awareness** is having the knowledge of Cluster topology or more specifically how the different data nodes are distributed across the racks of a Hadoop cluster.





## Rack Awareness

- Placement of replica is critical for ensuring high reliability and Fault-tolerance of HDFS.
- Optimizing replica placement via rack awareness is a distinguished feature of HDFS.
- Replication in multiple racks in HDFS is done using a policy as follows:
- **Replica Placement Policy** : “No more than one replica is placed on one node. And no more than two replicas are placed on the same rack.”

When the client is ready to load a file into the cluster, the content of the file will be divided into blocks(each Block size 128 MB)

Block A : 

Block B : 

Block C : 

Then client consults the Name node and gets the address of data nodes for the default 3 replication copies for every block. While placing in the data nodes,

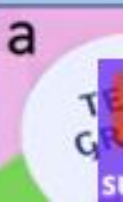
The key rule followed is "for every block of data, two copies will exist in one rack, third copy in the different rack". This rule is called as "**Replica Placement Policy**".





## Rack Awareness: Example

- **When a new block is created**: The First replica is placed on the local node. The Second one is placed on a different rack and the third one is placed on a different node at the local rack.
- **When re-replicating a block**, if the number of an existing replica is one, place the second one on the different rack. If the number of an existing replica is two and if the two existing replicas are on the same rack, the third replica is placed on a different rack.
- One simple way to store data block replicas could be each one on the separate Rack, however this could increase the latency of Write operations.
- So the **Replication policy** is designed in such a way that it does not impact on data reliability and availability guarantee. At the same time, it does reduce the aggregate network bandwidth used when reading data since a block replica is placed in only two unique racks rather than three.



## Rack Awareness: Performance

- Faster replication operation: Since the replicas are placed within the same rack it would use higher bandwidth and lower latency hence making it faster.
- If YARN is unable to create a container in the same data node where the queried data is located it would try to create the container in a data node within the same rack.
- This would be more performant because of the higher bandwidth and lower latency of the data nodes inside the same rack.



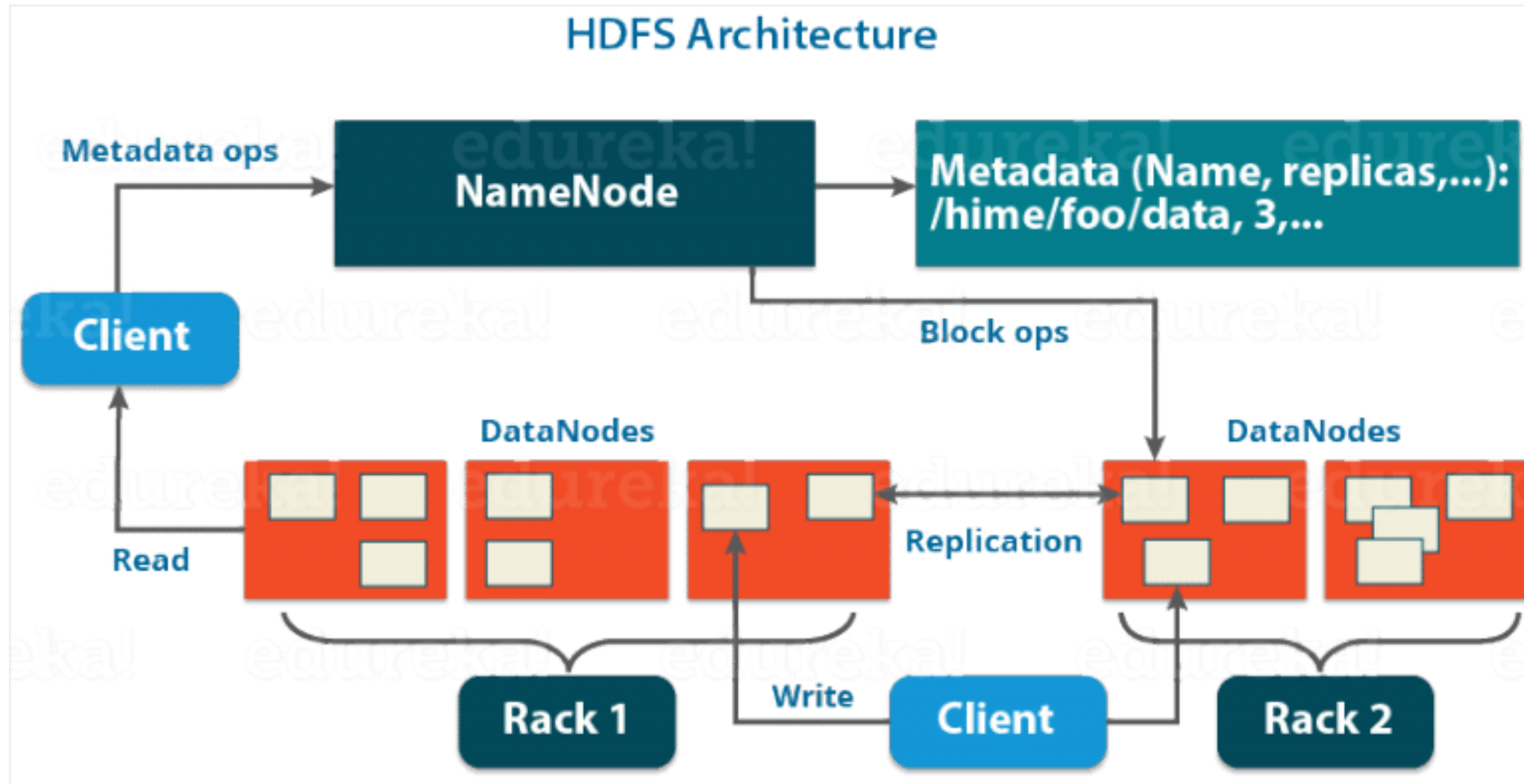
## Advantage

- **Minimize the writing cost and Maximize read speed** – Rack awareness places read/write requests to replicas on the same or nearby rack. Thus minimizing writing cost and maximizing reading speed.
- **Provide maximize network bandwidth and low latency** – Rack awareness maximizes network bandwidth by blocks transfer within a rack.
- It assigns tasks to nodes that are 'closer' to their data in terms of network topology i.e on the same RACK. This is particularly beneficial in cases where tasks cannot be assigned to nodes where their data is stored locally.
- **Data protection against rack failure** – By default, the namenode assigns 2nd & 3rd replicas of a block to nodes in a rack different from the first replica. This provides data protection even against rack failure.

# HDFS services

1. Name Node
2. Secondary Name node
3. Job Tracker
4. Data Node
5. Task Tracker

# HDFS Architecture





# HDFS Architecture

- Hadoop Distributed File System follows the **master-slave architecture**. Each cluster comprises a **single master node** and **multiple slave nodes**. Internally the files get divided into one or more **blocks**, and each block is stored on different slave machines depending on the **replication factor**.
- The master node stores and manages the file system namespace, that is information about blocks of files like block locations, permissions, etc. The slave nodes store data blocks of files.
- ***Namespace*** we *mean* a certain location on the *hdfs*. In Hadoop we refer to a Namespace as a file or directory which is handled by the *Name Node*. According to Hadoop, Name Node manages the file system *namespace*. It maintains the file system tree, and the ***metadata*** of all the files and the directories in the tree.
- Namespace act as a container where file name grouping and metadata which also contains things like the owners of files, permission bits, block location, size etc will be present.

# NameNode

- NameNode is the master node in the Apache Hadoop HDFS Architecture that maintains and manages the blocks present on the DataNodes (slave nodes).
- NameNode is a very highly available server that manages the File System Namespace and controls access to files by clients.

# Functions of Namenode

- It is the master daemon that maintains and manages the DataNodes (slave nodes)
- It records the metadata of all the files stored in the cluster
  - **FsImage:** Fsimage stands for File System image. It contains the complete state of the file system namespace since the start of the NameNode.
  - **EditLogs:** It contains all the recent modifications made to the file system with respect to the most recent FsImage.

# Functions of Namenode

- It records each change that takes place to the file system metadata.
- It regularly receives a Heartbeat and a block report from all the DataNodes in the cluster to ensure that the DataNodes are live.
- It keeps a record of all the blocks in HDFS and in which nodes these blocks are located.
- The NameNode is also responsible to take care of the replication factor of all the blocks.
- In case of the DataNode failure, the NameNode chooses new DataNodes for new replicas, balance disk usage and manages the communication traffic to the DataNodes.

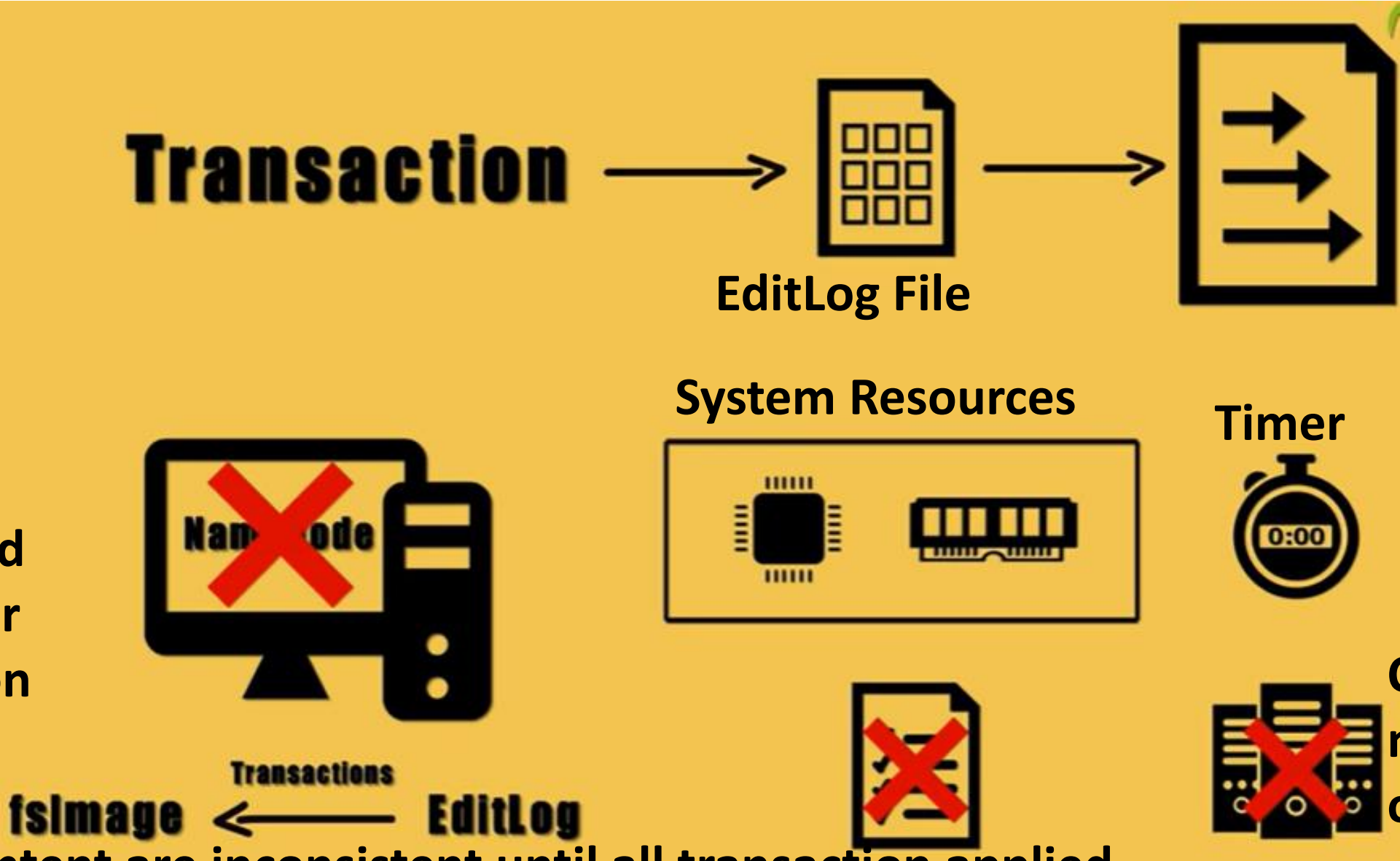
# Data Nodes

- DataNodes are the slave nodes in HDFS. Unlike NameNode, DataNode is a commodity hardware, that is, a non-expensive system which is not of high quality or high-availability.

# Functions of Data Node

- These are slave daemons or process which runs on each slave machine.
- The actual data is stored on DataNodes.
- The DataNodes perform the low-level read and write requests from the file system's clients.
- They send heartbeats to the NameNode periodically to report the overall health of HDFS, by default, this frequency is set to 3 seconds.

# Secondary NameNode



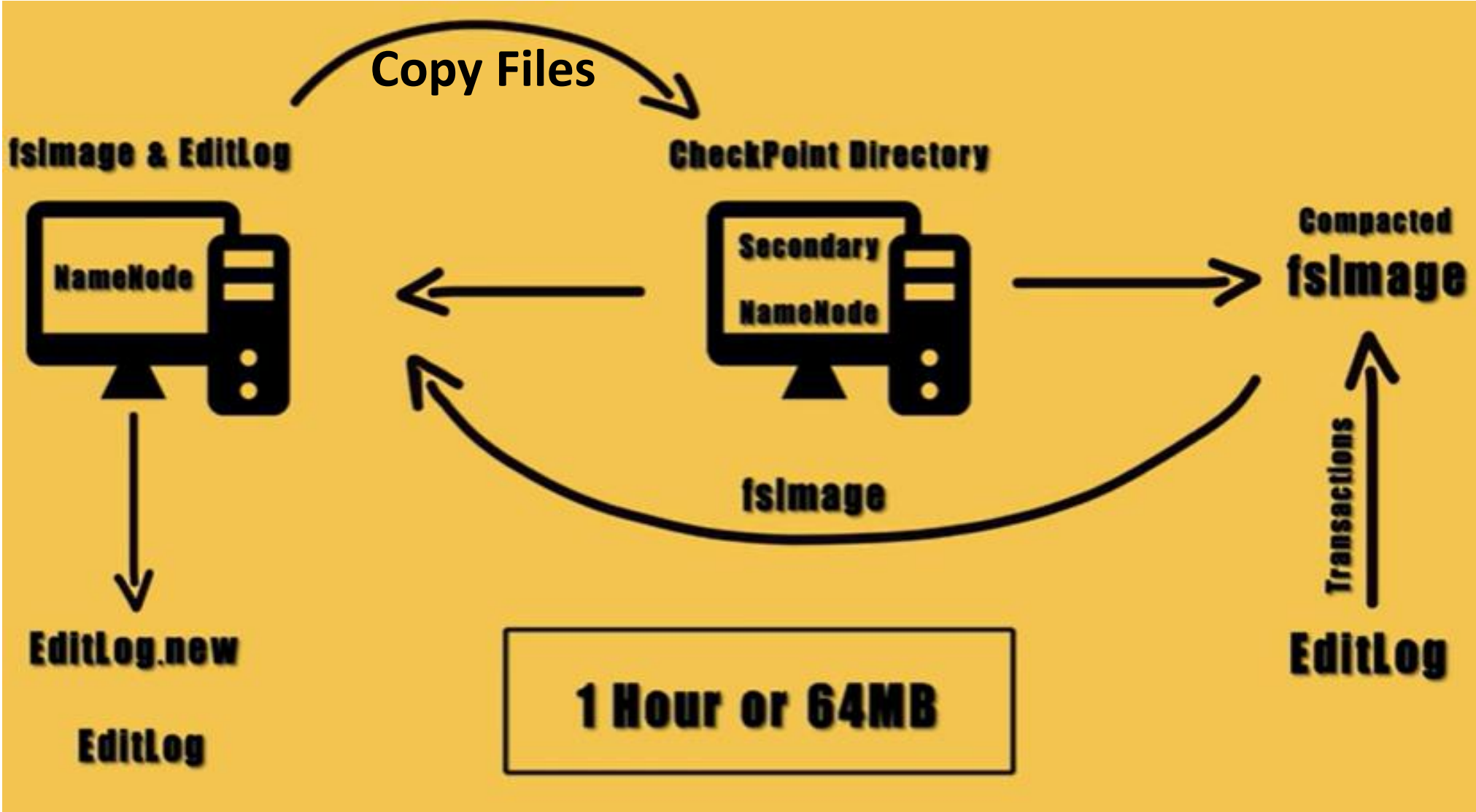
Failed due  
to corrupted  
metadata or  
other reason

Cluster  
not  
operable

FsImage content are inconsistent until all transaction applied



# Secondary NameNode

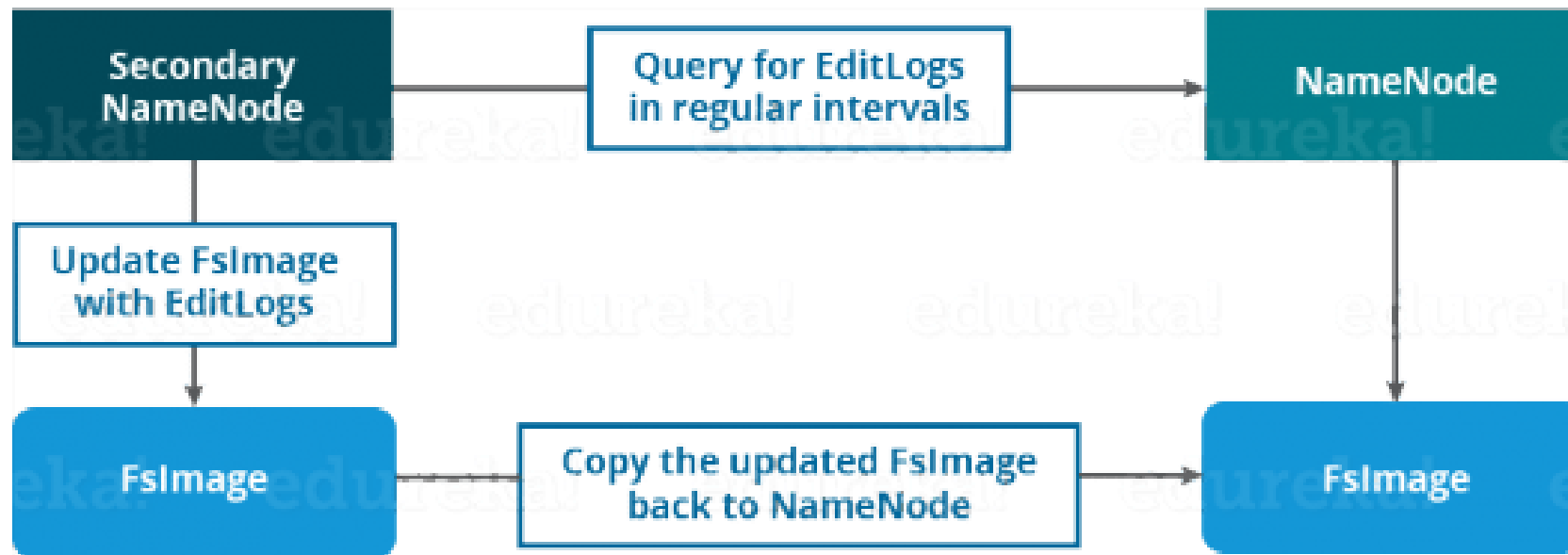


Rename EditLog.New to EditLog

Size of EditLog > 64MB

# Secondary NameNode

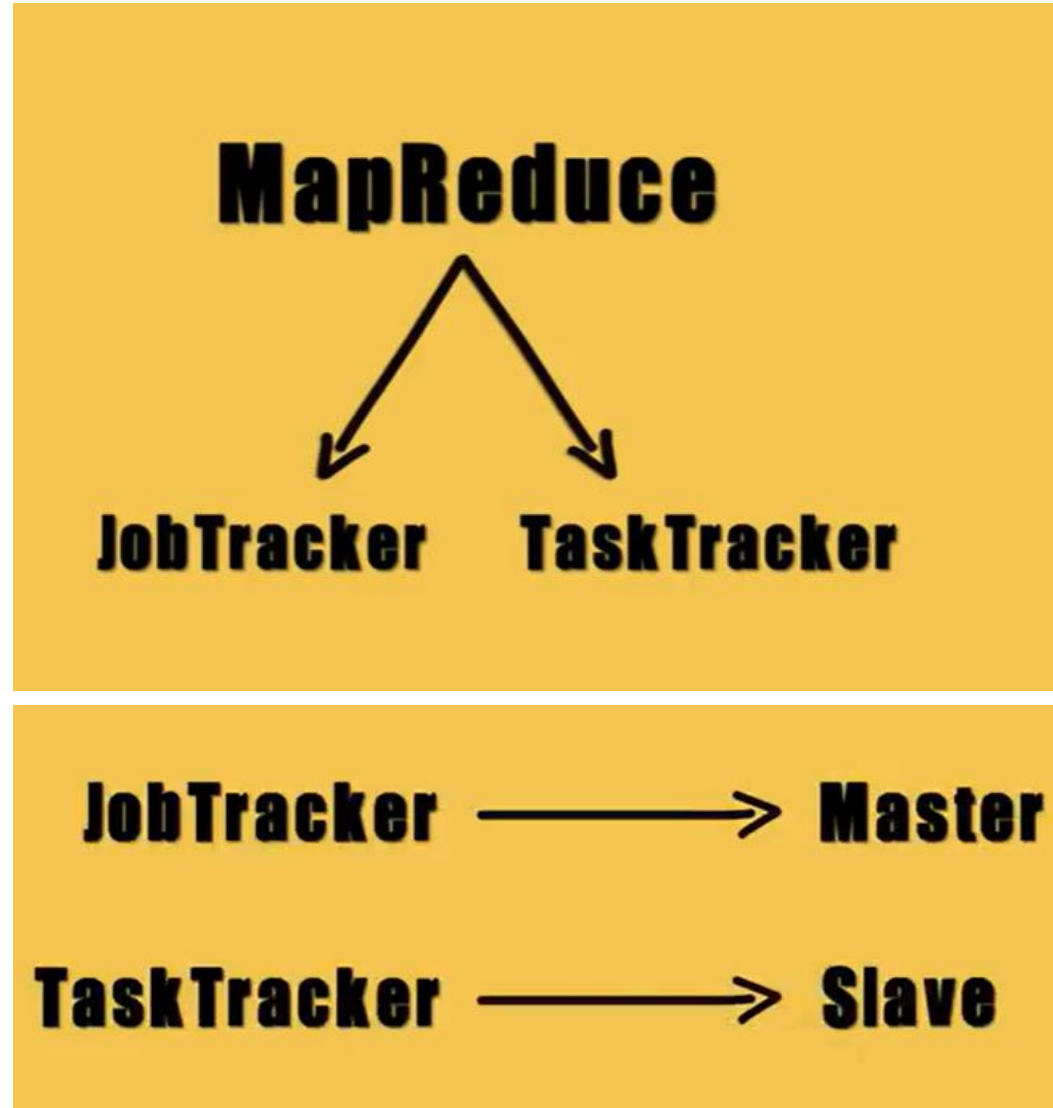
- Apart from these two daemons, there is a third daemon or a process called Secondary NameNode. The Secondary NameNode works concurrently with the primary NameNode as a **helper daemon**.

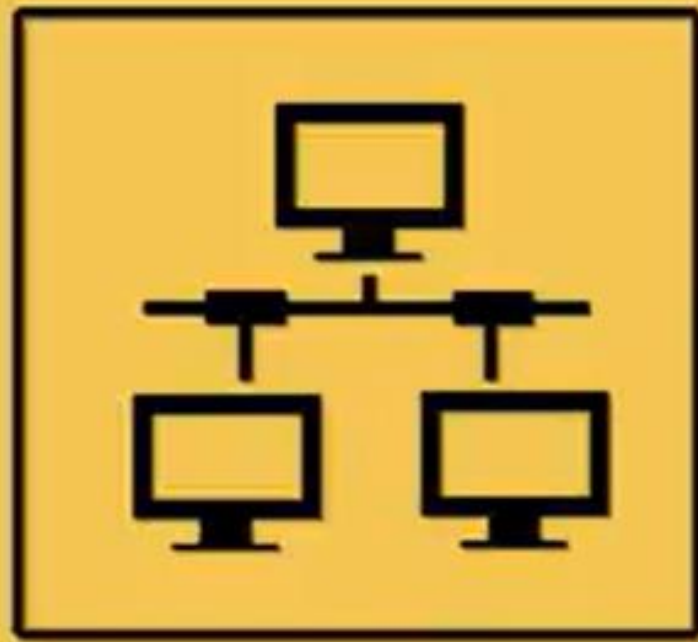


# *Functions of Secondary NameNode*

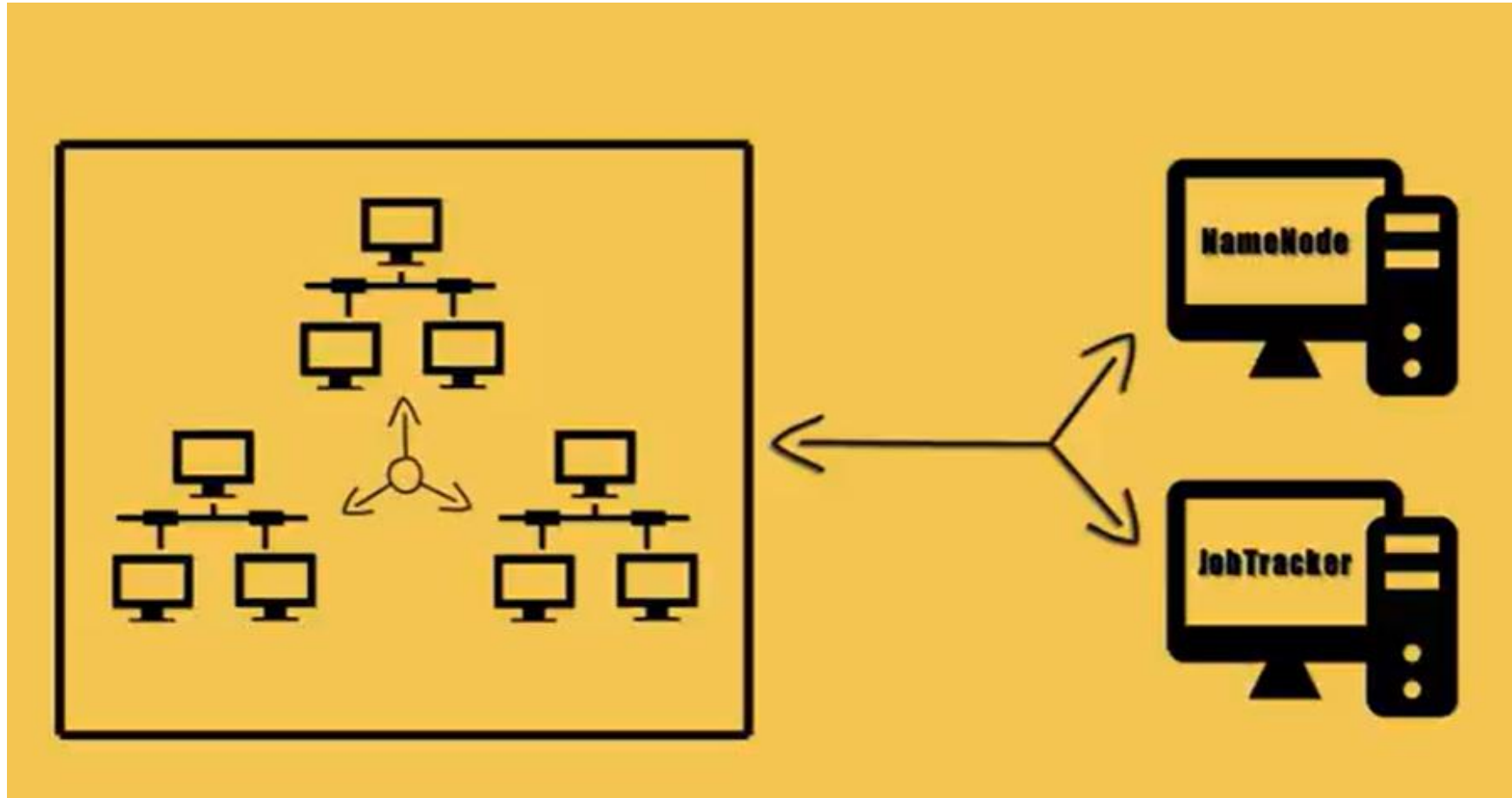
- The Secondary NameNode is one which constantly reads all the file systems and metadata from the RAM of the NameNode and writes it into the hard disk or the file system.
- It is responsible for combining the **EditLogs** with **FsImage** from the NameNode.
- It downloads the EditLogs from the NameNode at regular intervals and applies to FsImage. The new FsImage is copied back to the NameNode, which is used whenever the NameNode is started the next time.

# MapReduce Demons: Job Tracker and Task Tracker



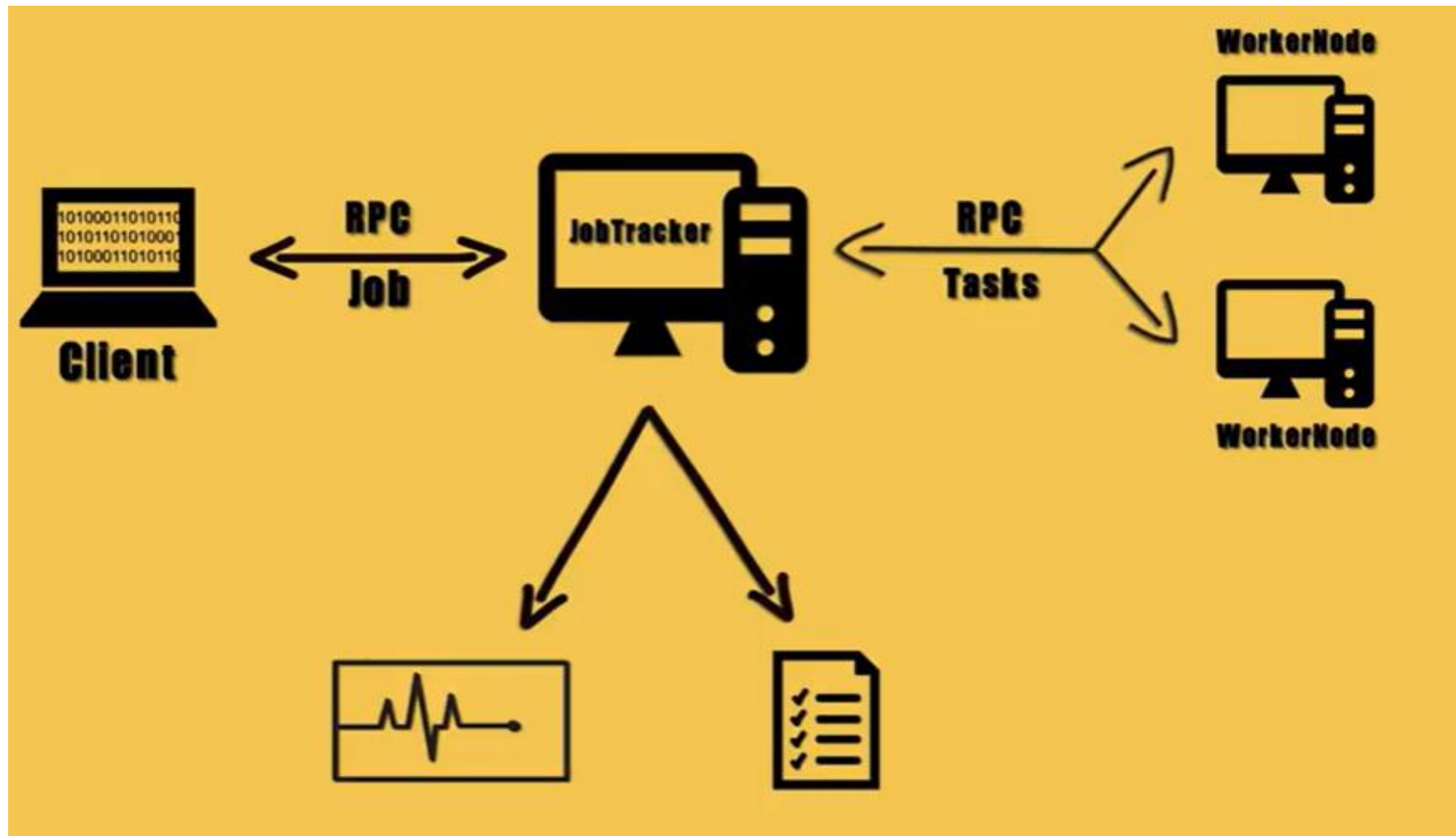


# MapReduce Demons: Job Tracker and Task Tracker

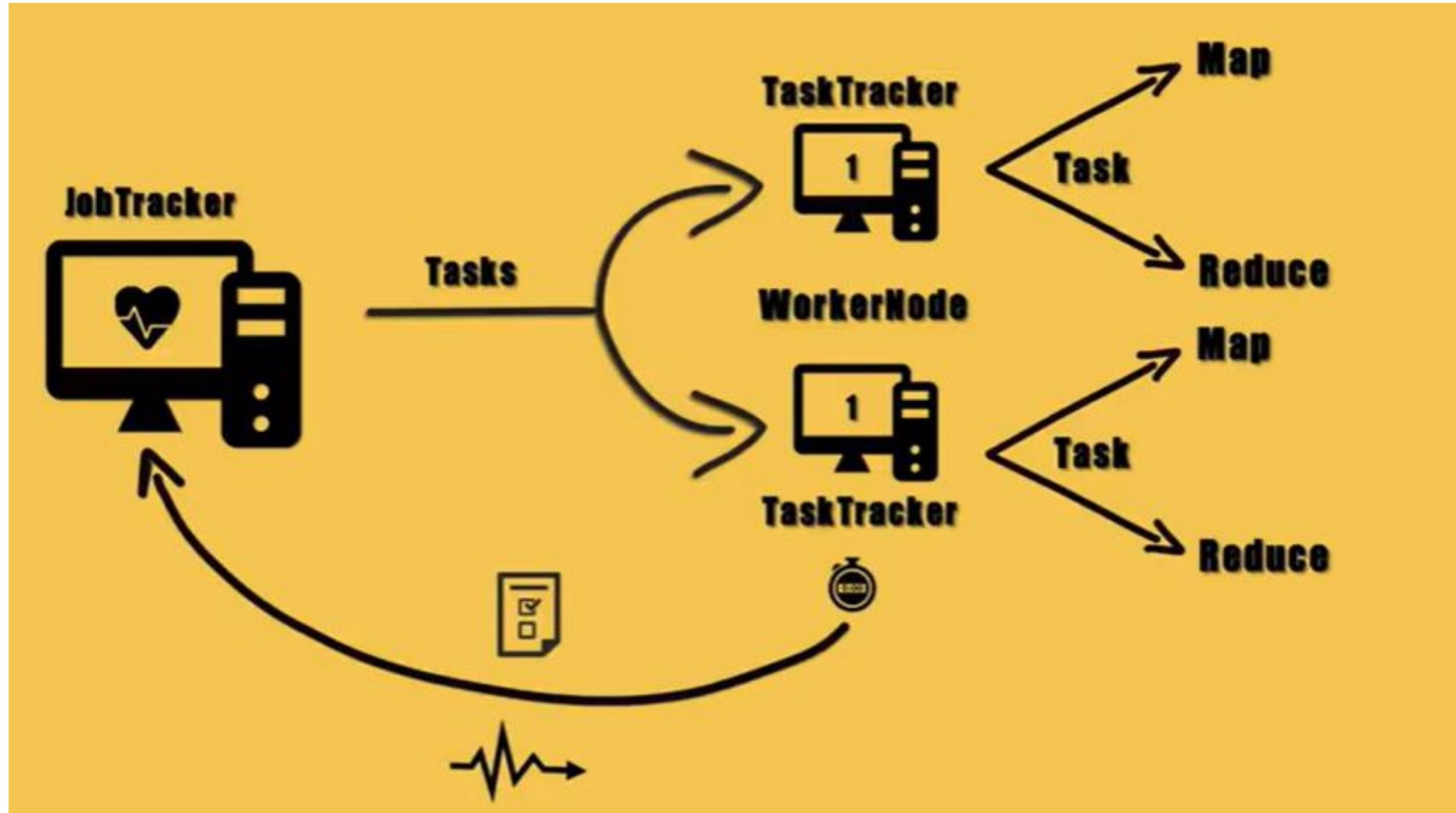


Large Cluster have Namenode and JobTracker in different Machine

# Job Tracker

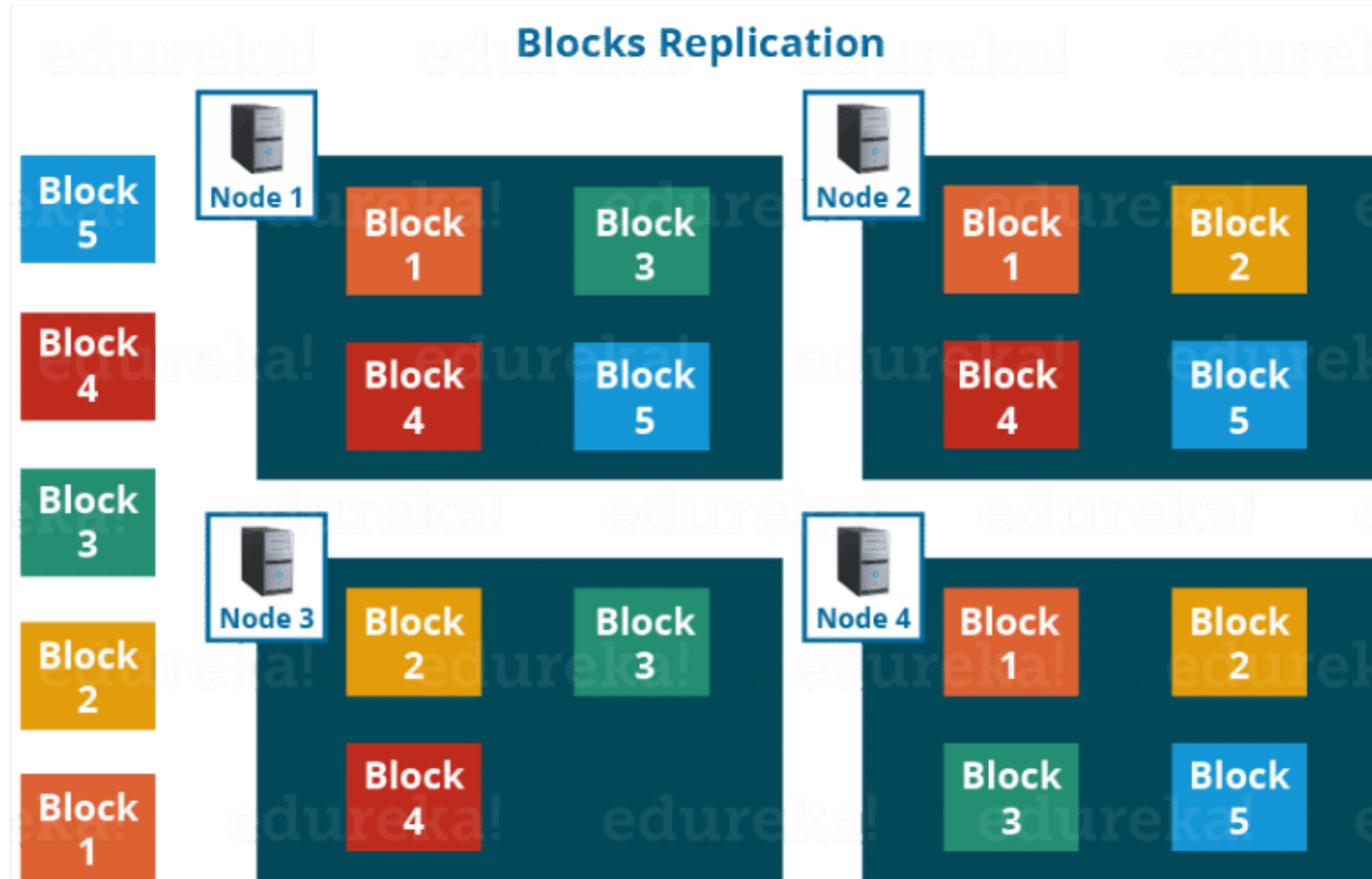


# Task Tracker





# Replication Management



if you are storing a file of 128 MB in HDFS using the default configuration, you will end up occupying a space of 384 MB ( $3 \times 128$  M) as the blocks will be replicated three times and each replica will be residing on a different DataNode.

# NETWORK TOPOLOGY AND HADOOP

What does it mean for two nodes in a local network to be “close” to each other?

- Processes on the same node
- Different nodes on the same rack
- Nodes on different racks in the same data center
- Nodes in different data centers

