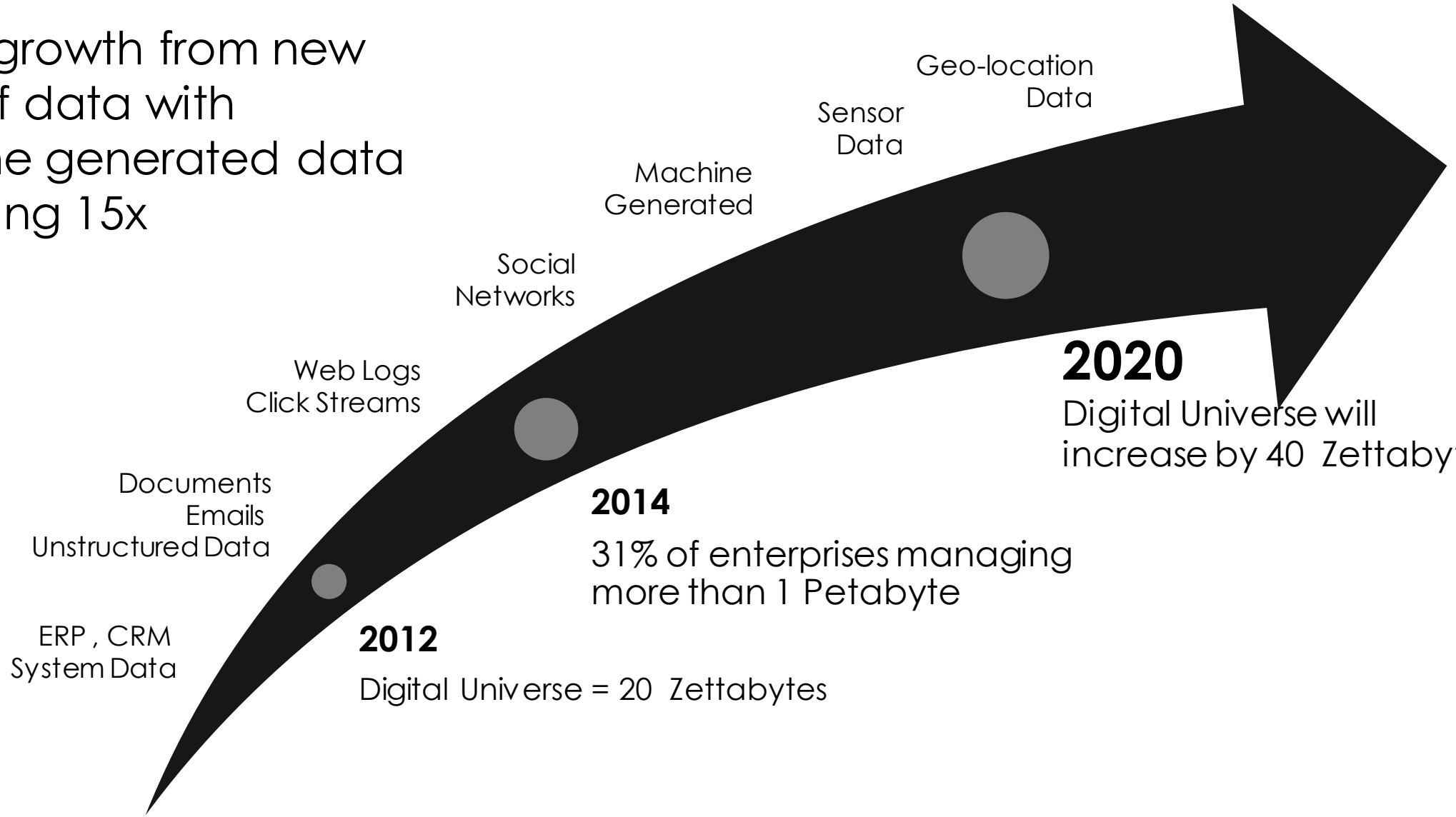
BIG DATA

HADOOP

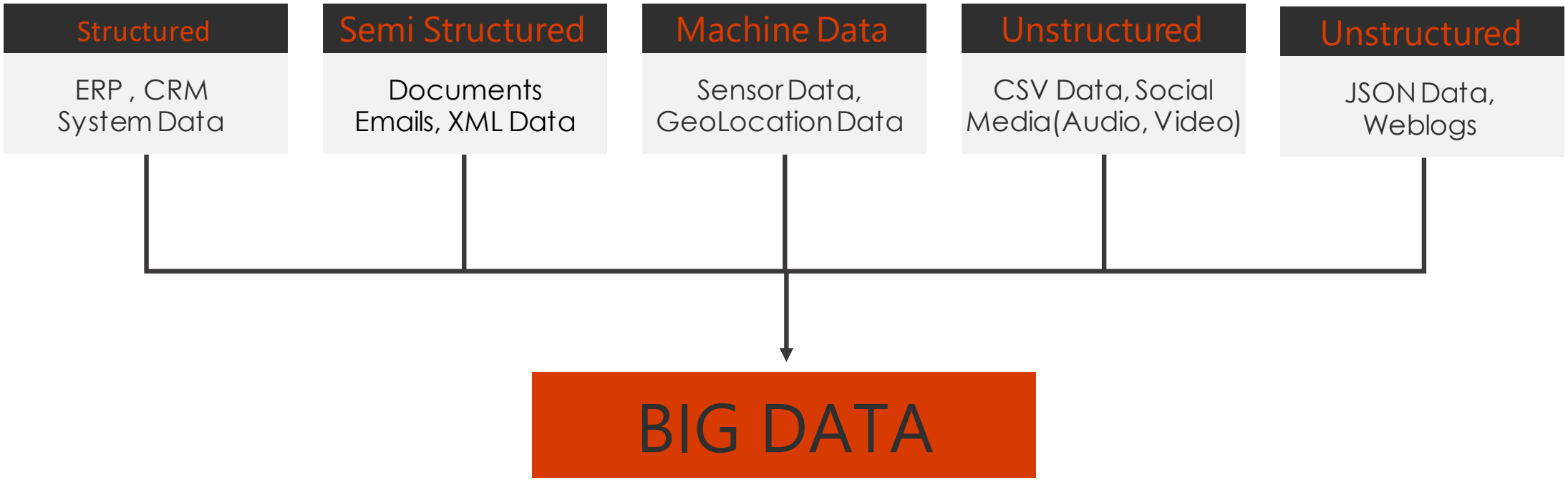
HADOOP : INTRODUCTION

85% of growth from new types of data with Machine generated data increasing 15x

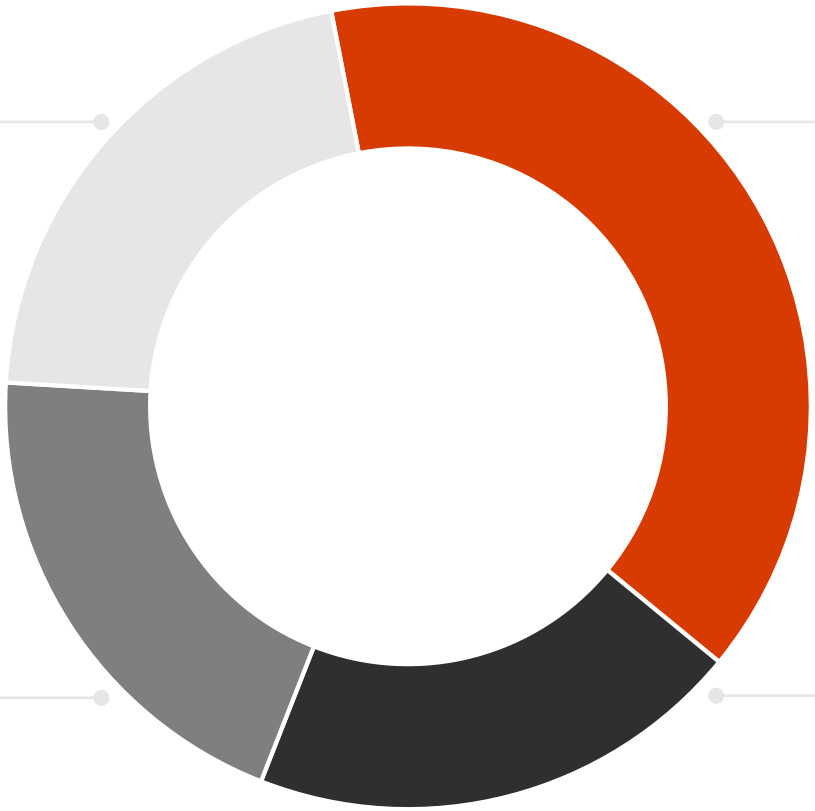


HADOOP : INTRODUCTION

Over decades , there has been a humongous explosion in data. From each sector its increasing per day.



MAJOR CONTRIBUTORS



STRUCTURED DATA

21%

UNSTRUCTURED DATA

39%

SENSOR & M2M DATA

20%

SEMISTRUCTURED DATA

20%

MAJOR COMPETITORS

- **STRUCTURED**

  Represented in a tabular format and Stored in relational dB like MyQL, Oracle Database & DB2

- **SEMI-STRUCTURED**

  Does not have formal Data modelling

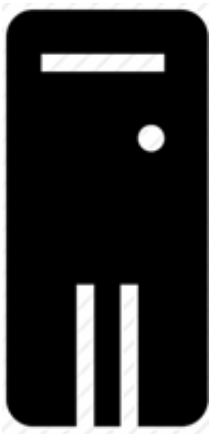
- **UNSTRUCTURED**

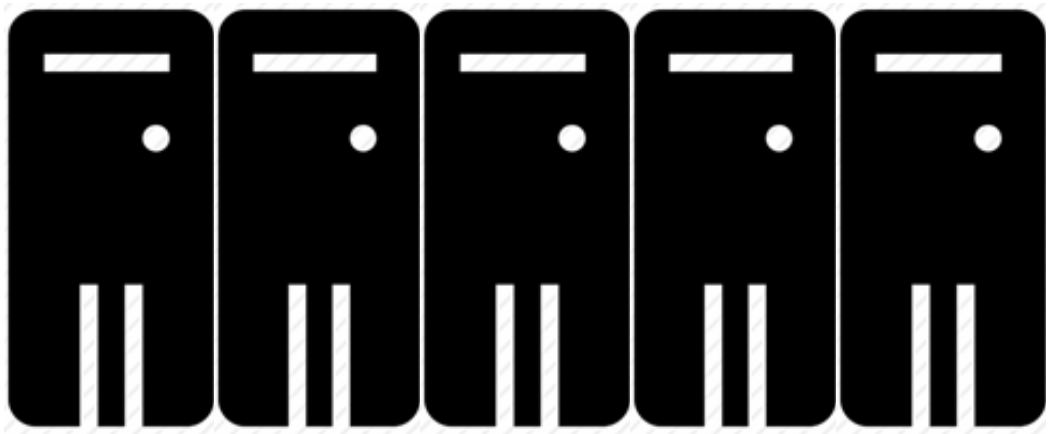
  Does not follow a pre-defined data modelling, Stored like a Flat-Structured model, Text files, web logs & machine logs.

HADOOP FRAMEWORK

- HADOOP is an open-source software framework developed on Distributed File System (DFS) for storing data and running applications on clusters of commodity hardware.
- HDFS is highly fault tolerant and designed using low-cost hardware.
- It provides massive storage for any kind of data, enormous processing power and the ability to handle virtually limitless concurrent tasks or jobs.
- Apache Hadoop is to break up unstructured data and distribute it into many parts for concurrent data analysis.

DISTRIBUTED FILE SYSTEM

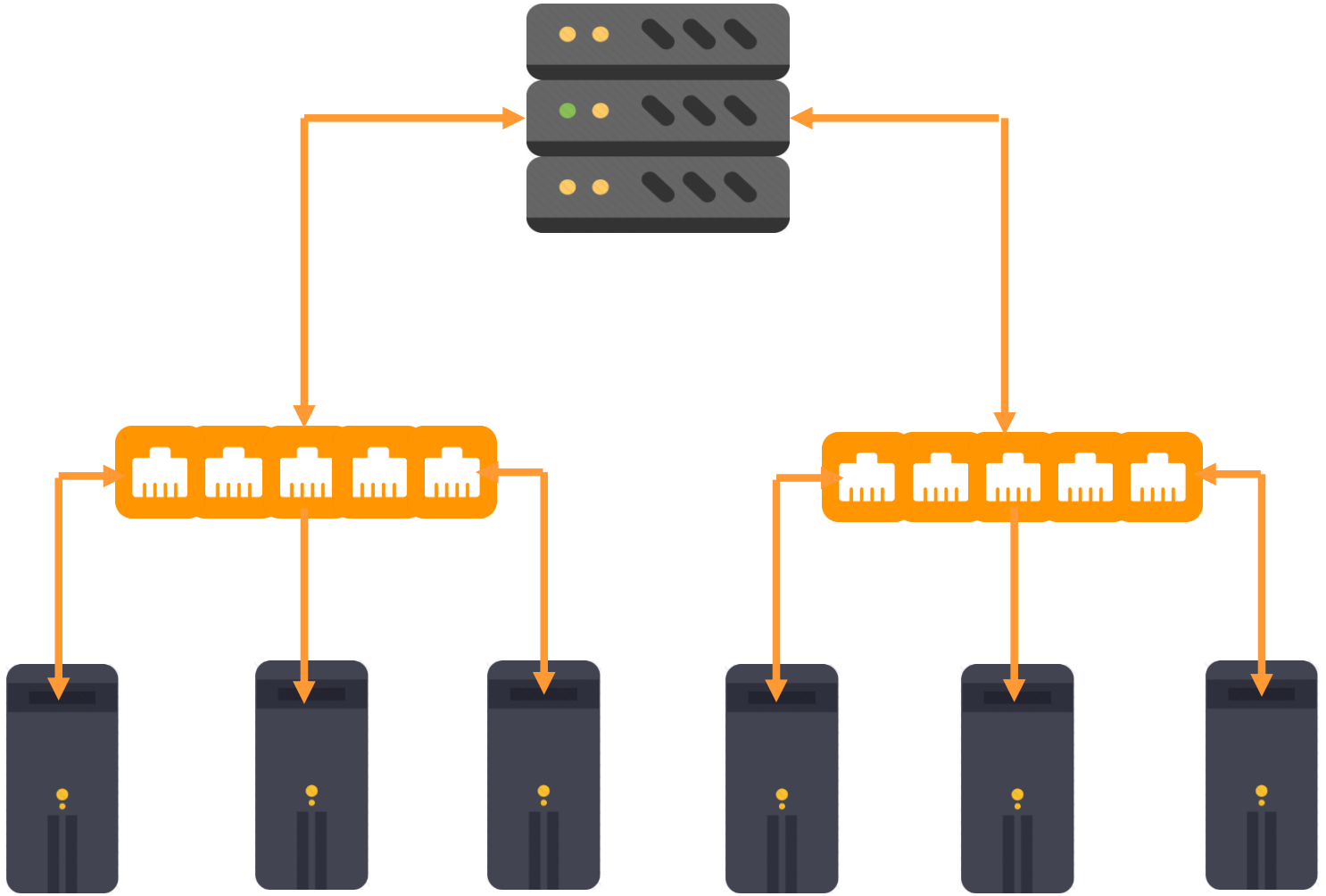




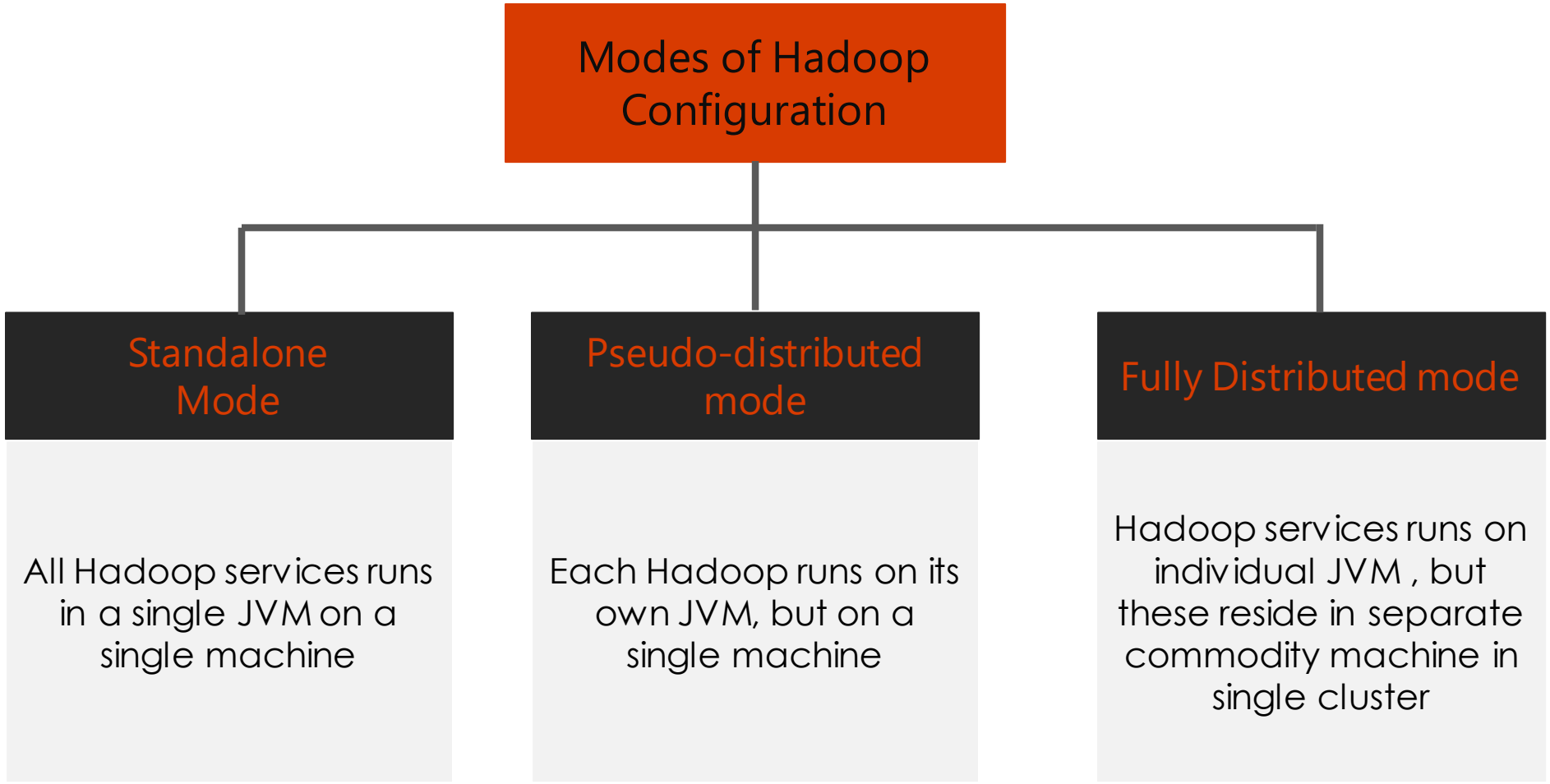
1 Machine
4 I/O operational channels
Processing speed : 100MB/s

5 Machine
4 I/O operational channels
Processing speed : 100MB/s

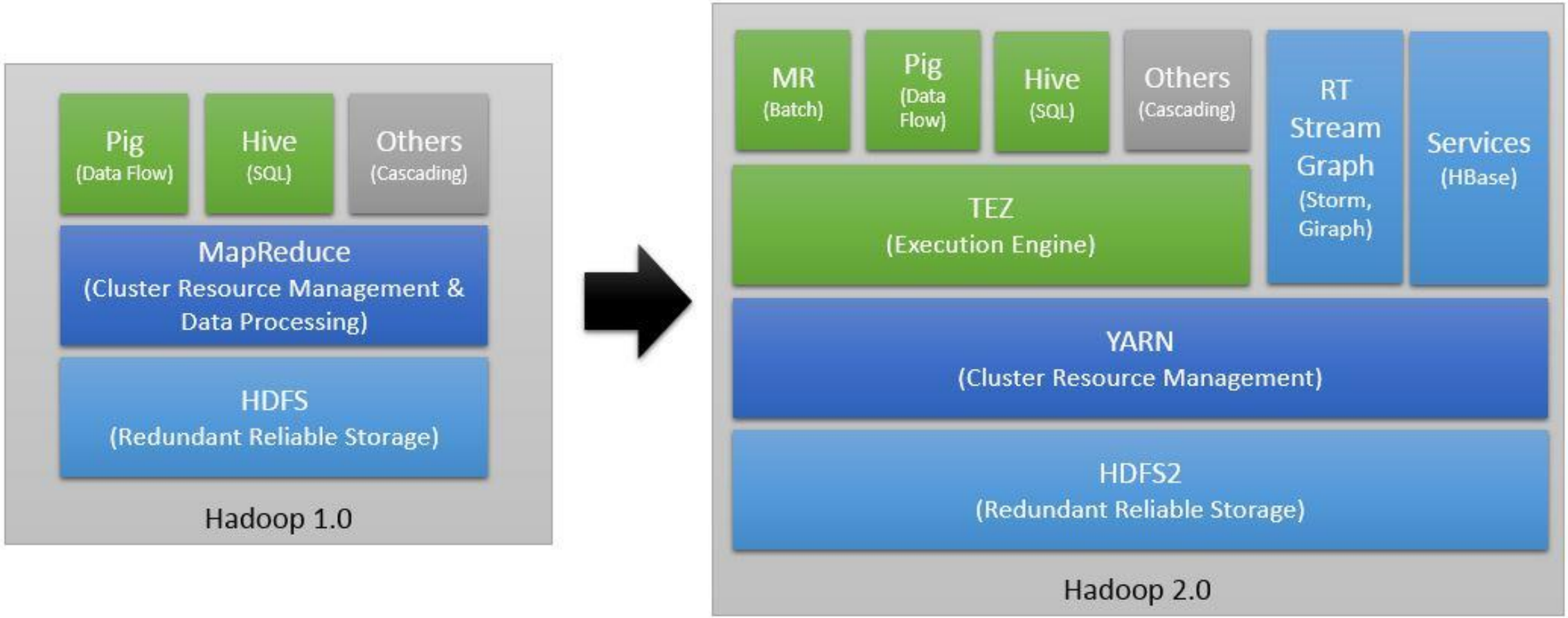
HADOOP CLUSTER



HADOOP CONFIGURATION



HADOOP 1.0 VS HADOOP 2.0

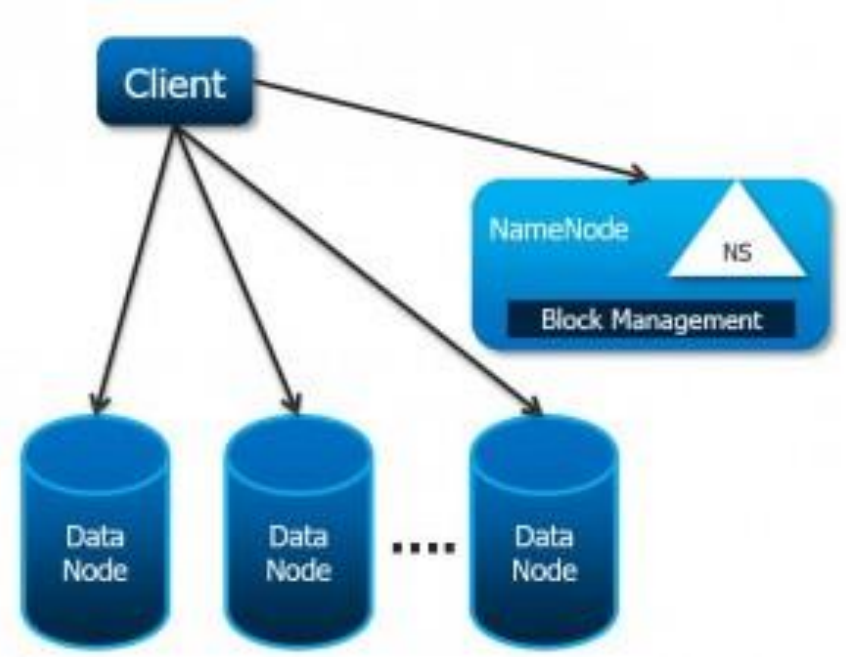


HADOOP 1.0 VS HADOOP 2.0

HDFS Federation

- Hadoop Cluster can scale up to hundreds of DataNodes, and the NameNode keeps all its metadata in memory (RAM). This results in the limitation on maximum number of files a Hadoop Cluster can store (typically 50-100M files). As your data size and cluster size grow this becomes a bottleneck as size of your cluster is limited by the NameNode memory.

- Hadoop 2.0 feature HDFS Federation which allows horizontal scaling for Hadoop distributed file system (HDFS). HDFS Federation supports **Multiple NameNodes and manages Namespaces**. The DataNodes are used as common storage for blocks by all the Namenodes. DataNodes send periodic heartbeats and block reports and handle commands from its associated NameNodes.



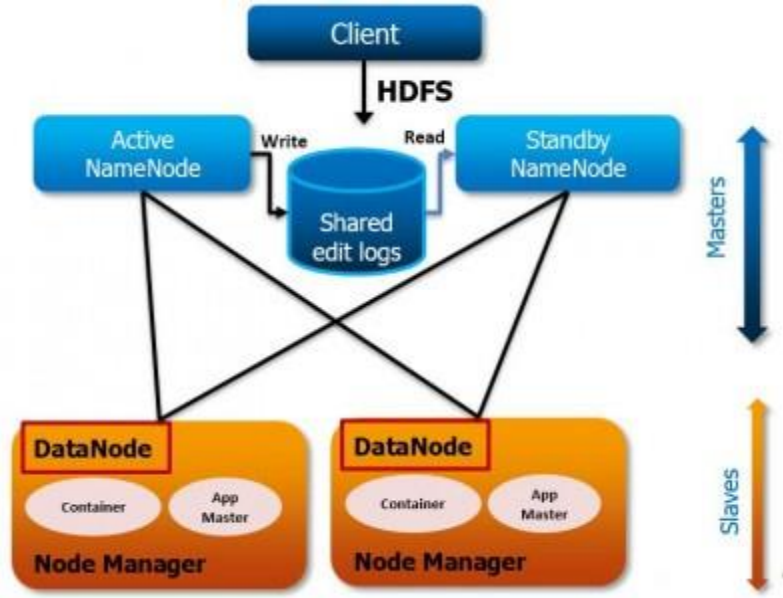
HADOOP 1.0 VS HADOOP 2.0

NameNode High Availability

- In Hadoop 1.x, NameNode was single point of failure. NameNode failure makes the Hadoop Cluster inaccessible. Usually, this is a rare occurrence because of sever failure.

- In case of NameNode failure, Hadoop Administrators need to manually recover the NameNode using Secondary NameNode.

  Hadoop 2.0 Architecture supports multiple NameNodes to remove this bottleneck. Hadoop 2.0, NameNode High Availability feature comes with support for a Passive Standby NameNode. These Active-Passive NameNodes are configured for automatic failover.



HADOOP 1.0 VS HADOOP 2.0

YARN

- In Hadoop 1.0 JobTracker is responsible for both managing the cluster's resources and driving the execution of the MapReduce job.

- YARN splits up the two major functionalities of overburdened Jobtracker (resource management and job scheduling/monitoring) into two separate daemons:
  - **A Global Resource Manager** and
  - **Application Master.**

  A Resource Manager (RM) focuses on managing the cluster resources

  An Application Master (AM) one-per-running-application, manages each running application & is responsible for containers, monitoring their resource usage (cpu, memory, disk, network) and reporting the same to the ResourceManager/Scheduler.

  YARN provides central resource manager. With YARN, you can now run multiple applications in Hadoop, all sharing a common resource.

HADOOP 1.0 VS HADOOP 2.0

YARN

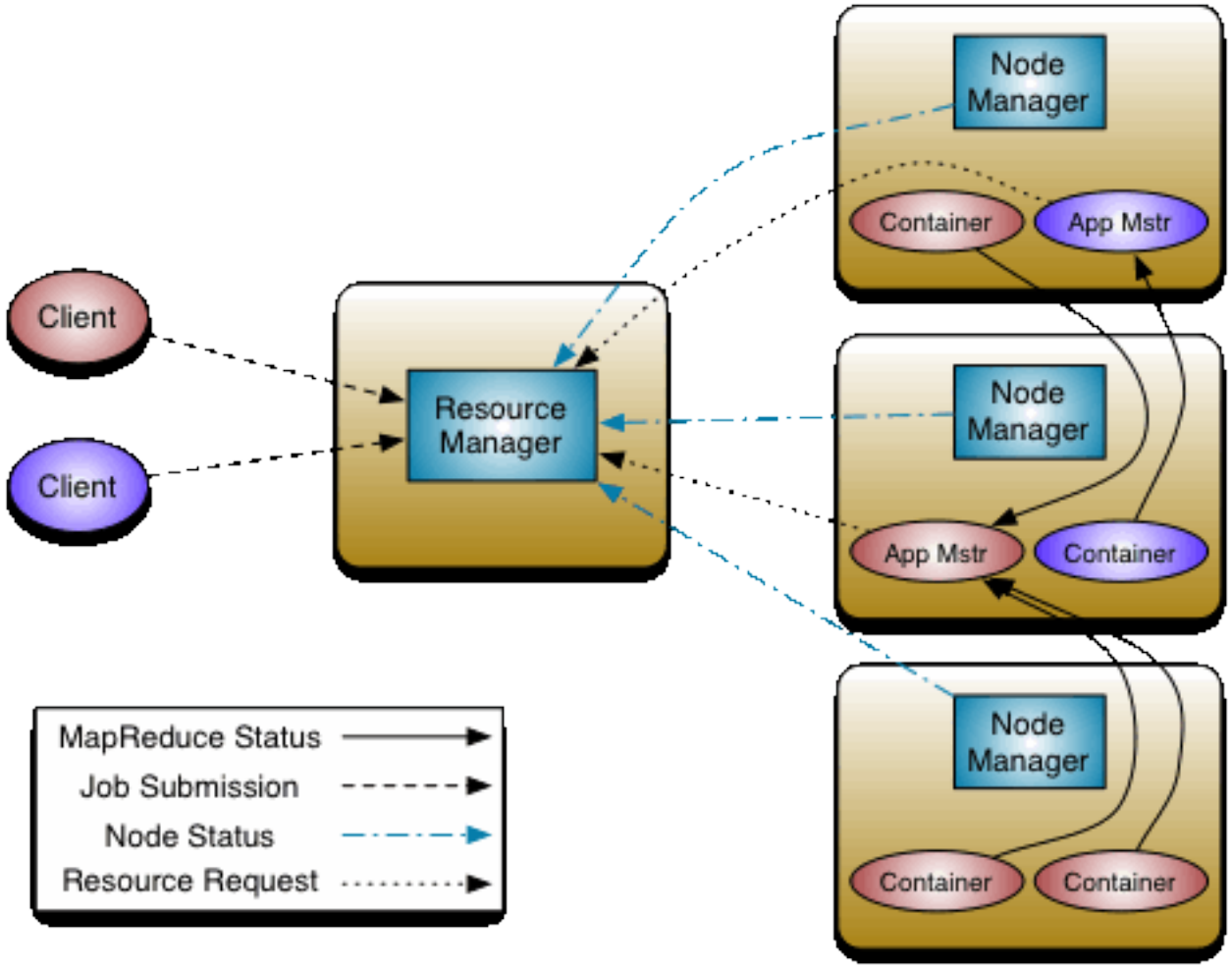
RESOURCE MANAGER

- It is a cluster level (one for each cluster) component and runs on the master machine
- It has two components: **Scheduler** & **Applications Manager**
- The Scheduler is responsible for allocating resources to the various running applications subject to familiar constraints of capacities, queues etc. The Scheduler is pure scheduler in the sense that it performs no monitoring or tracking of status for the application.
- The Applications Manager is responsible for accepting job-submissions, negotiating the first container for executing the application specific Applications Master and provides the service for restarting the Applications Master container on failure.

- It keeps a track of the heartbeats from the Node Manager

HADOOP 1.0 VS HADOOP 2.0

YARN (Yet Another Resource Negotiator)



HADOOP 1.0 VS HADOOP 2.0

HDFS SNAPSHOT

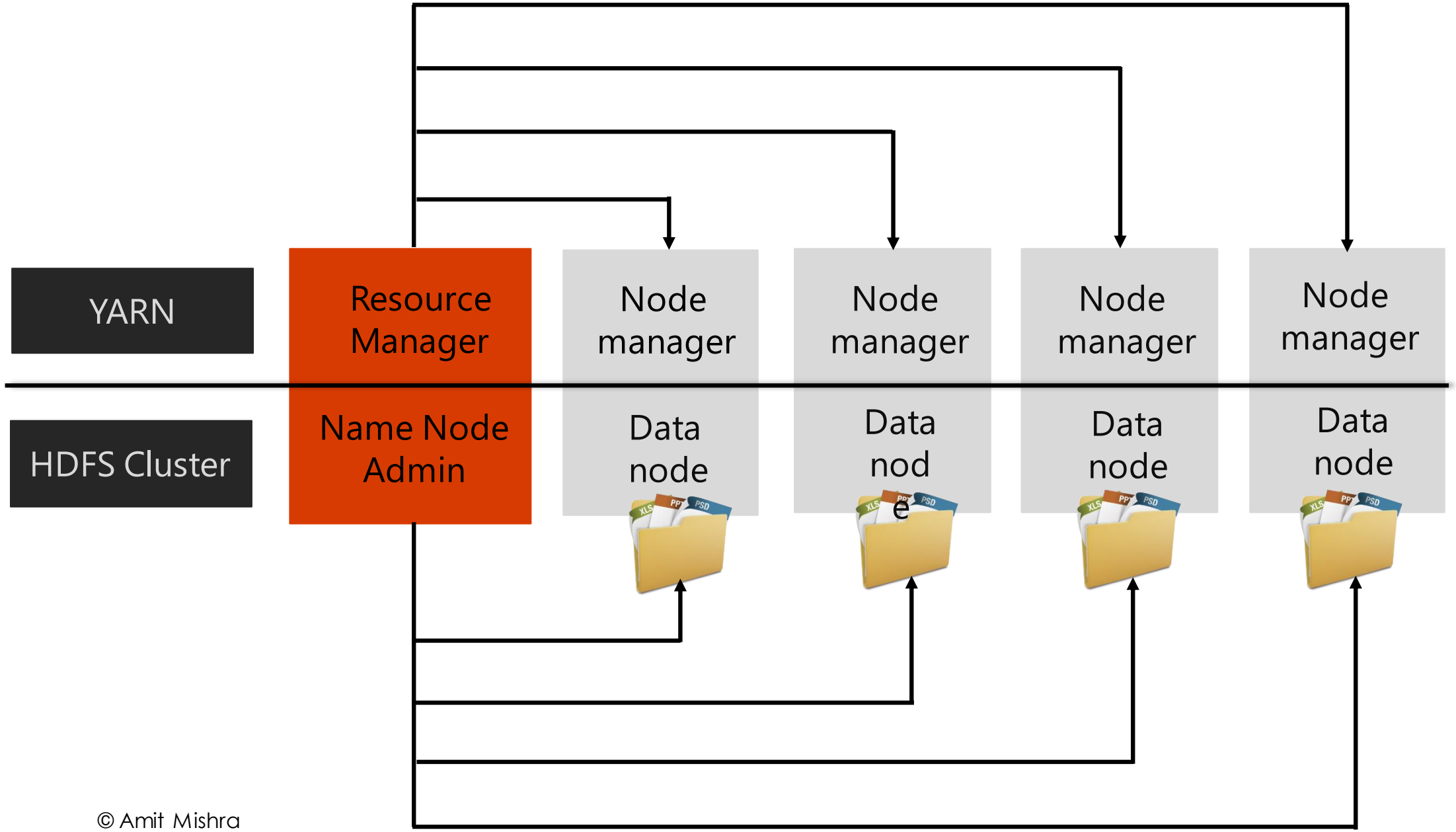
- Hadoop 2 adds support for file system snapshots. A snapshot is a point-in-time image of the entire file system or a sub tree of a file system.
  A snapshot has many uses:

- Protection against user errors: An admin can set up a process to take snapshots periodically. If a user accidentally deletes files, these can be restored from the snapshot that contains the files.

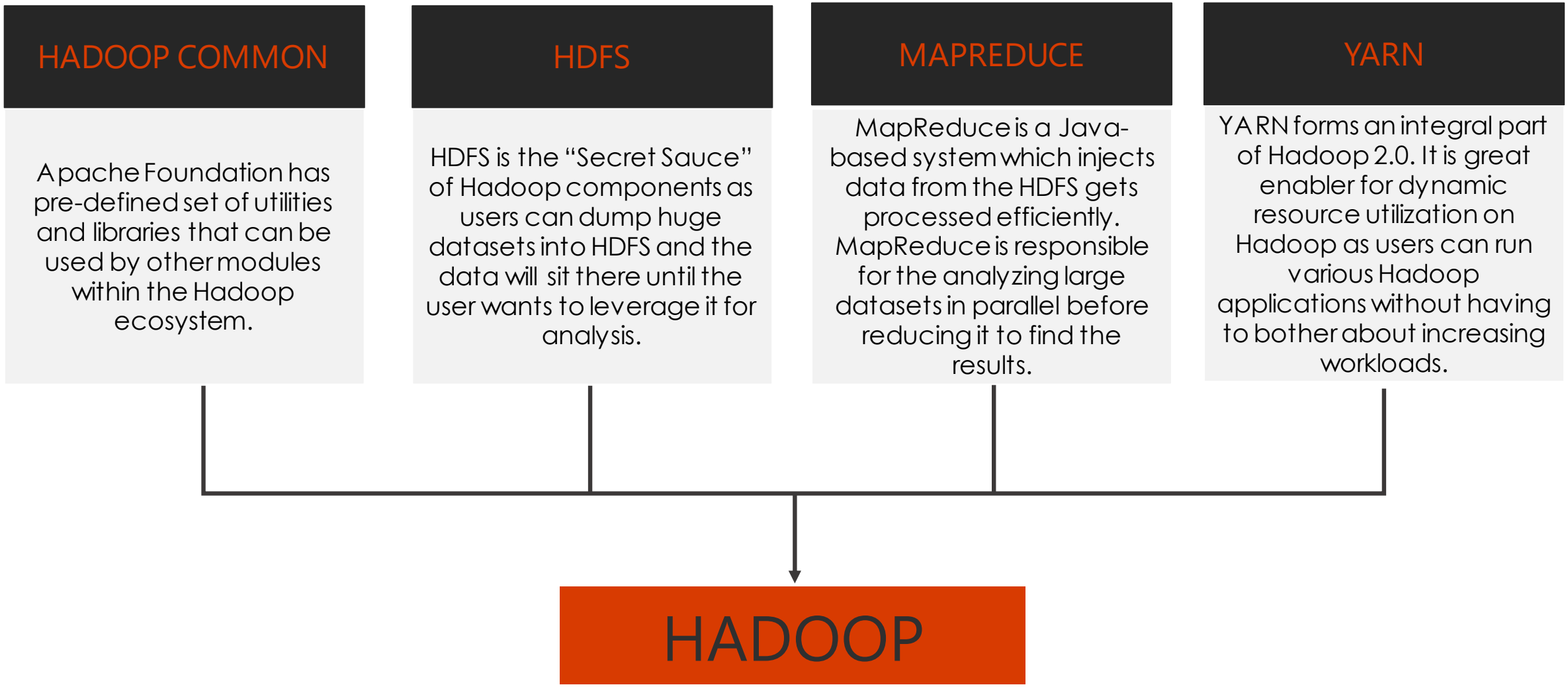
- Backup: If an admin wants to back up the entire file system or a subtree in the file system, the admin takes a snapshot and uses it as the starting point of a full backup.

- Disaster recovery: Snapshots can be used for copying consistent point-in-time images over to a remote site for disaster recovery.

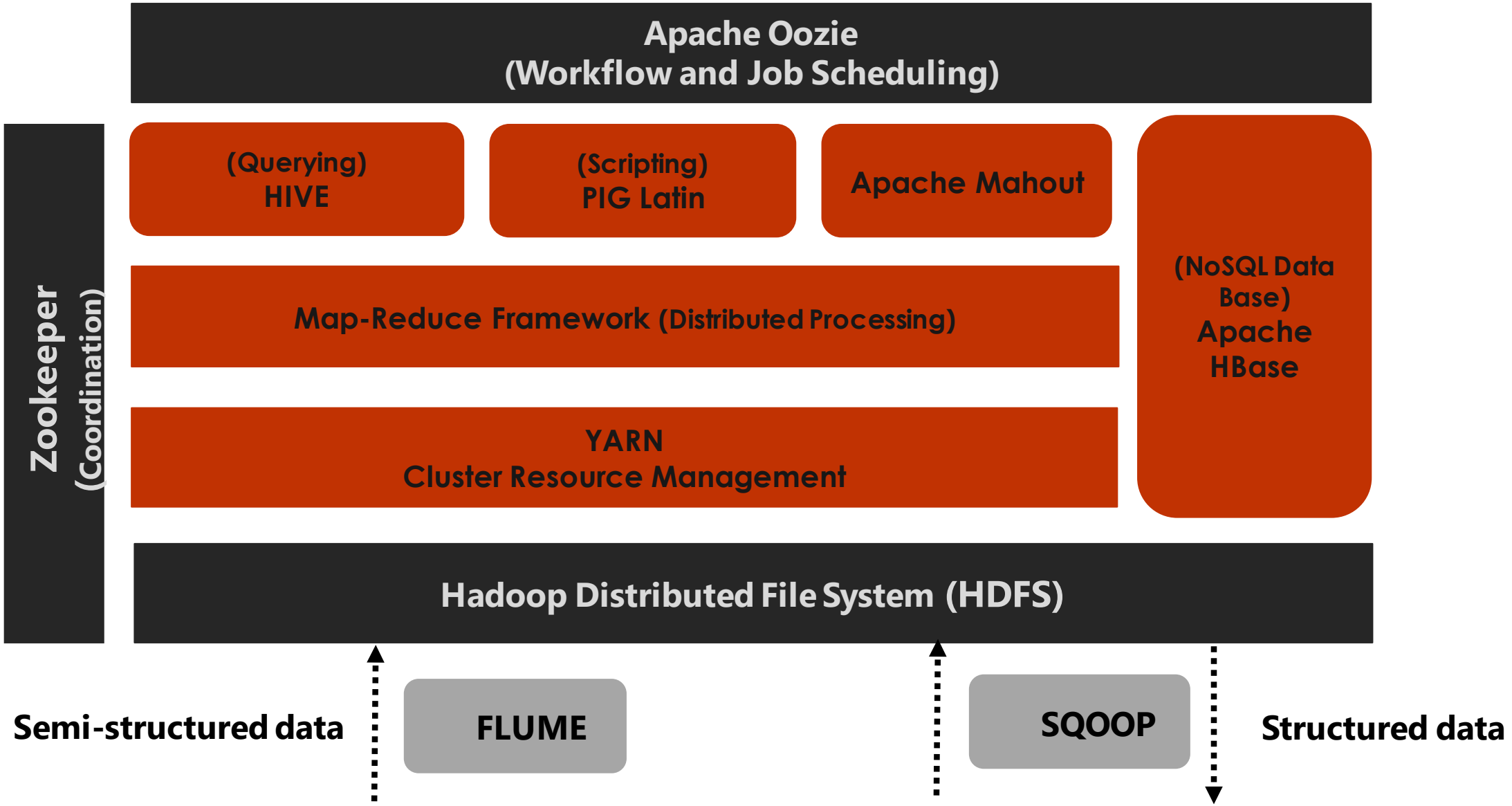
HADOOP 2.x Core Components



HADOOP COMPONENTS



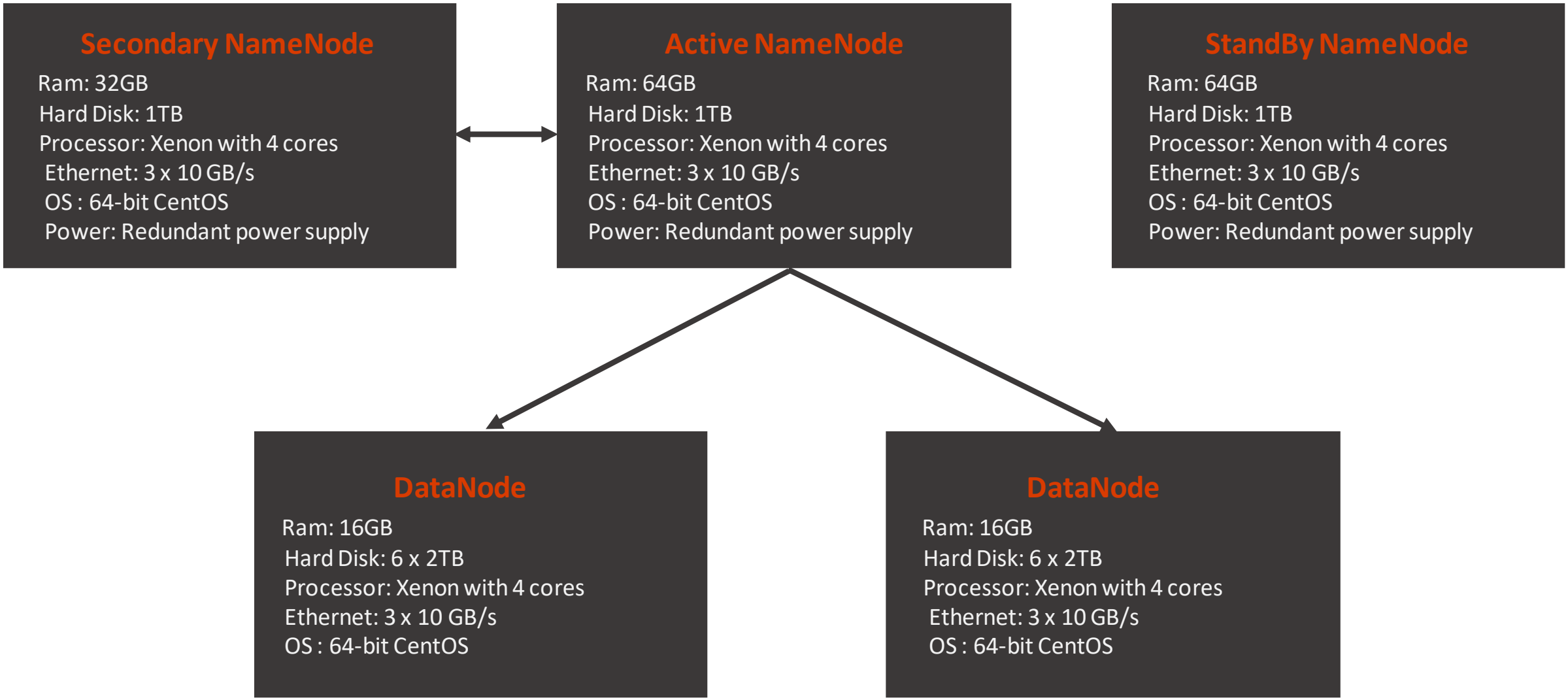
HADOOP ECOSYSTEM



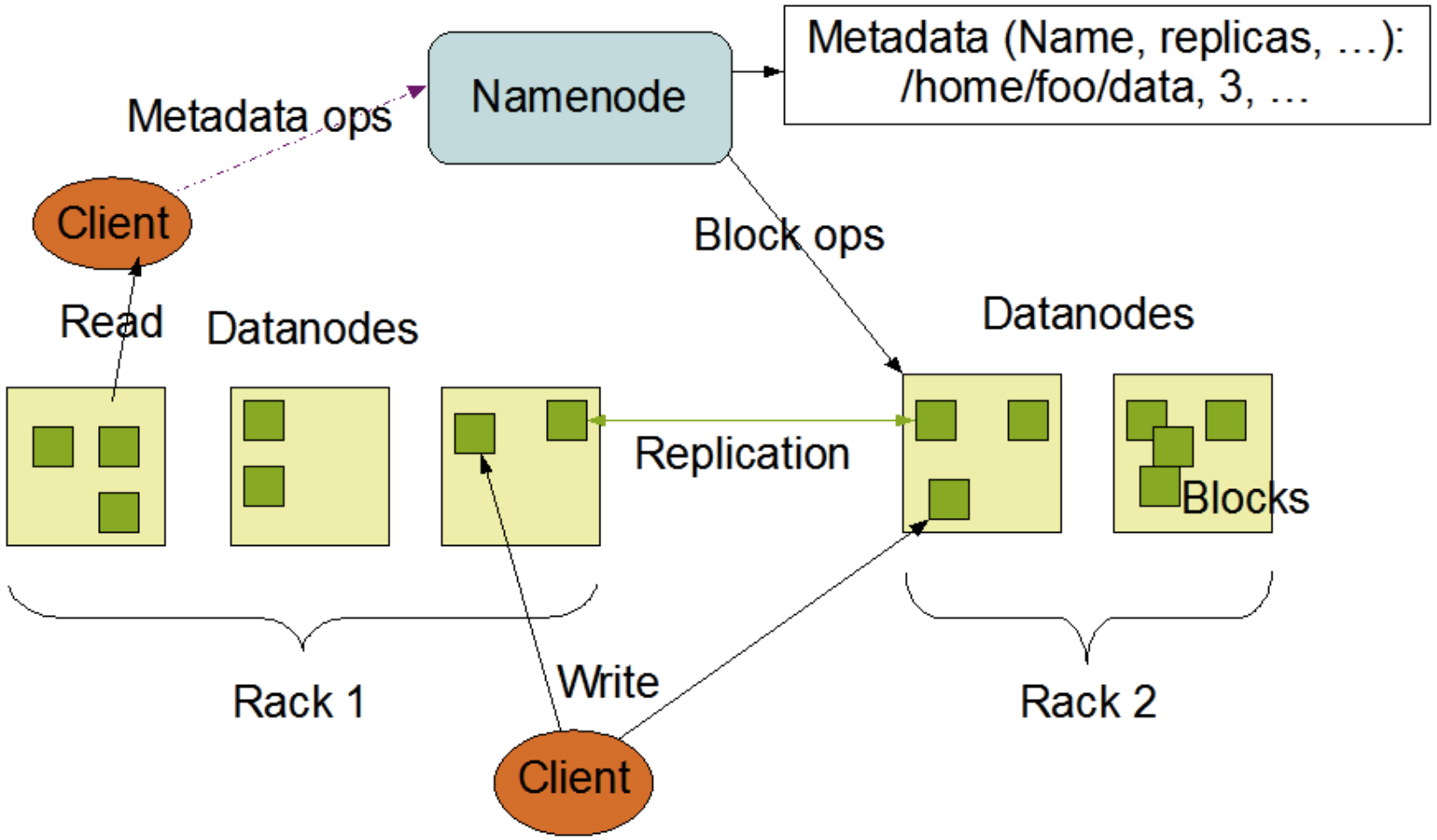
BIG DATA

HADOOP HDFS

HADOOP HDFS



HDFS ARCHITECHTURE



HDFS RACK AWARENESS

In a large cluster of Hadoop, in order to improve the network traffic while reading/writing HDFS file, namenode chooses the datanode which is closer to the same rack or nearby rack to Read/Write request. Namenode achieves rack information by maintaining the rack id's of each datanode. This concept that chooses closer datanodes based on the rack information 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. Default Hadoop installation assumes that all data nodes belong to the same rack.

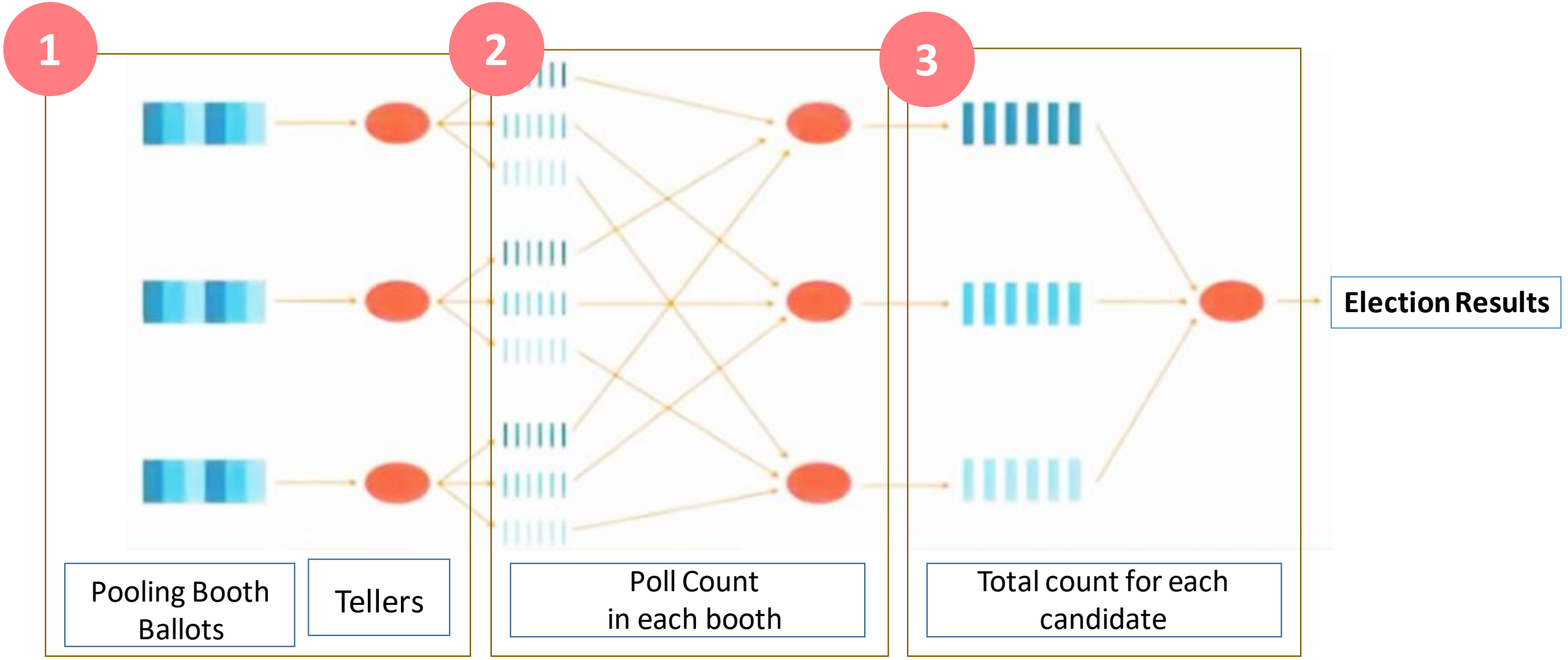
HDFS RACK AWARENESS

- To improve data high availability and reliability.
- Improve the performance of the cluster.
- To improve network bandwidth.
- Avoid losing data if entire rack fails though the chance of the rack failure is far less than that of node failure.
- To keep bulk data in the rack when possible.
- An assumption that in-rack id's higher bandwidth, lower latency.

BIG DATA

MAPREDUCE PROCESS

MAPREDUCE ANALOGY



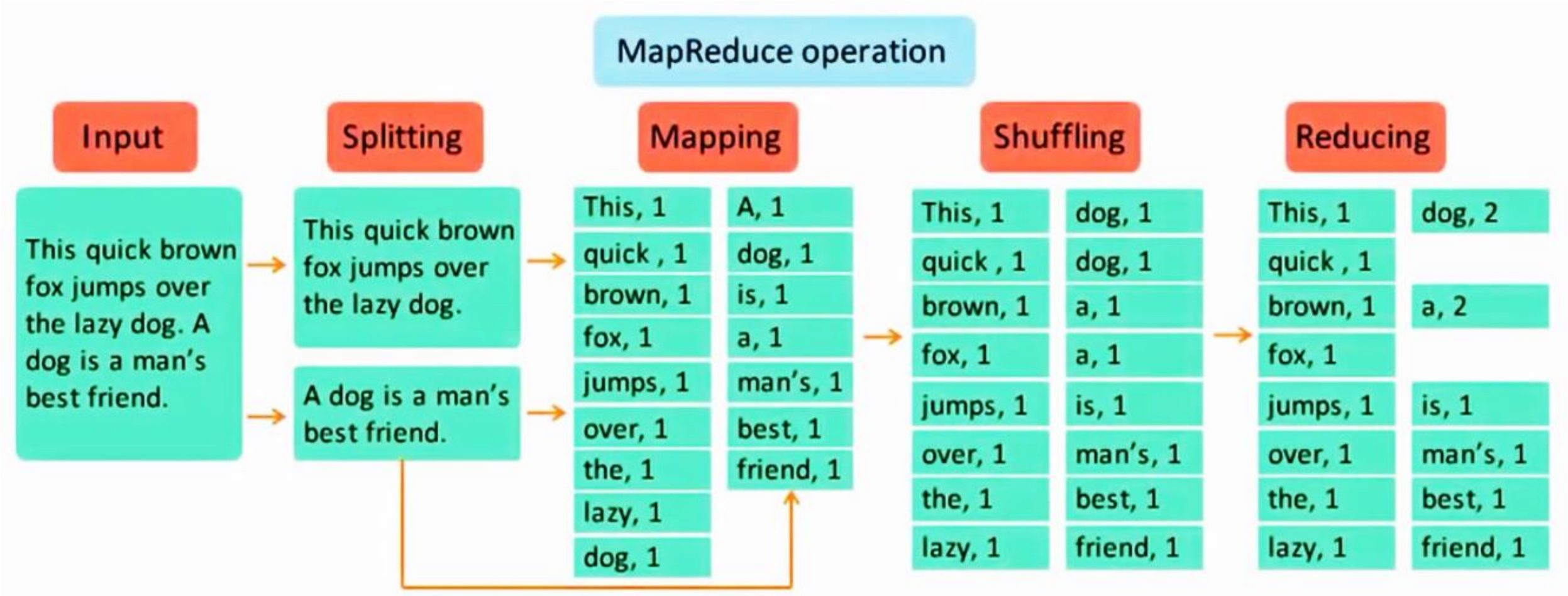
MAPREDUCE PARALLELISM

The Key Reason to perform mapping and reducing is to speed up the execution of a specific process by splitting a process into number of tasks, thus it encourages parallelism in job flow.

Individual Work

Parallel Work

MAPREDUCE ANALOGY



MAPREDUCE PERFORMANCE TUNING

Mapper task is the first phase of processing that processes each input record (from RecordReader) and generates an intermediate key-value pair. Hadoop Mapper store intermediate-output on the local disk.

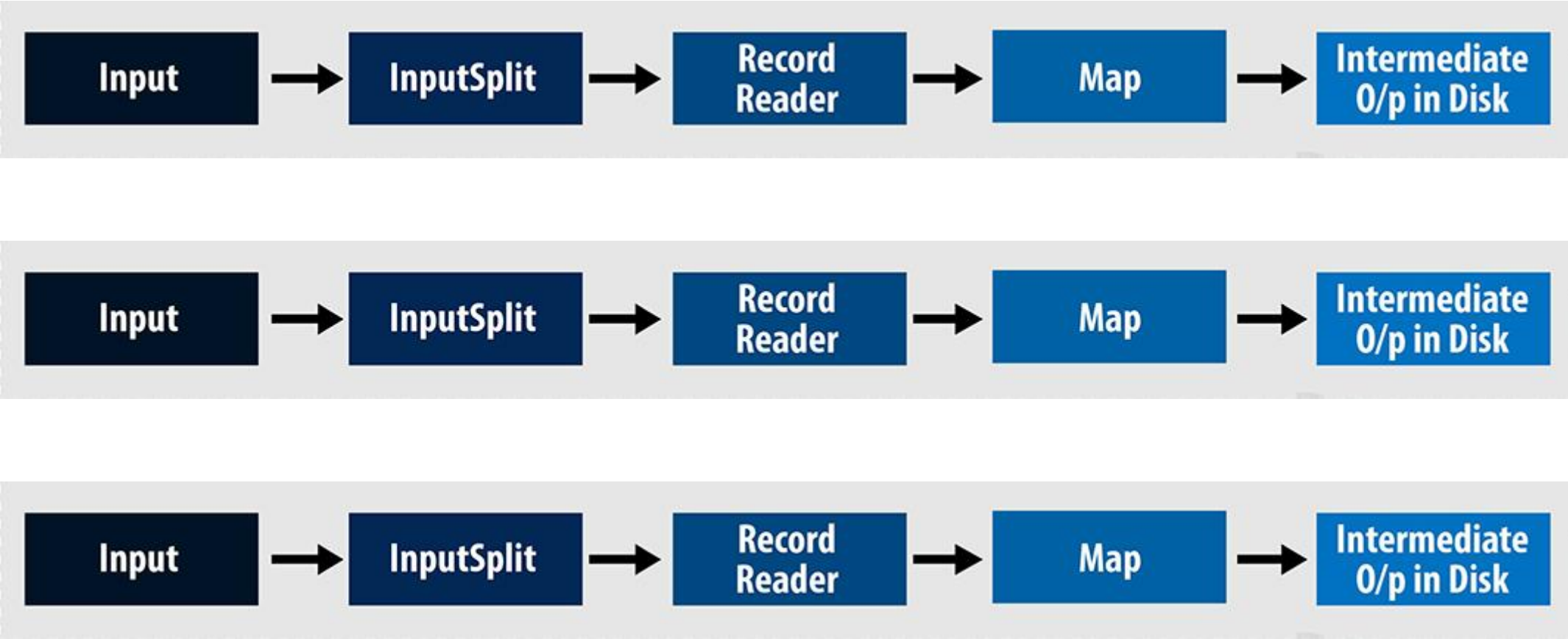
Hadoop Mapper task processes each input record and it generates a new <key, value> pairs. The <key, value> pairs can be completely different from the input pair. In mapper task, the output is the full collection of all these <key, value> pairs.

Key-Value PAIR GENERATION IN HADOOP

Input Split – It is the logical representation of data. It describes a unit of work that contains a single map task in a MapReduce program.

Record Reader – It communicates with the InputSplit and it converts the data into key-value pairs suitable for reading by the Mapper. By default, it uses TextInputFormat for converting data into the key-value pair. RecordReader communicates with the Inputsplit until the file reading is not completed.

INPUT SPLIT & RECORD READER

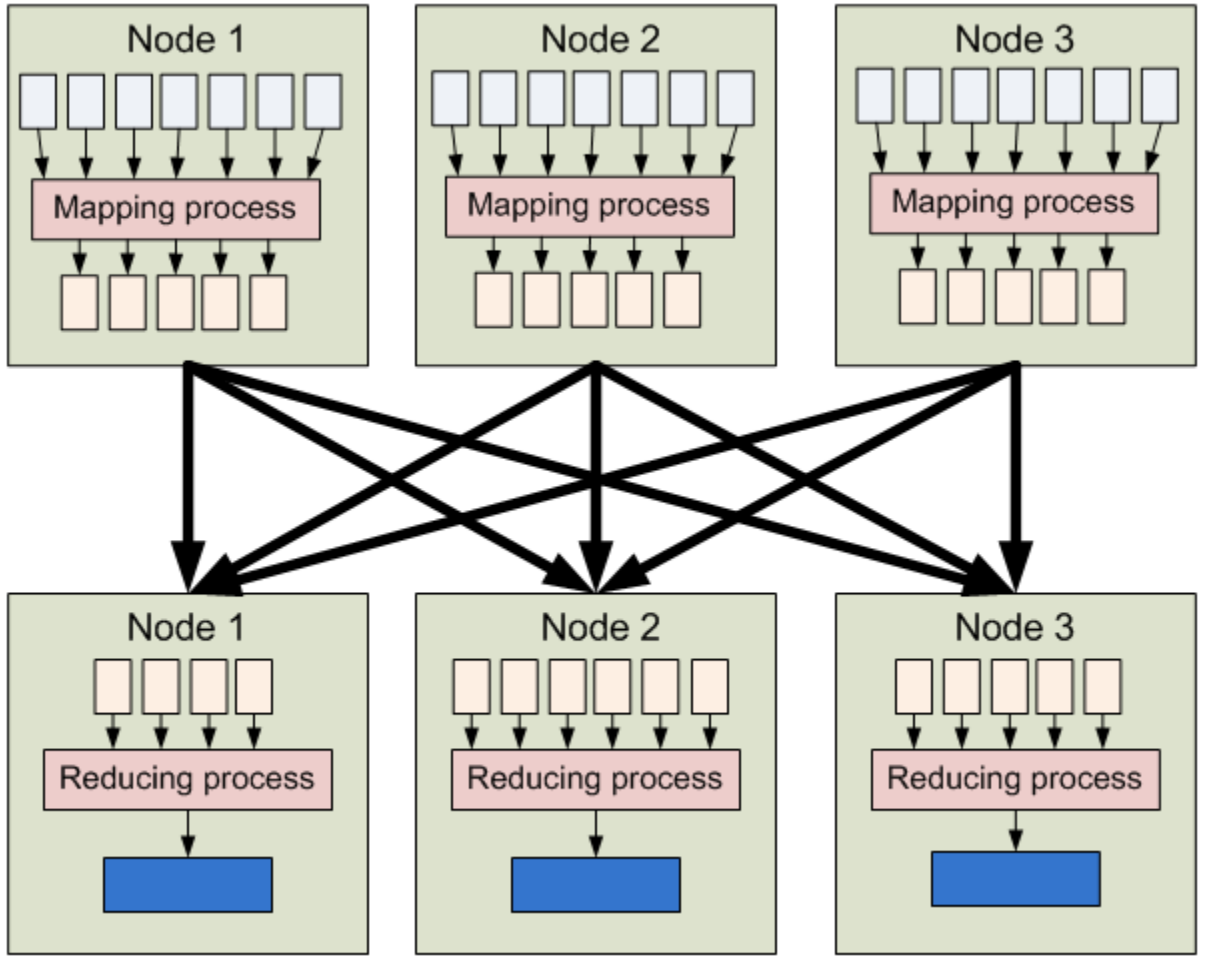


INPUT SPLIT & RECORD READER

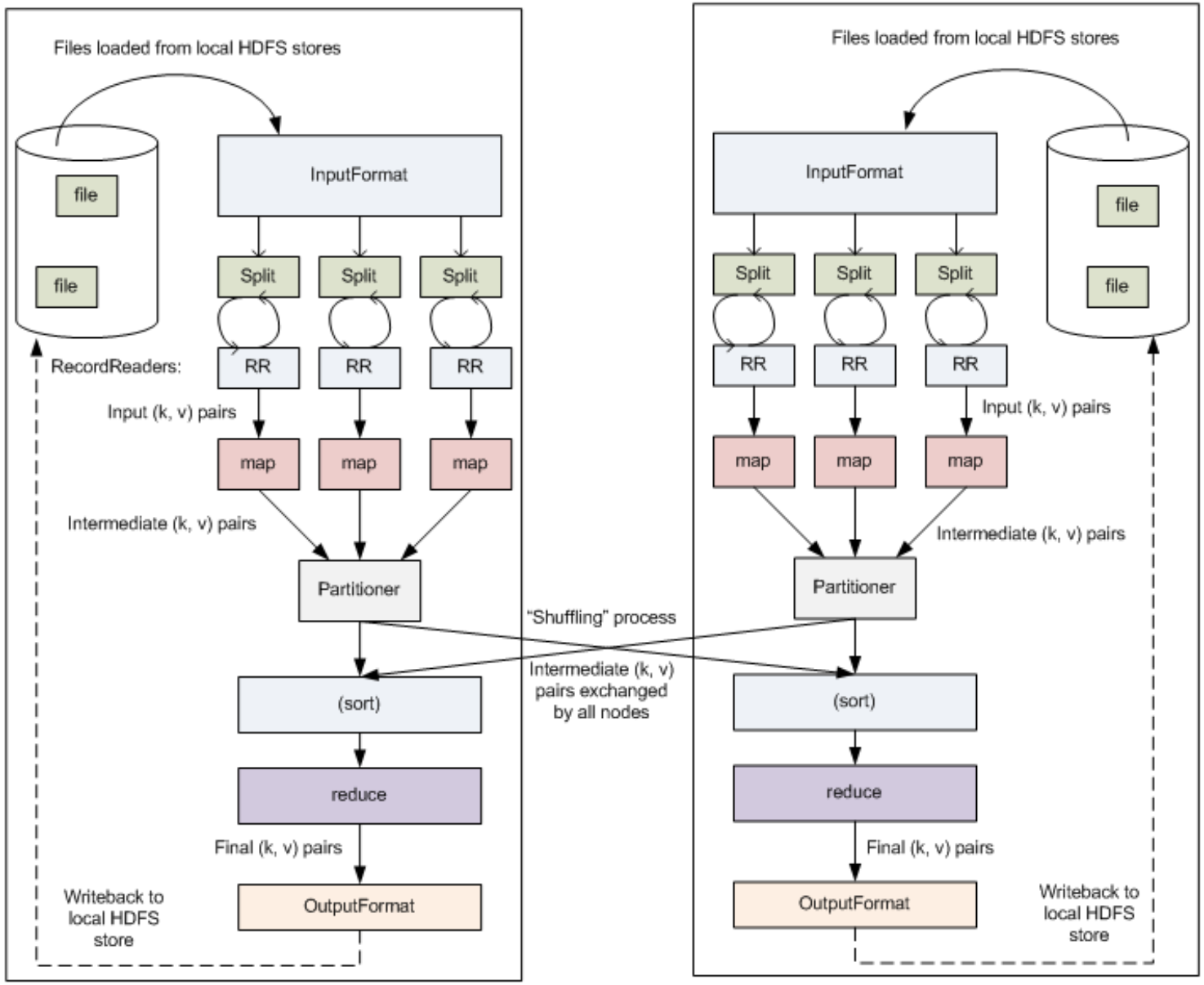
InputSpits do not always depend on the number of blocks, we can customize the number of splits for a particular file by setting ***mapred.max.split.size*** property during job execution.

RecordReader's responsibility is to keep reading/converting data into key-value pairs until the end of the file. Byte offset (unique number) is assigned to each line present in the file by RecordReader. Further, this key-value pair is sent to the mapper. The output of the mapper program is called as intermediate data

MAPREDUCE PROCESS



MAPREDUCE PROCESS CLOSER LOOK



DISABLE REDUCER

We can achieve this by setting ***job.setNumreduceTasks(0)*** in the configuration in a driver. This will make a number of reducer as 0 and thus the only mapper will be doing the complete task.

Advantages of Map only job in Hadoop:

In between map and reduces phases there is key, sort and shuffle phase. Sort and shuffle are responsible for sorting the keys in ascending order and then grouping values based on same keys. This phase is very expensive and if reduce phase is not required we should avoid it, as avoiding reduce phase would eliminate sort and shuffle phase as well. This also saves network congestion as in shuffling, an output of mapper travels to reducer and when data size is huge, large data needs to travel to the reducer.

MODIFY HDFS Block Size

To configure the data block at Cluster level we need to specify in the **hdfs.site.xml**, eg. For 128 mb,value will be 64*1024*1024

<property>
<name>dfs.block.size</name>
<value>134217728</value>
<description>Block size</description>
</property>

For multinode, in a Cluster. we need to update the same in the node(Name Node and Data Node) and restart the daemons.

This change doesn't affect the existing files in Hadoop HDFS.

To change a block size for specific file in a cluster:
hadoop fs -Ddfs.blocksize=134217728 -put /home/hduser/test/test.text /hdfs

ftp://ftp.ncdc.noaa.gov/pub/data/uscrn/products/daily01

HAPPY LEARNING