Big Data + Hadoop Full Course

(Telugu)

Hadoop Full Course

(Telugu)

AGENDA

01 Big Data

02 Hadoop

Introduction to Big data

Introduction to Big data

Big Data

- 01 What is Big Data?
- 02 Characteristics of Big Data
- 03 Types of Big Data
- 04 Tools and Applications of Big Data
- 05 Big Data Analytics & Stages

How Big Data came?

What is Data?

Data means simply we can say that Information

What is Big Data?

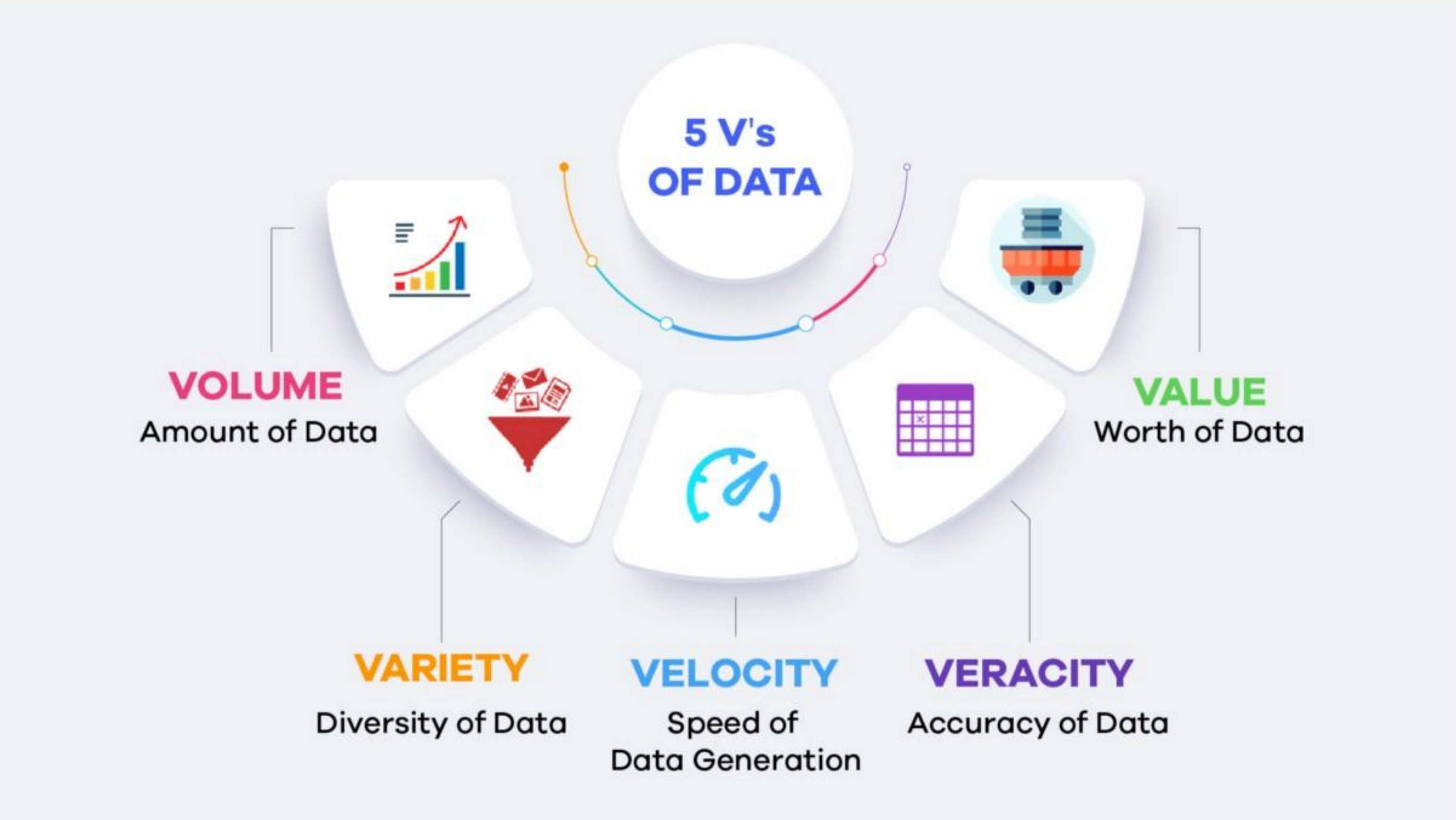
Big Data is also data but with a huge size. Big Data is a term used to describe a collection of data that is huge in volume and yet growing exponentially with time.


BIG
DATA
STRUCTURED
UNSTRUCTURED
SEMI-STRUCTURED

Types of Big Data

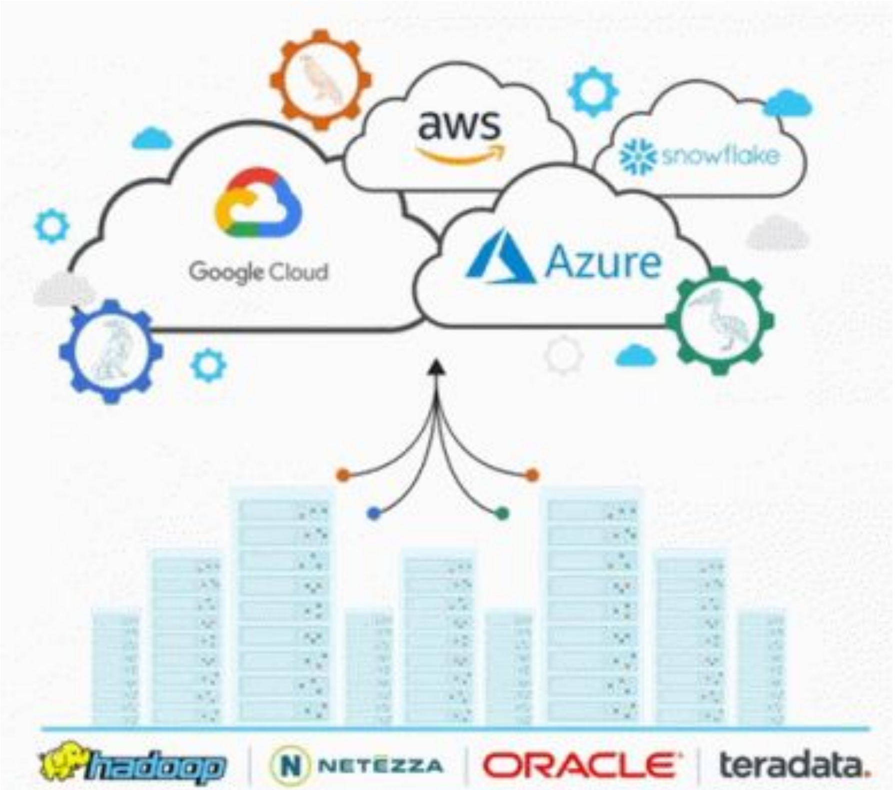
Structured	• Organized Data • Ex: Excel Sheets, Database,
Unstructured	• Unorganized Data • Comments, Videos, Posts
Semi Structured	• Structured + Unstructured Data • Json Files, XML files

Characteristics Big Data



Tools in Big Data

1. Apache Spark
2. Apache Hadoop
3. Apache Flink
4. Google Cloud Platform
5. MongoDB
6. Sisense
7. RapidMiner



Applications of Big Data

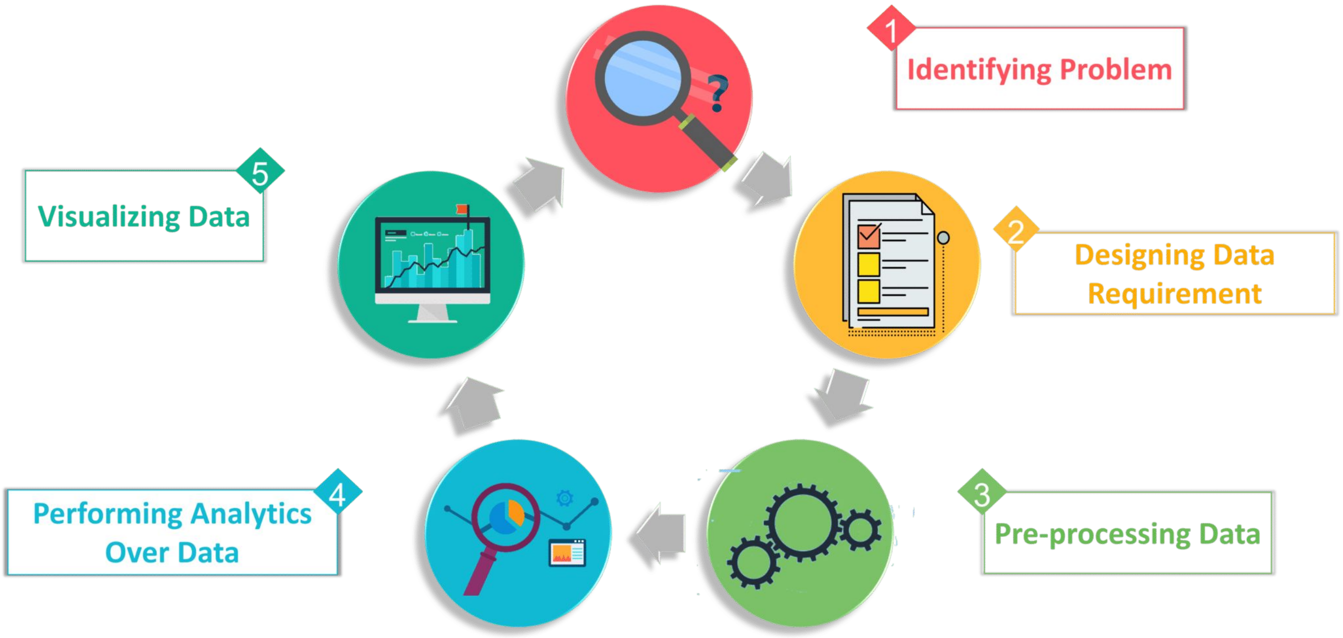
1. Banking and Securities
2. Communications, Media and Entertainment
3. Healthcare Providers
4. Education
5. Manufacturing and Natural Resources
6. Government
7. Insurance
8. Retail and Wholesale trade
9. Transportation
10. Energy and Utilities

Big Data Analytics

- Fraud Management Report, used in Banking Sectors to find the fraud transactions, hacking, unauthorized access to the account and so on.
- Live Tracking Report which is generally used by Transport Sectors such as Meru, Ola, Uber, and Mega to track the vehicles, customer's requests, payment management, emergency alert and to find the daily needs and revenues and so on.

Why Big Data Analytics

Stages Big Data Analytics



Introduction to Hadoop

Introduction to Hadoop

Hadoop

01 How Hadoop Started?

02 How Big Data Similar to Hadoop

03 How Hadoop Solving Big Data Problems

04 Understanding Hadoop in Detail

05 Features of Hadoop

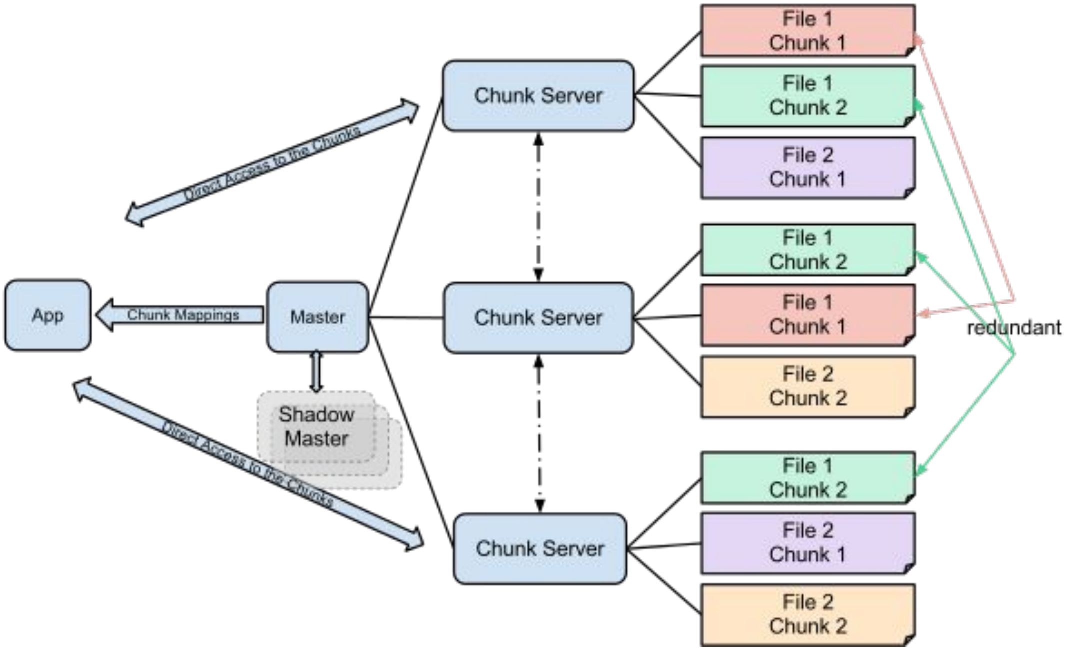
How Hadoop Started?

Mike Cafarella & Doug Cutting want to build search engine system

To build That Search engine with 1 billion pages it may cost $35000/month

How Hadoop Started?

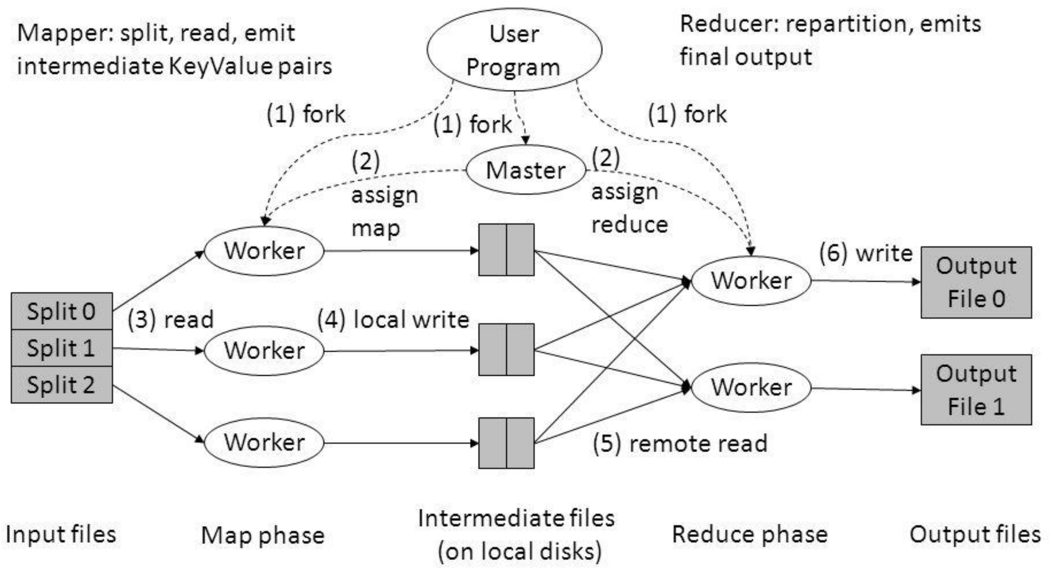
In 2003 GFS(Google Distributed File System) Introduced



How Hadoop Started?

In 2004 Google again Introduced MapReduce

Google MapReduce



How Hadoop Started?

HADOOP = GFS + MapReduce

$\downarrow$

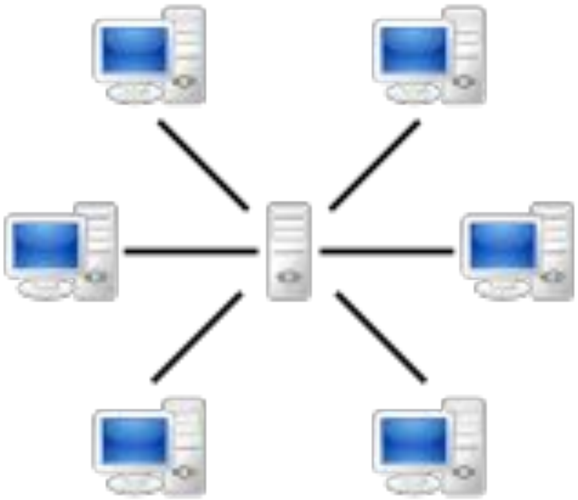
HADOOP = HDFS + MapReduce

How Big Data Similar to Hadoop?

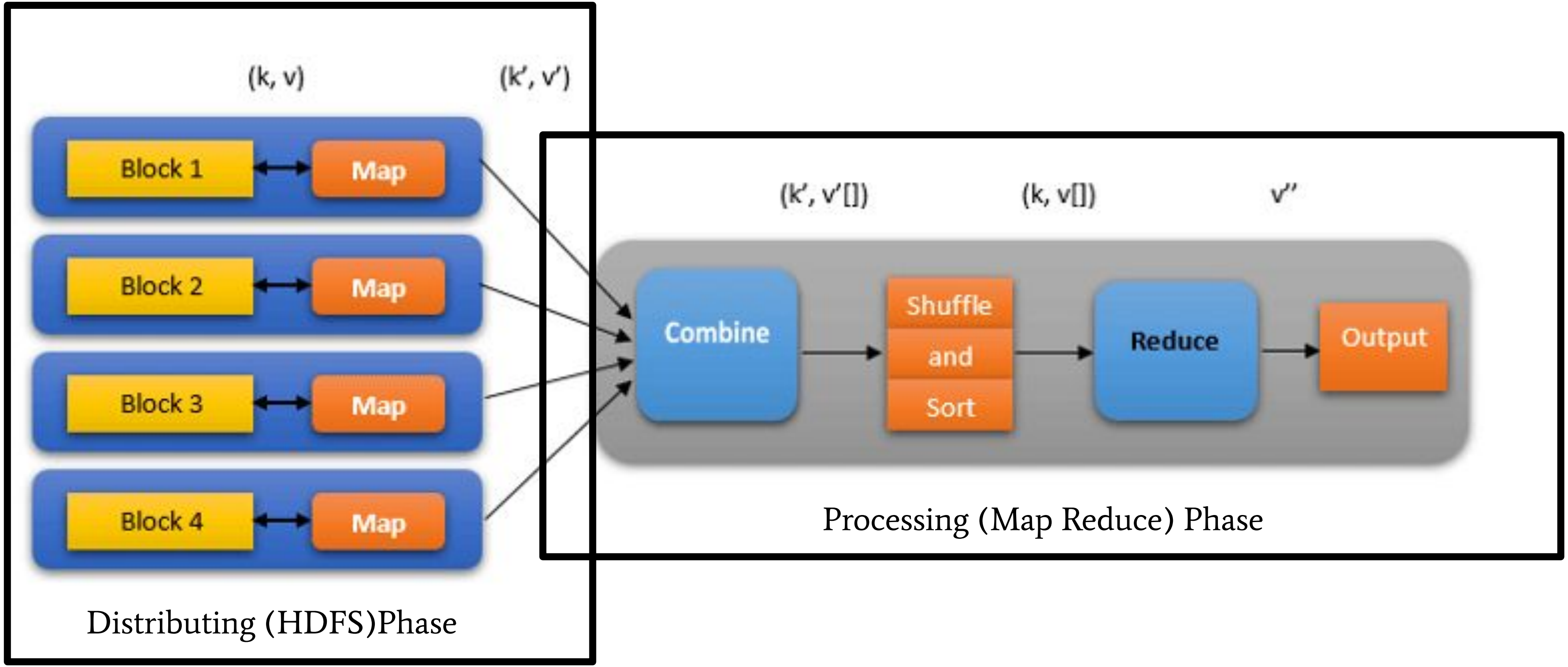
Traditional DB cannot handle large data

For Large Data Hadoop using distributed Processing

Multiple processing units were installed to process data in parallel



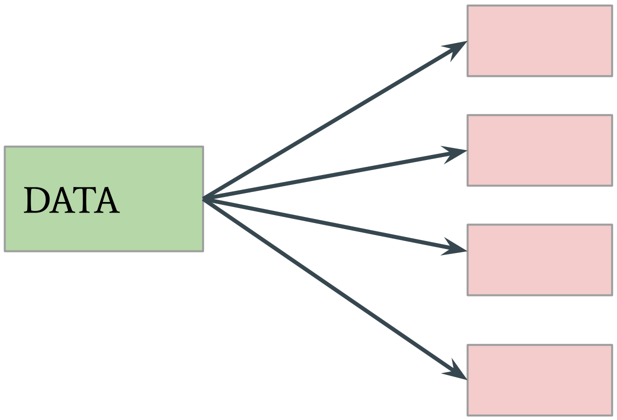
How MapReduce used?



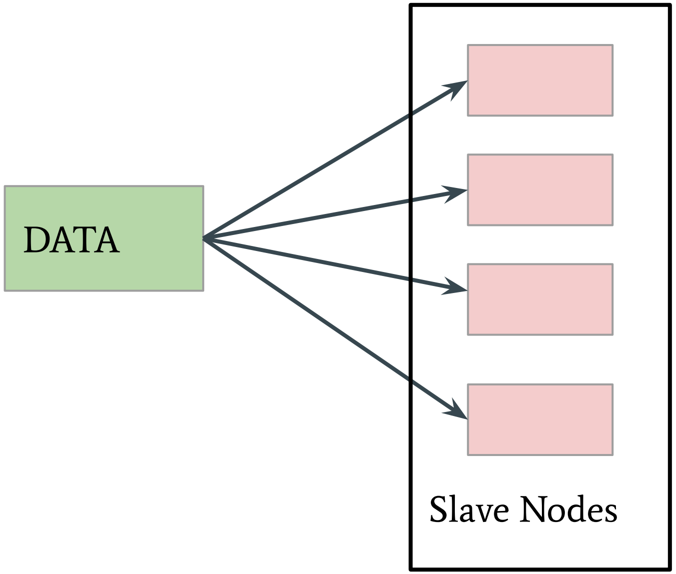
Parallel Processing

DATA

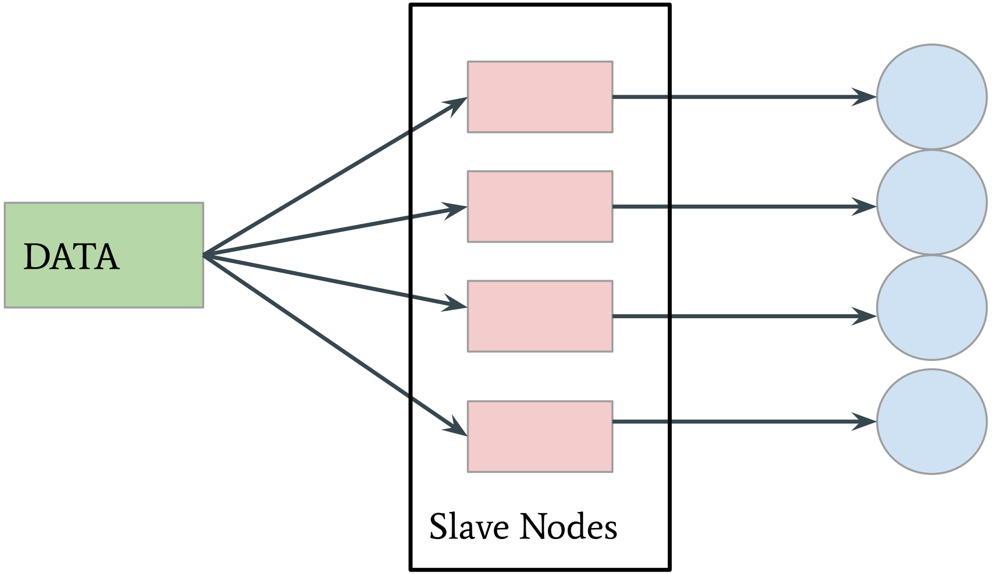
Parallel Processing



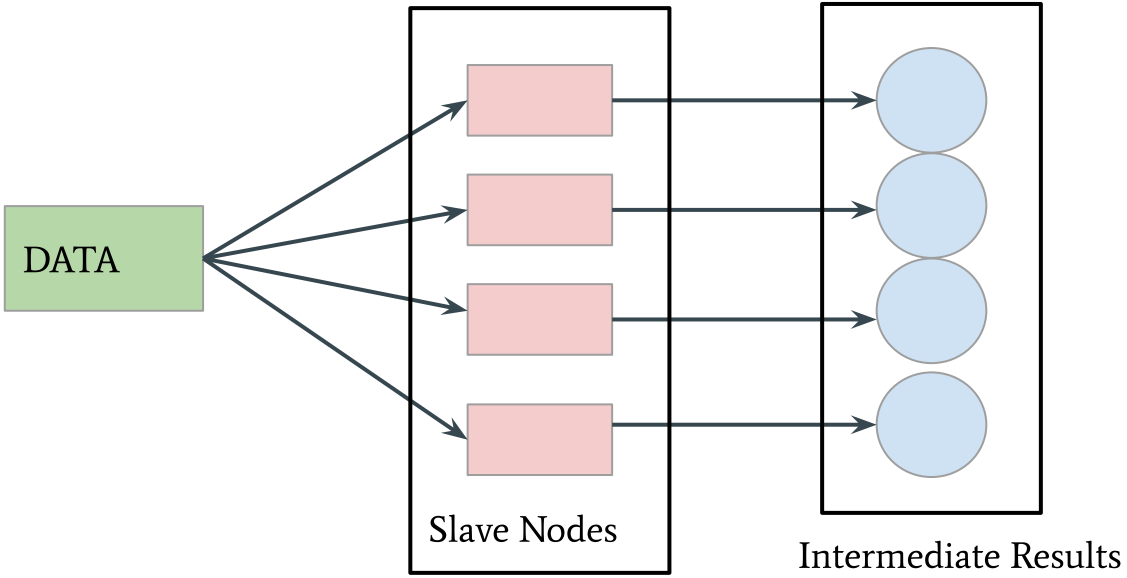
Parallel Processing



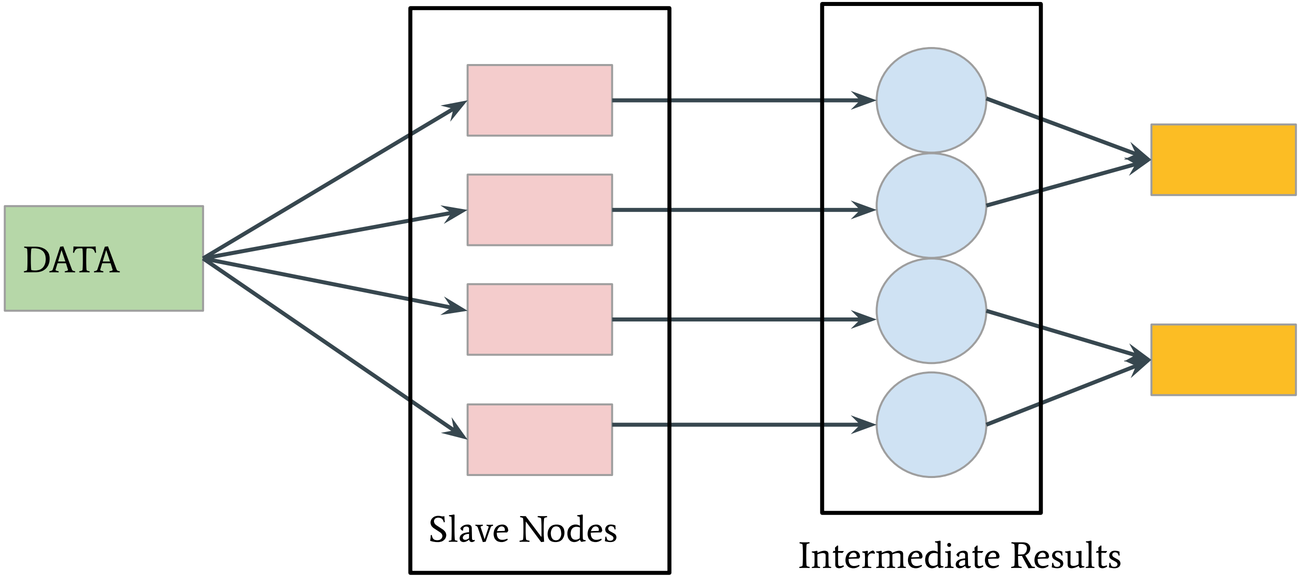
Parallel Processing



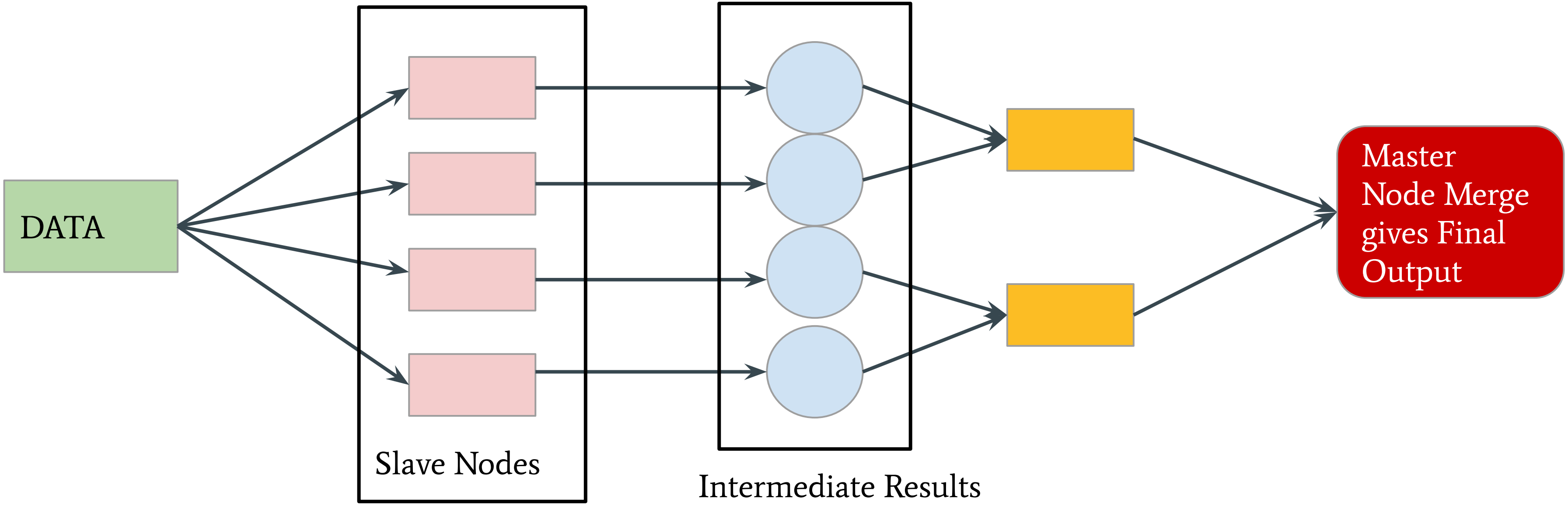
Parallel Processing



Parallel Processing

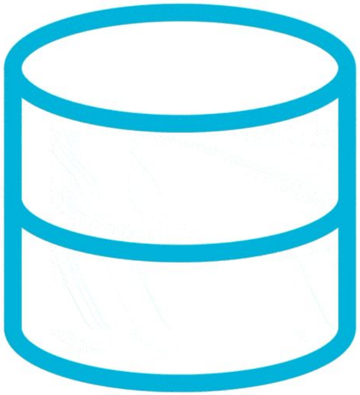


Parallel Processing



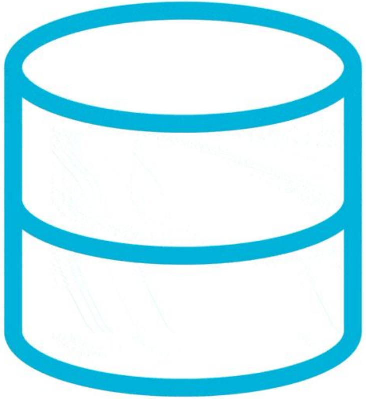
Big Data Problems

Problem 1: Storing Large amount of Data



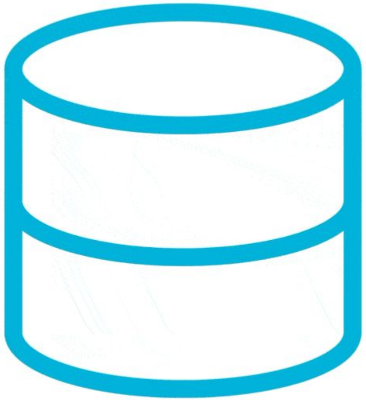
Big Data Problems

Problem 2: Storing Different Types of Data



Big Data Problems

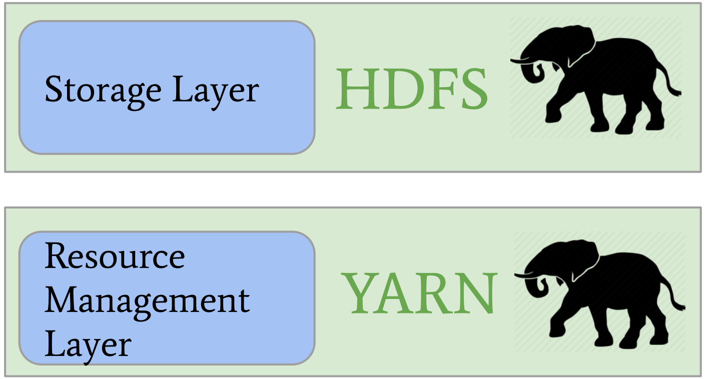
Problem 3: To process high amount of Data speed will decrease



How Hadoop Introduced

To Solve Storage and processing issues Hadoop created to components HDFS, YARN

HDFS- Store Data in Distributed Manner
YARN-Solve Processing issue by reducing process time



What is Hadoop Exactly?

- Open Source Software Framework used for storing and processing big data.
- Hadoop Written in JAVA
- Hadoop is one of the top project in APACHE Projects

How Hadoop Solve Big Data Problems?

Problem 1: Storing Large amount of Data , Hadoop used HDFS

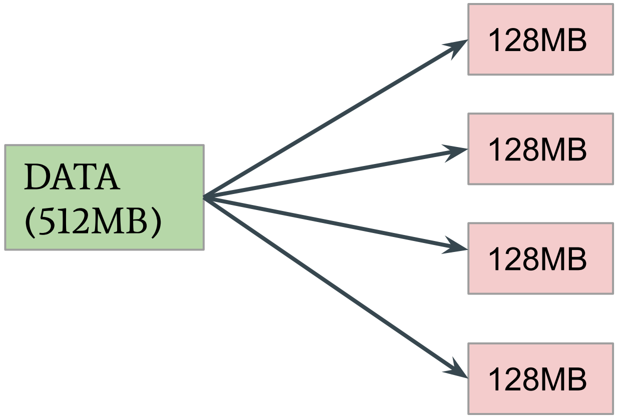
HDFS- provides Distributed way to store huge data

DATA
(512MB)

How Hadoop Solve Big Data Problems?

Problem 1: Storing Large amount of Data , Hadoop used HDFS

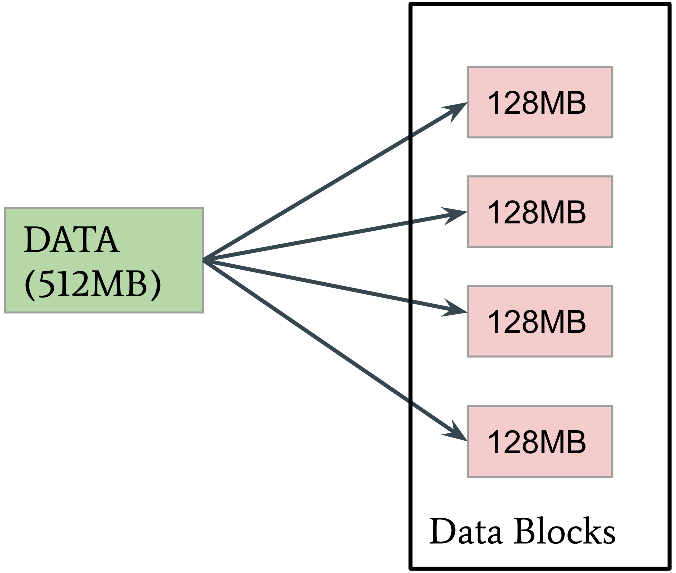
HDFS- provides Distributed way to store huge data



How Hadoop Solve Big Data Problems?

Problem 1: Storing Large amount of Data , Hadoop used HDFS

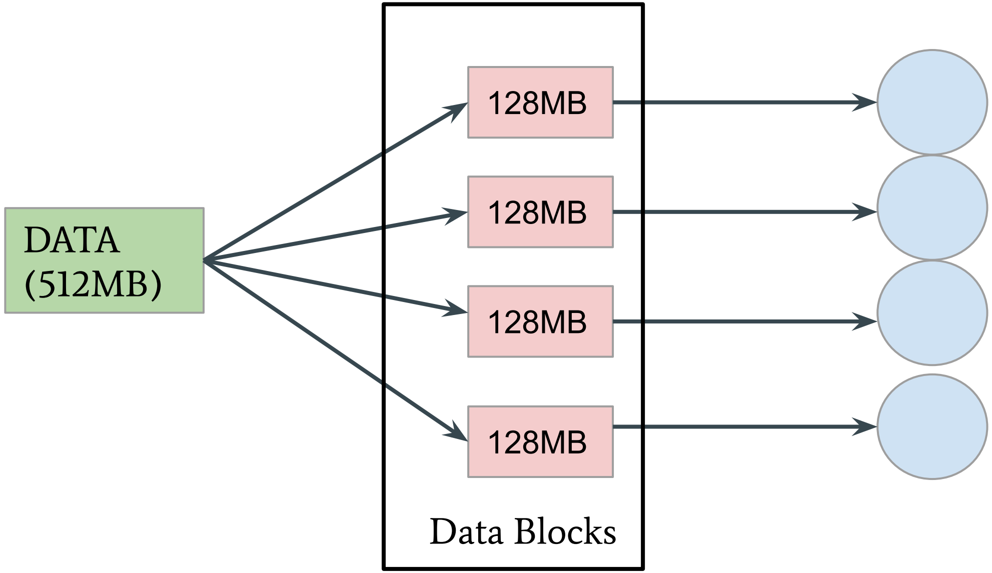
HDFS- provides Distributed way to store huge data



How Hadoop Solve Big Data Problems?

Problem 1: Storing Large amount of Data , Hadoop used HDFS

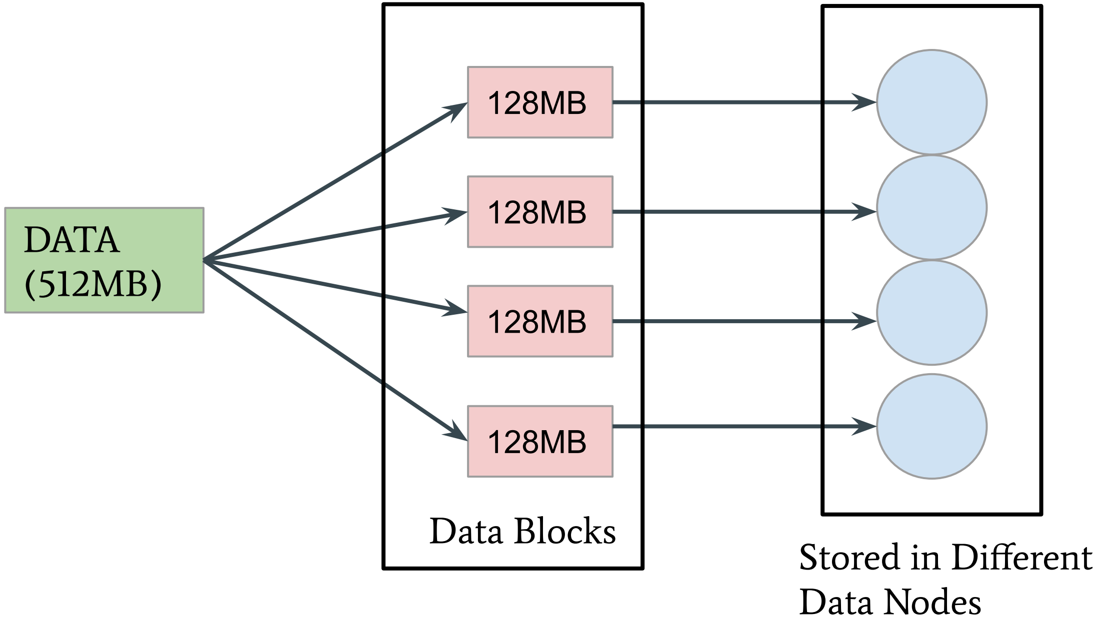
HDFS- provides Distributed way to store huge data



How Hadoop Solve Big Data Problems?

Problem 1: Storing Large amount of Data , Hadoop used HDFS

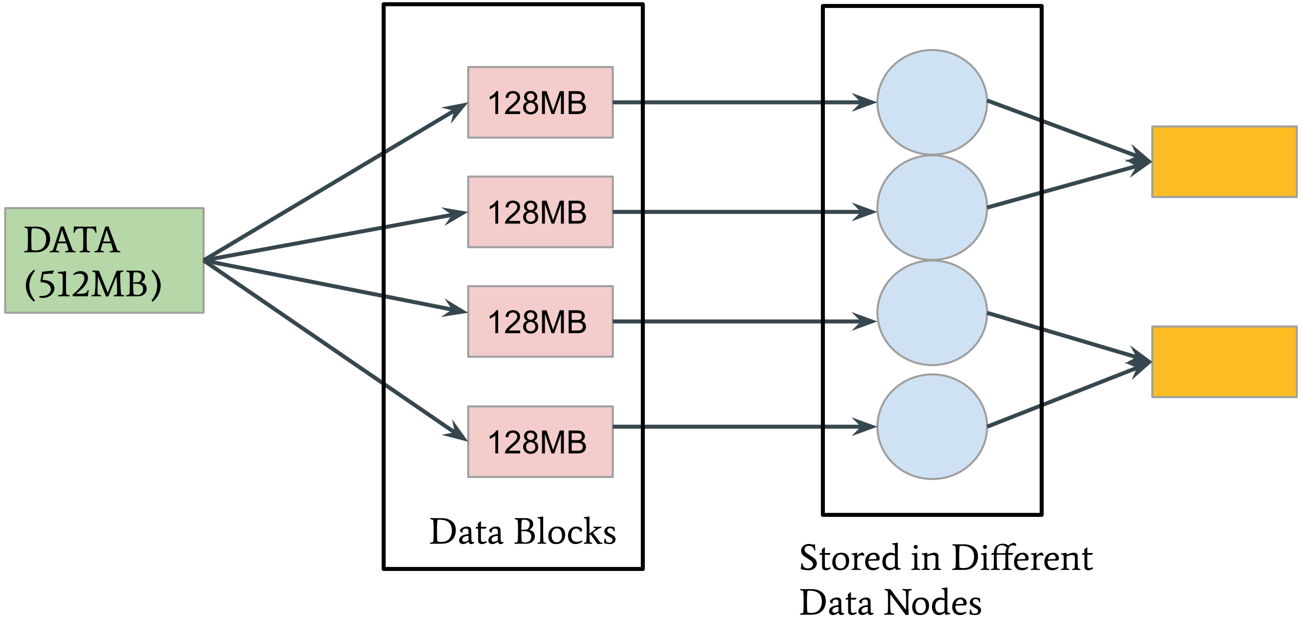
HDFS- provides Distributed way to store huge data



How Hadoop Solve Big Data Problems?

Problem 1: Storing Large amount of Data , Hadoop used HDFS

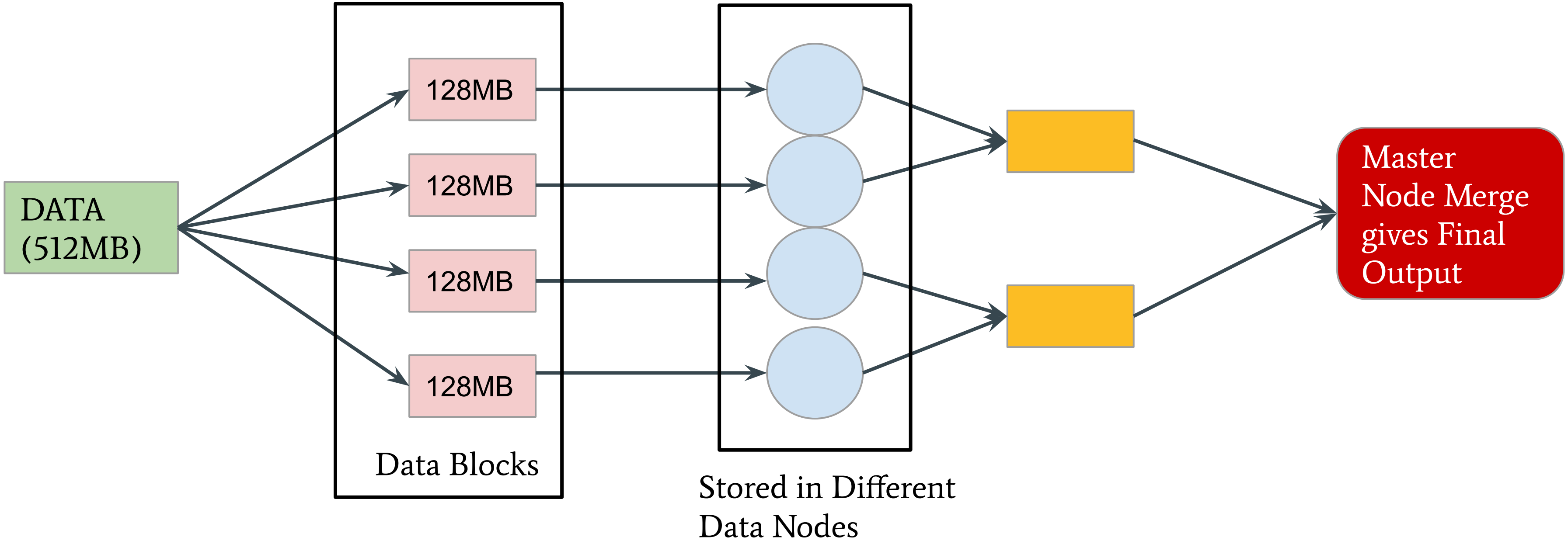
HDFS- provides Distributed way to store huge data



How Hadoop Solve Big Data Problems?

Problem 1: Storing Large amount of Data , Hadoop used HDFS

HDFS- provides Distributed way to store huge data



How Hadoop Solve Big Data Problems?

Hadoop Follows Horizontal Scaling

Horizontal Scaling:
You can add new nodes to HDFS Cluster as per requirement, instead of increasing Hardware Stack present each node.

Hardware Stack:
Directly connecting switches to form a large system.

How Hadoop Solve Big Data Problems?

Problem 2: Variety of Data
In HDFS, we can store all kind of data.
In HDFS there is no Pre-dumping schema validations

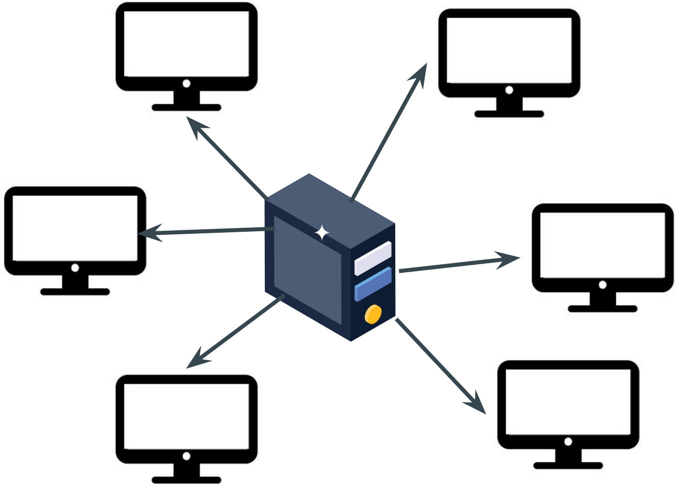
Pre-dumping schema validations:
Data once written can be read multiple times without any problem

HDFS follow write once read many modules, So due to this you can write once and read it multiple times.

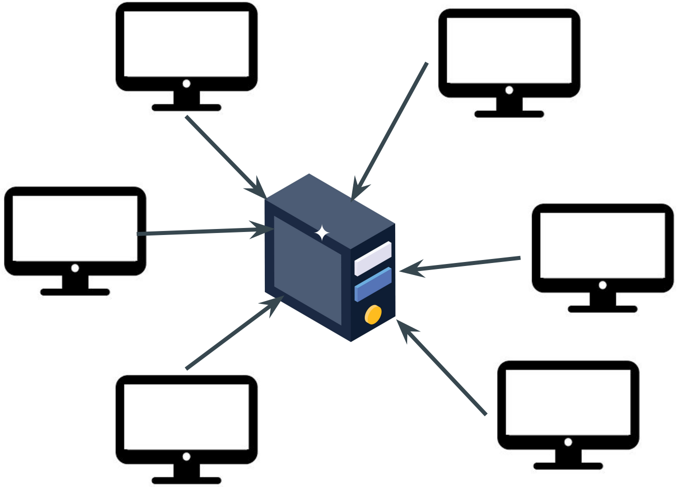
How Hadoop Solve Big Data Problems?

Problem 3: Processing Data Faster.

Traditional Approach



Hadoop Approach



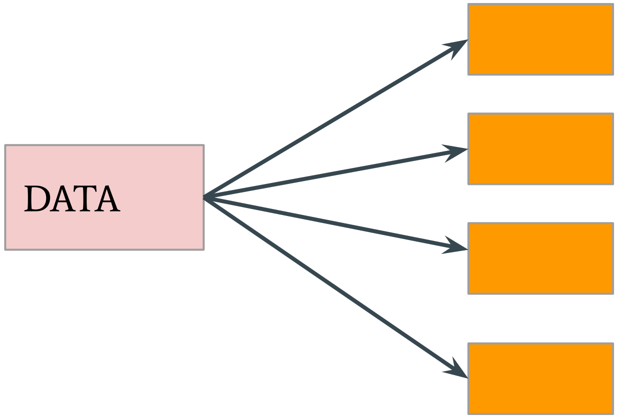
Features of Hadoop

1. **Reliability:**

DATA

Features of Hadoop

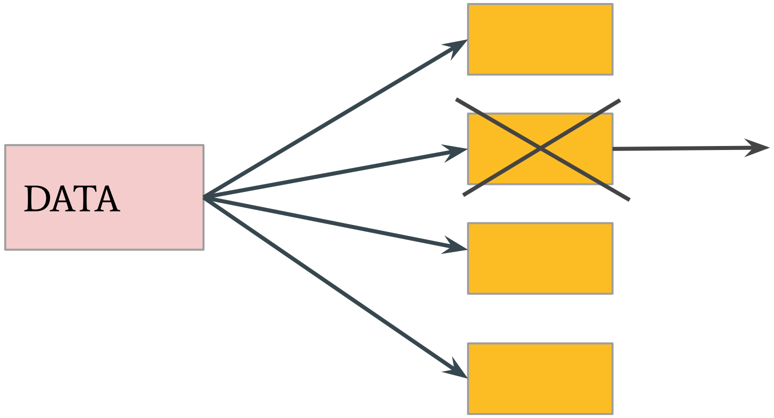
1. **Reliability:**



Different
single units

Features of Hadoop

1. Reliability:



If it fails remaining node can take responsibility

Different
single units

Features of Hadoop

2. Economical:

Hadoop uses commodity hardware [PC, Laptop]

Small Hadoop Cluster contains
8-16 GB RAM
5-10 TB Hard Disk and Xeon processors

Hadoop is cheaper comparing to other and it is open source.

Features of Hadoop

3. Scalability

Hadoop has inbuilt capability of cloud based services

If we install hadoop on cloud no need to worry we can expand within minutes whenever required.

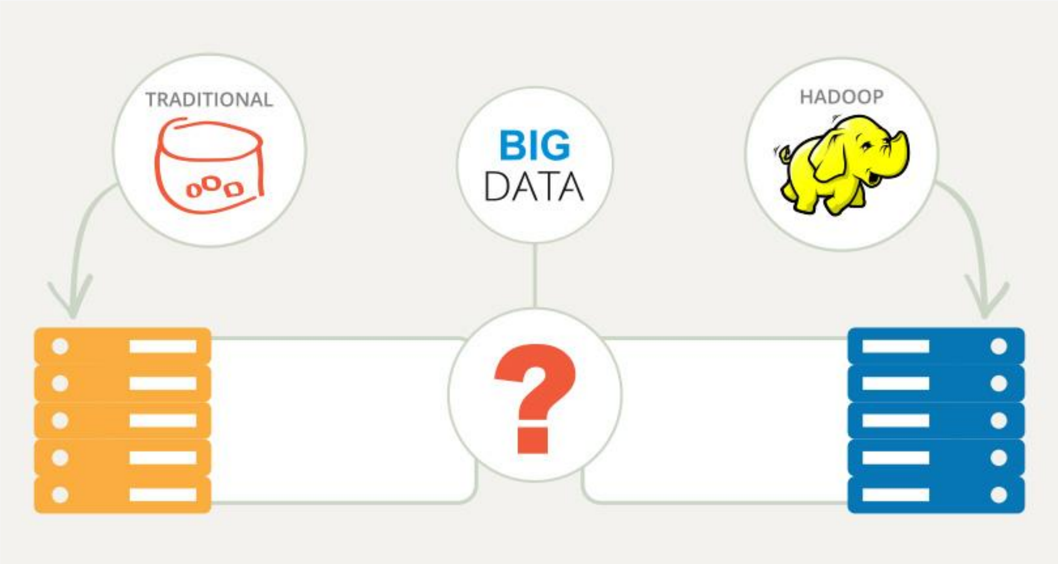
Features of Hadoop

4. Flexibility

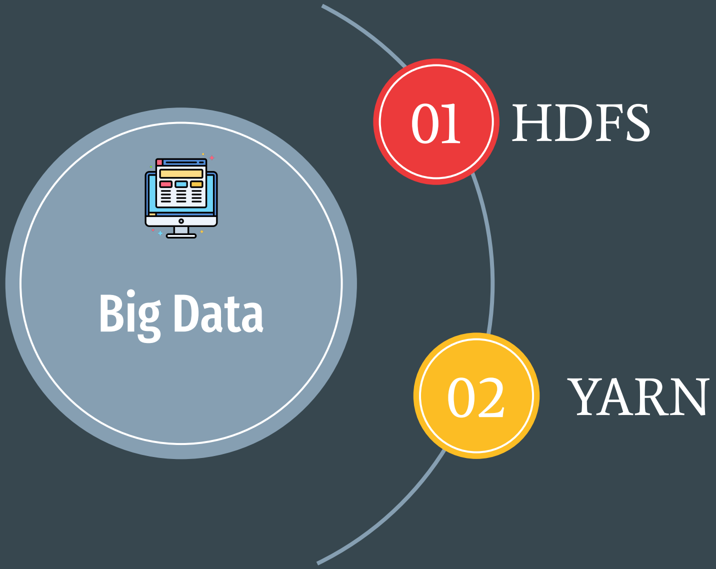
Hadoop is very flexible.
Dealing with all kinds of Data.
Hadoop process any kind of data.

How Hadoop Improves on Traditional Databases

1. Capacity: Hadoop stores large volumes of data.
2. Speed: Hadoop stores and retrieves data faster.



Hadoop Core Components



hadoop

HDFS

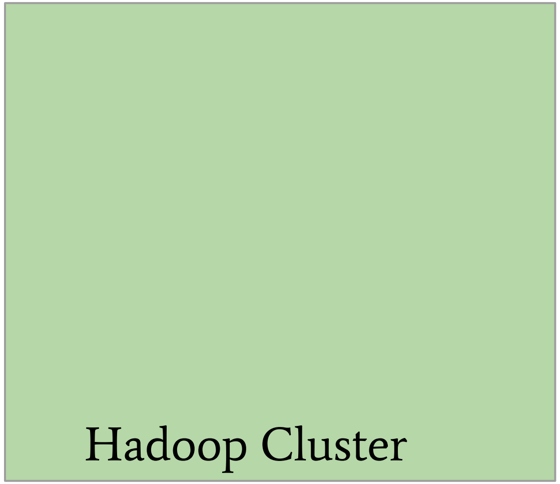
HDFS

Distributed File System:
Managing Data [Files, Folders across multiple servers, Computers]

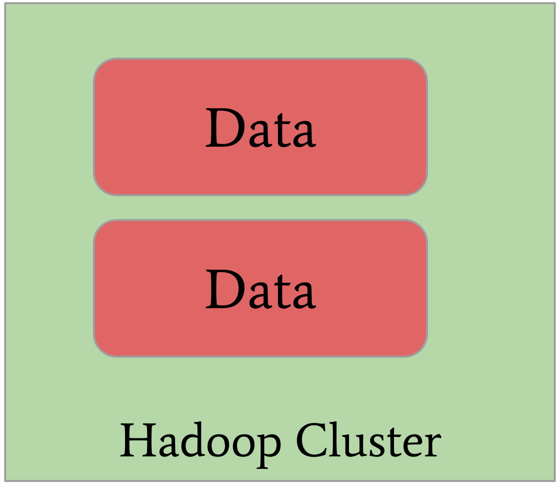
DFS allow us to store data in multiple nodes or multiple machines in a cluster, multiple users can access data.

Ex: Works like File System
Windows—> NTFS [New Technology File System]
Mac —>HFS [Hierarchical File System]
NTFS, HFS Stores data in single machine
DFS Stores data in multiple machines

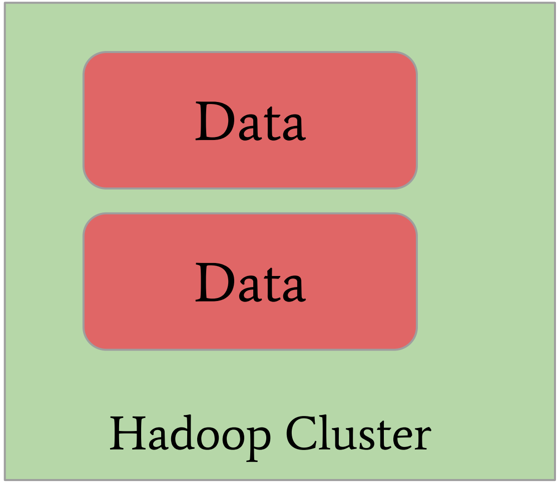
HDFS

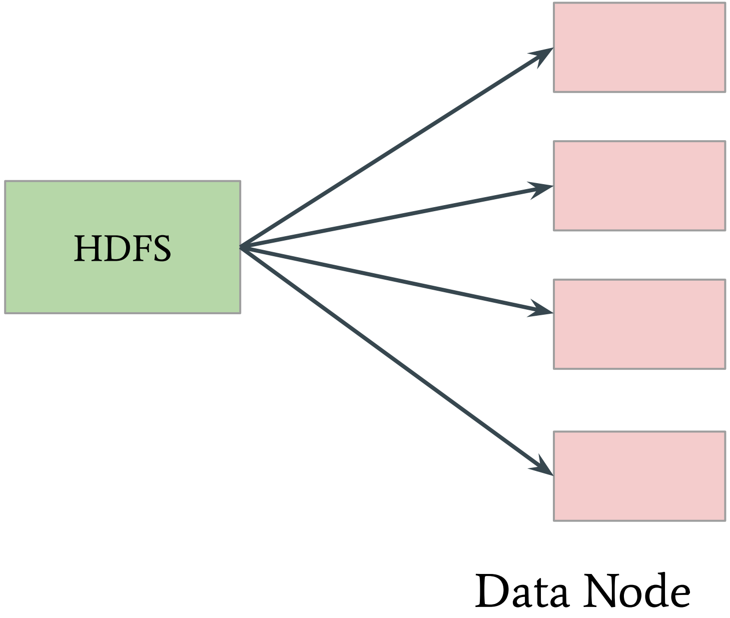


HDFS



HDFS



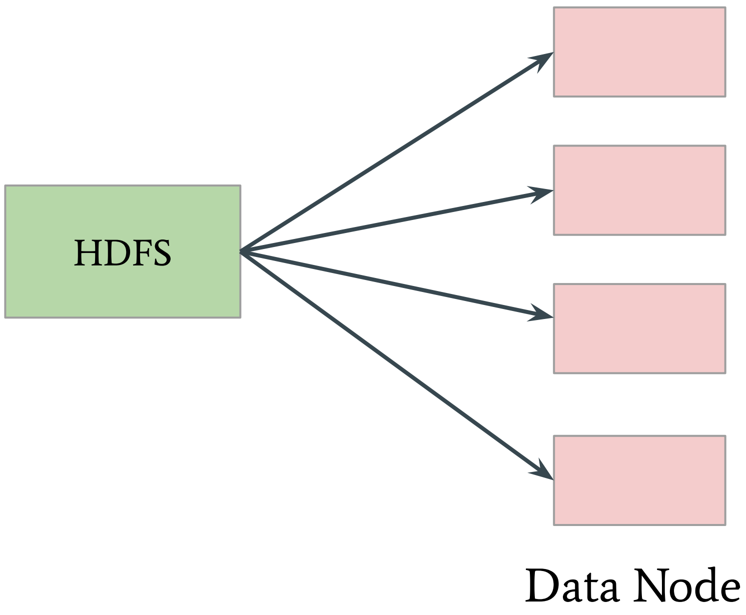


HDFS

Advantages of Hadoop:

1. **Distributed Storage:**

Example:
10TB data stored in 10 machines
But it looks like all 10 TB data in Single machine



HDFS

Advantages of Hadoop:

2. Distributed & Parallel Computation:

Example:

30 mins to process 1TB data in machine

Taking 1 TB Data and processing in 10 machines

1 Machine -30 mins

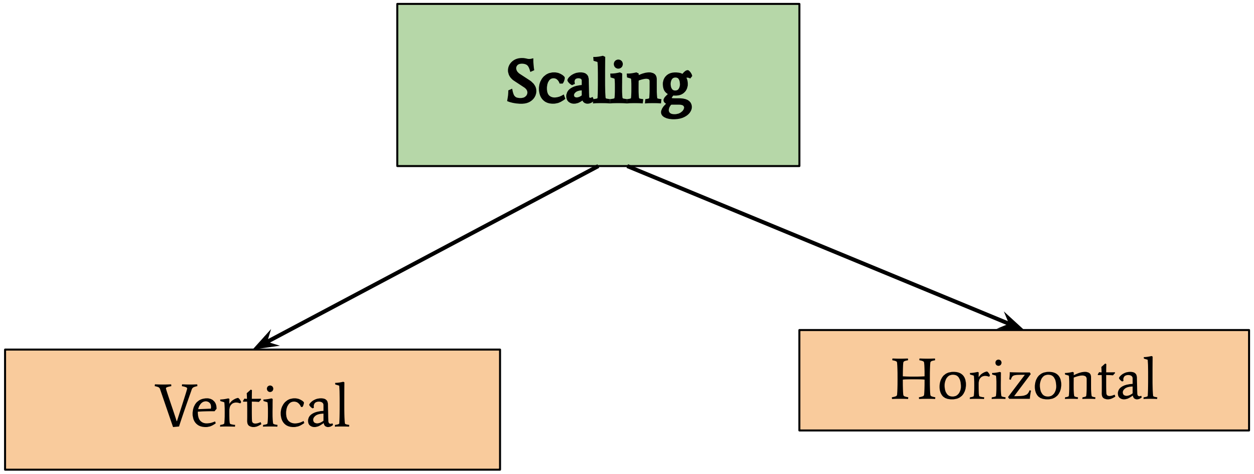
10 machines-? -----> 3 mins

Data Divided and processed over 10 machines parallely.

HDFS

Advantages of Hadoop:
3. Horizontal Scalability:

Vertical , Horizontal



HDFS

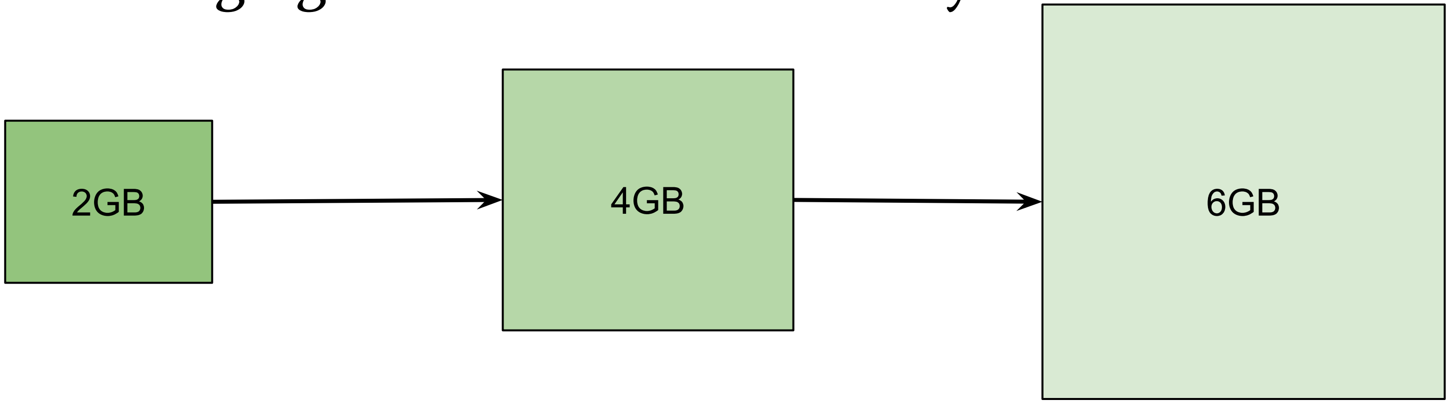
Advantages of Hadoop:

Vertical Scalability[Scale up]:

Increase hardware capacity of system

There is a limit to increase hardware capacity

After changing size we have to restart system



HDFS

Advantages of Hadoop:

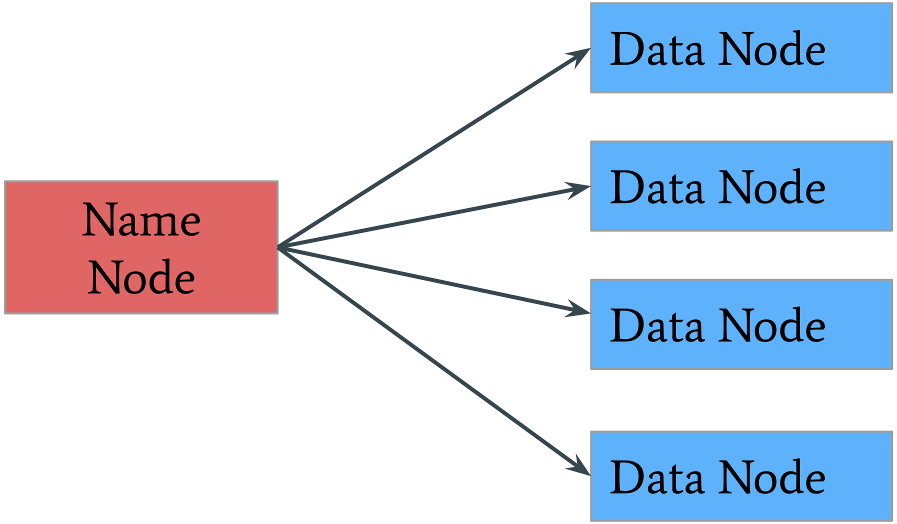
Horizontal Scalability:

Add more nodes to existing cluster instead of increasing hardware capacity

We can add more machines without stopping them

At last all machines run parallel as per requirement

HDFS Architecture



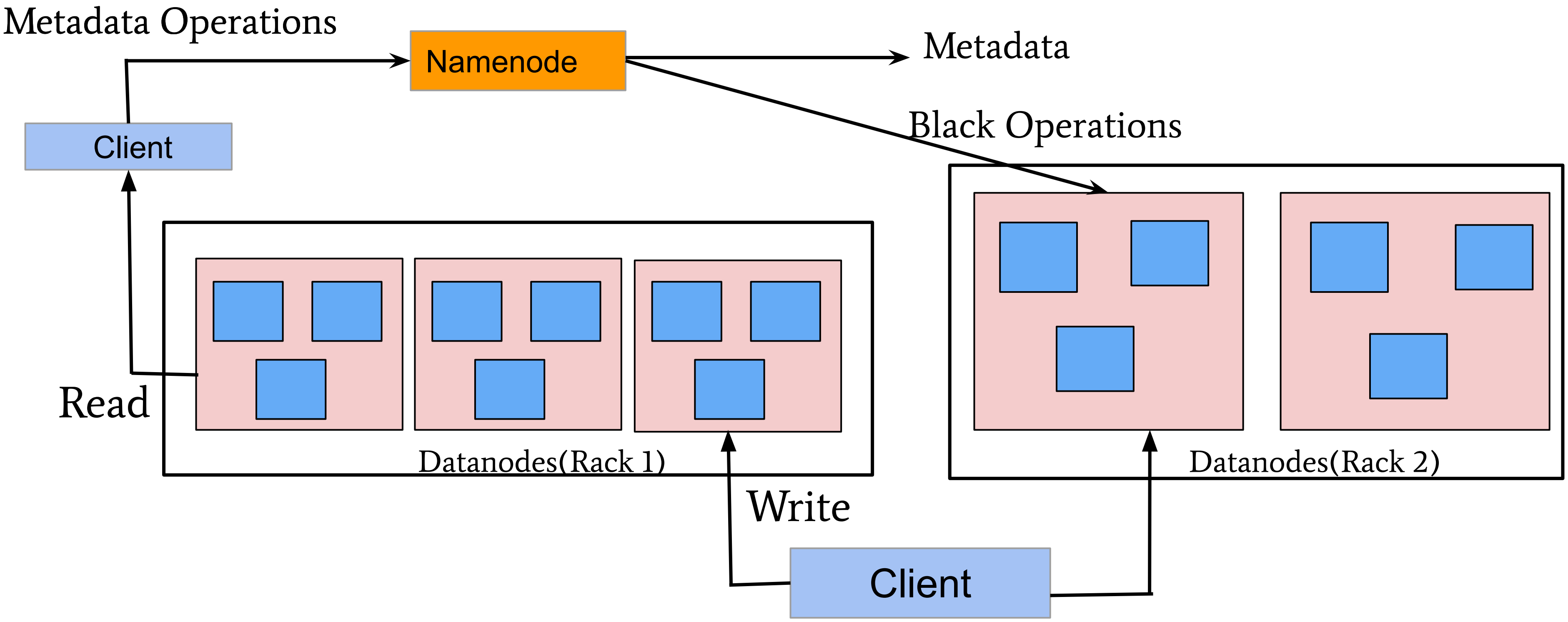
HDFS is Block Structured File System
Each file divided into particular size

These blocks stored across cluster one or more machines

Follow Master/ Slave Architecture
Master- Namenode
Slave - Multiple Datanodes

Datanodes can run on single machine or several machines

HDFS Architecture



HDFS Architecture

Functions of Namenode:

1. Maintain & manages Data Nodes
2. Records Metadata

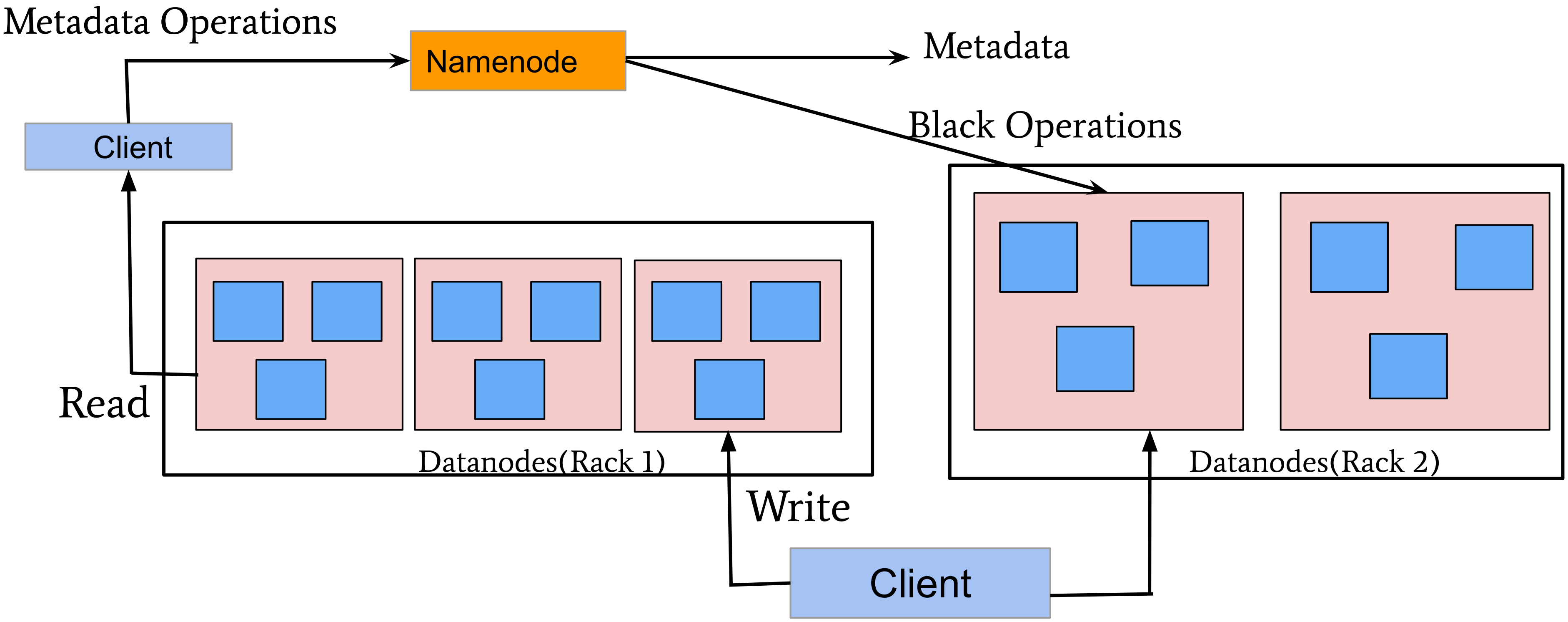
   Metadata contains 2 components

   FsImage: Complete state of file system stores from start

   Edit logs: All recent modifications with respect to FsImage
3. Records all Changes
4. Receives regular report of all datanodes, blocks
5. Also responsible for replication factor.
6. If Datanode fails Namenode Choose new datanode for new replicas

HDFS Architecture

Data Node



HDFS Architecture

Datanode:

Slave nodes in HDFS

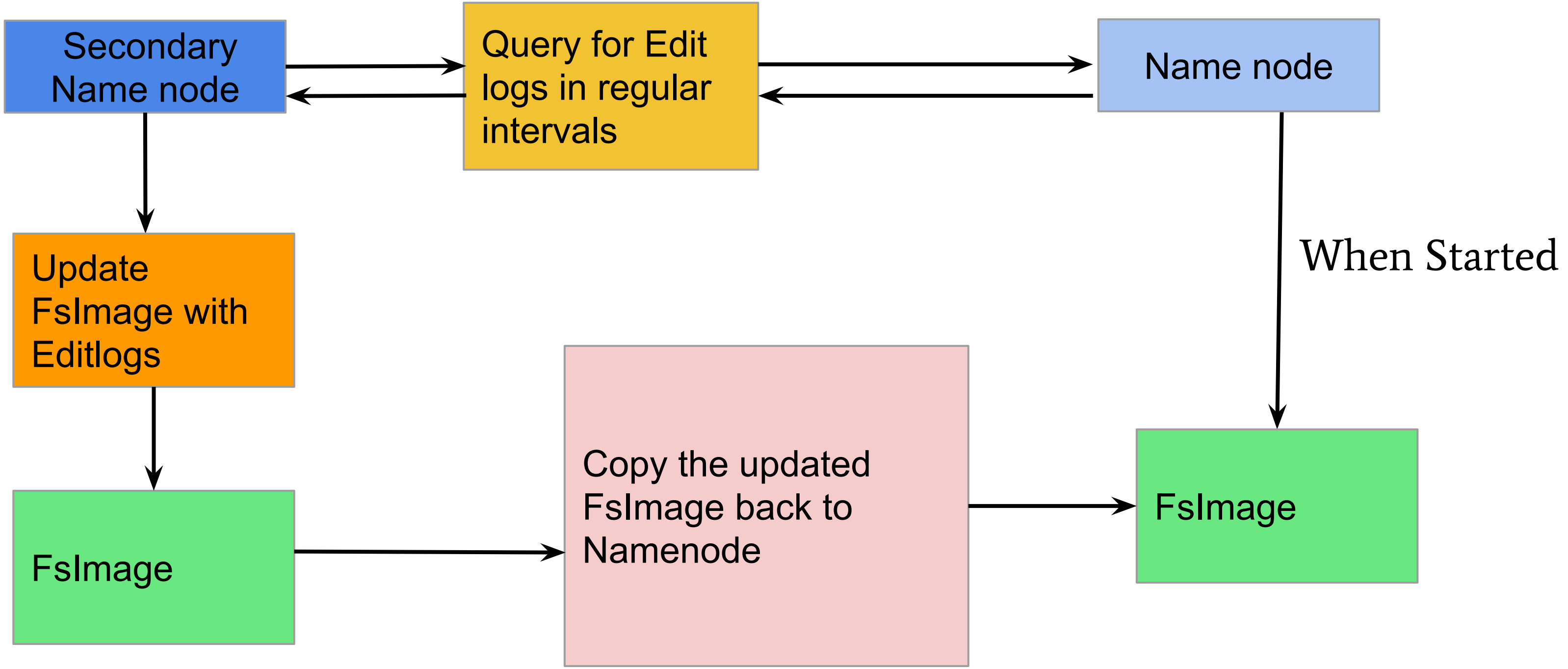
Datanodes is Block server stores data in local file.

Functions of Datanodes:

1. Each run on slave machine.
2. Actual Data is stored in datanodes
3. Performs low level read & write requests from file system's client
4. Send heartbeat to Namenode periodically.

HDFS Architecture

Secondary Namenode: [Not backup of Namenode]



HDFS Architecture

Secondary Namenode:

Works Currently with primary namenode

Functions of Secondary Namenode:

1. Constantly reads all files systems & metadata from RAM of Namenode and write to hard disk or file system.
2. Responsible for combining edit log with FsImage from Namenode.
3. Secondary namenode regularly checkpoint in HDFS, So it called as Checkpoint Node

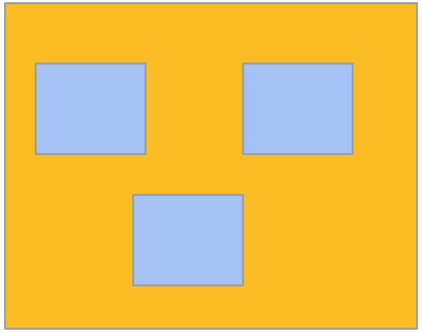
HDFS Architecture

Datanode, Blocks

DataNode

HDFS Architecture

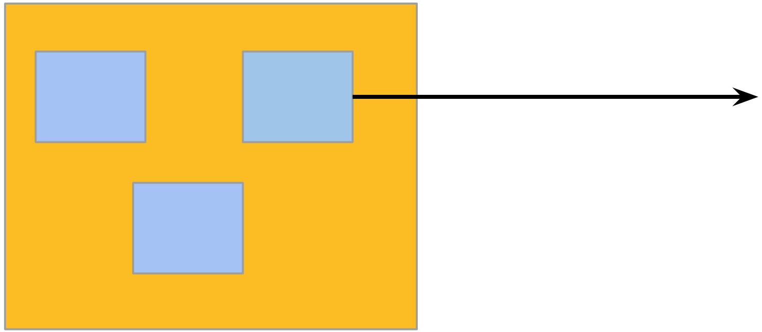
Block:



DataNode

HDFS Architecture

Block:



DataNode

Blocks:
Smallest continuous location on
your hard drive where data stored

Default Size of Block -128MB
HDFS Stores each file as Block

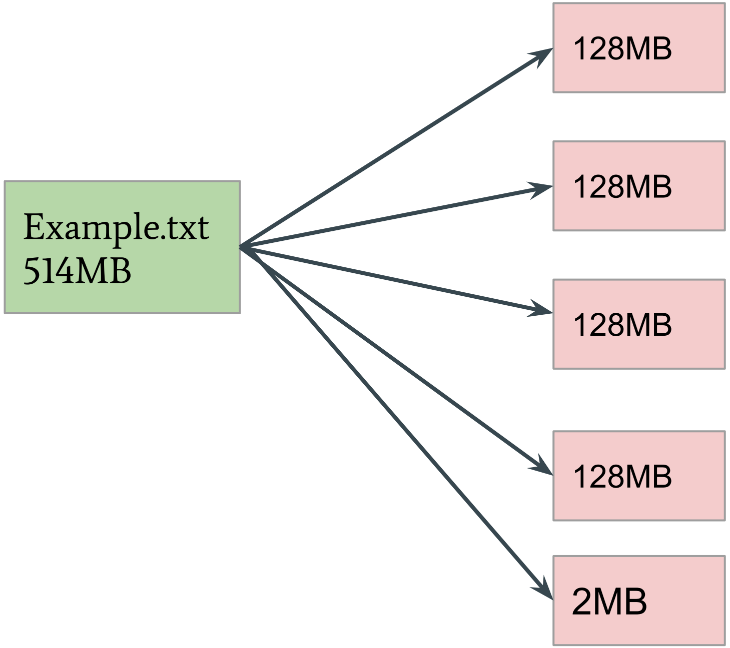
HDFS Architecture

Block:

Example.txt
514MB

HDFS Architecture

Block:



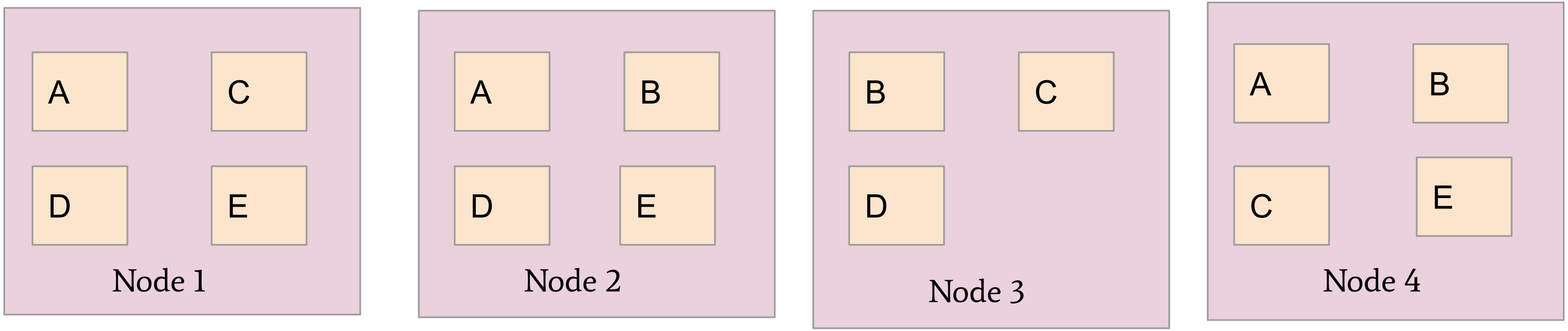
HDFS Architecture

Replication Management:

Data Blocks replicated to provide fault tolerance

Default Replication Factor 3 (We can change replication factor)

Every node having 3 times



Name node collects block report from datanode periodically to maintain the replication factor.

If block is over replicated or under replicated the name node deletes or add replicas as needed.

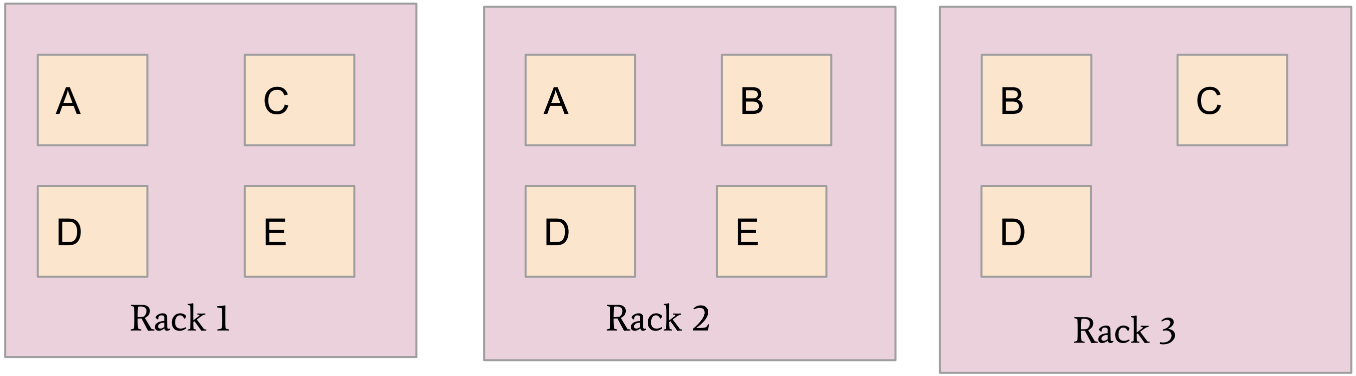
HDFS Architecture

Rack Awareness:

Name node takes place as all replicas not stored in same rack or single rack

It follows in-built rack awareness to reduce latency to provide fault tolerance.

First replica stored in one block and next to replicas stored in another block



HDFS Architecture

Hadoop Production Cluster

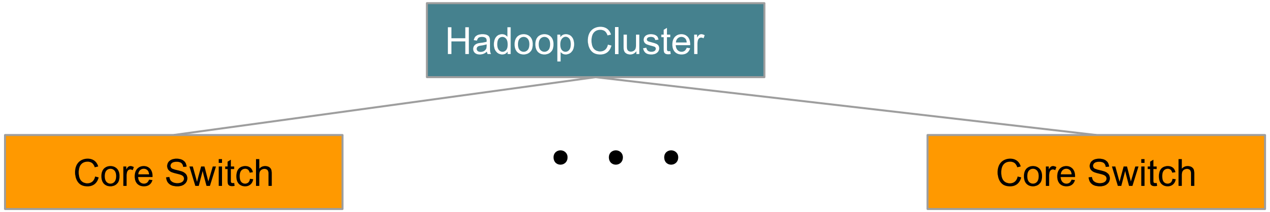
Multiple Racks Populated with data Node

Hadoop Cluster

HDFS Architecture

Hadoop Production Cluster

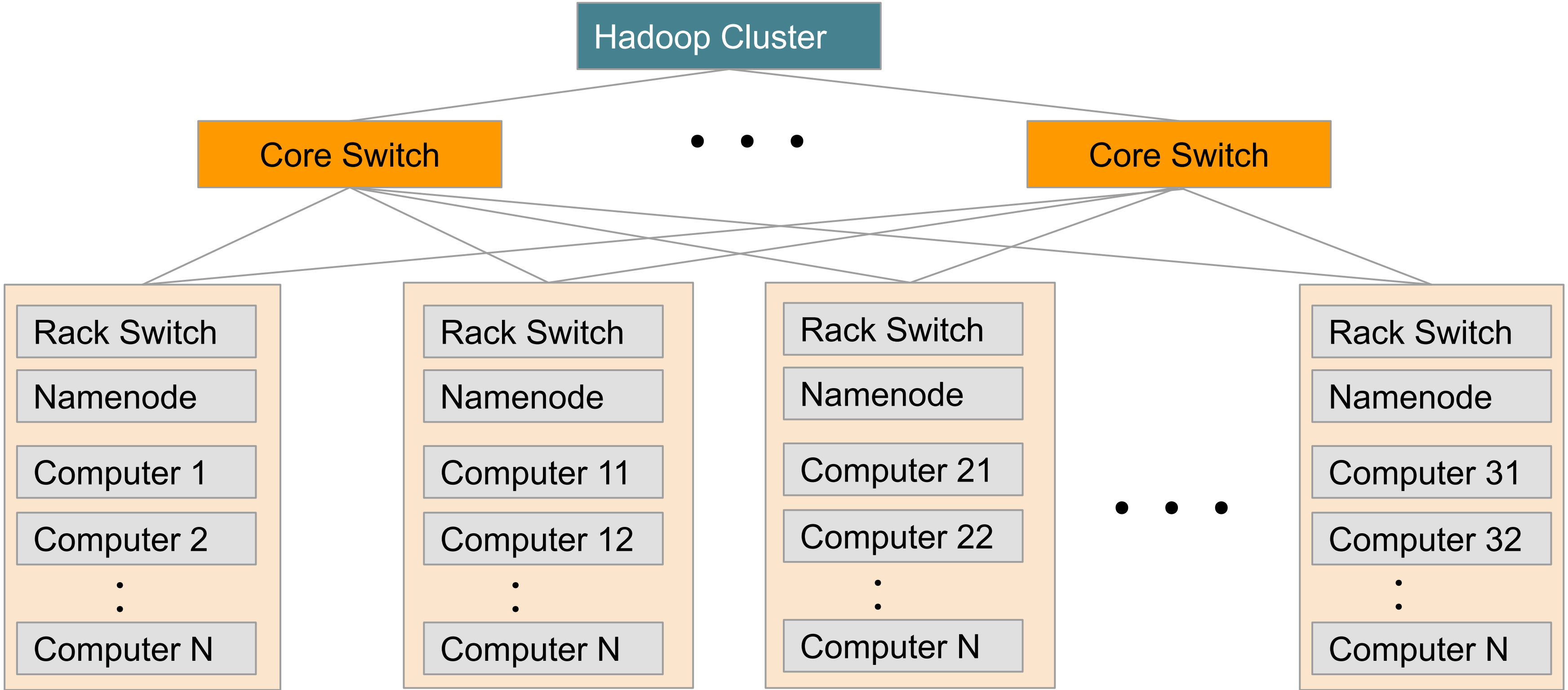
Multiple Racks Populated with data Node



HDFS Architecture

Hadoop Production Cluster

Multiple Racks Populated with data Node



HDFS Architecture

Advantages of Rack Awareness:

1. To improve Network Performance:
   All racks connected with core switch, Same rack having bandwidth comparing to different rack.

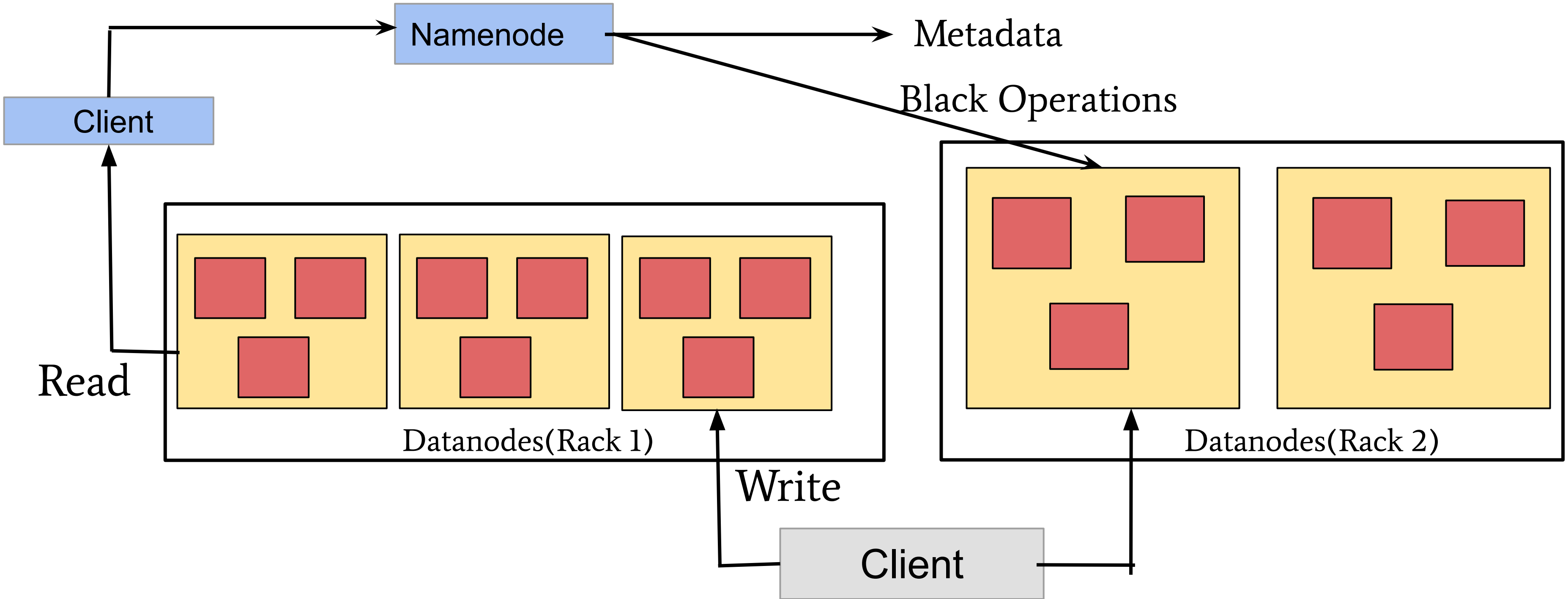
2. To prevent loss of data:
   If one rack fails but all data stored in different rack so data may not loss.

HDFS Architecture

HDFS follow write once read many philosophy

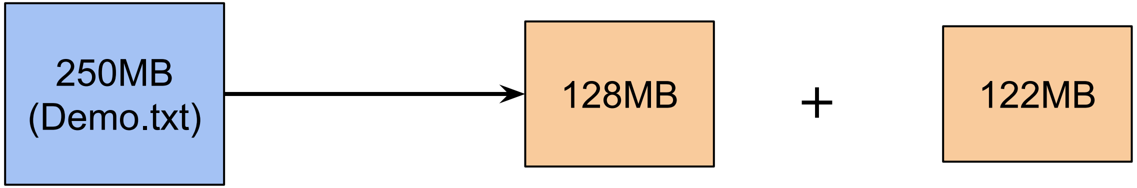
We can't edit files stored in HDFS.

We can append by reopening file.



HDFS Write Architecture

Ex: Client wants to write a file “Demo.txt” with size 250MB

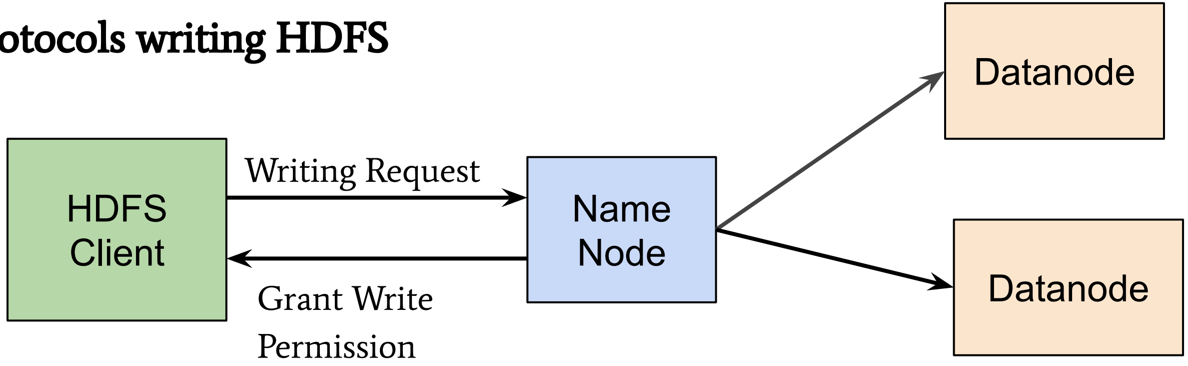


Default Size 128MB

A–128MB
Remaining Size 122MB

HDFS Write Architecture

Protocols writing HDFS



And shares IP Address of Data nodes

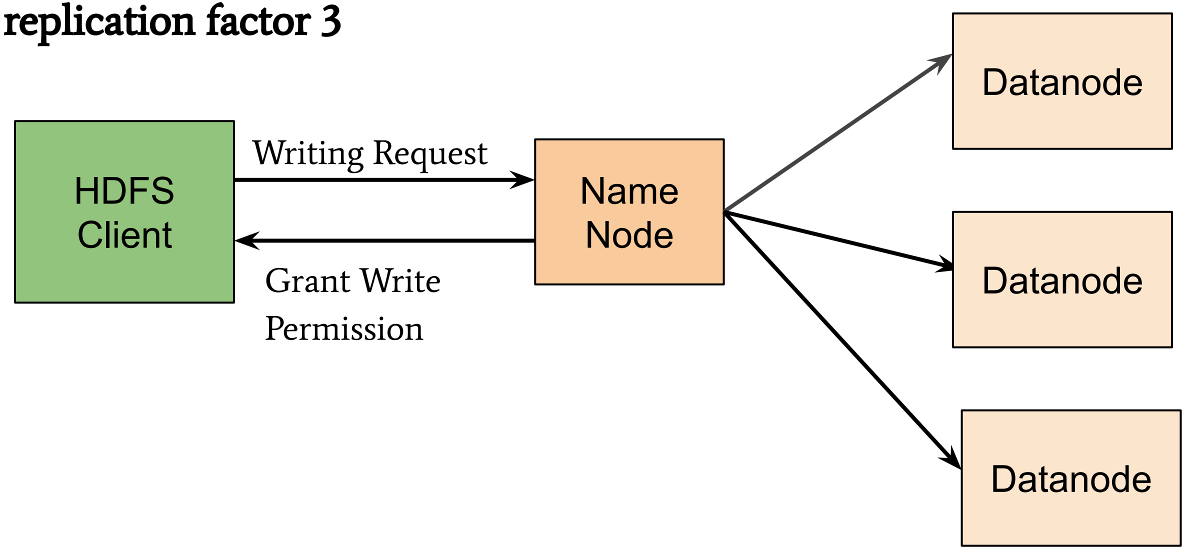
Now Selection of ip address purely randomized based on availability, replication factor , rack awareness

Ex: If replication factor 3

Only replicate 3 —-> Provide 3 Ip addresses of Data nodes

HDFS Write Architecture

If replication factor 3



Namenode provide list to client of 3 Ip address Datanodes

List will be unique for each block

Ex:

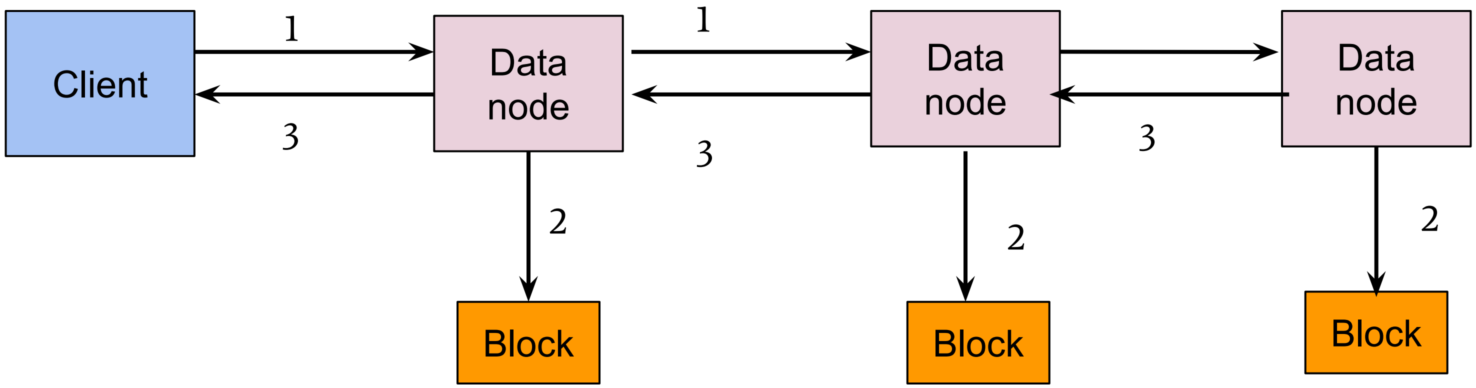
For Block A List A= {IP Address of Datanodes 1, 4, 6}

Block B List B= {IP Address of Datanodes 3, 7, 9}

HDFS Write Architecture

Each block copied in 3 different datanodes, to maintain replication factor.

1. Set up of pipeline
2. Data Streaming & replication
3. Shutdown of pipeline (Acknowledgment)

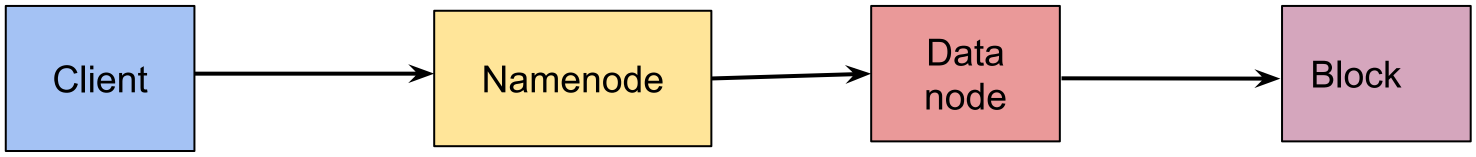


HDFS Write Architecture

Each block copied in 3 different datanodes, to maintain replication factor.

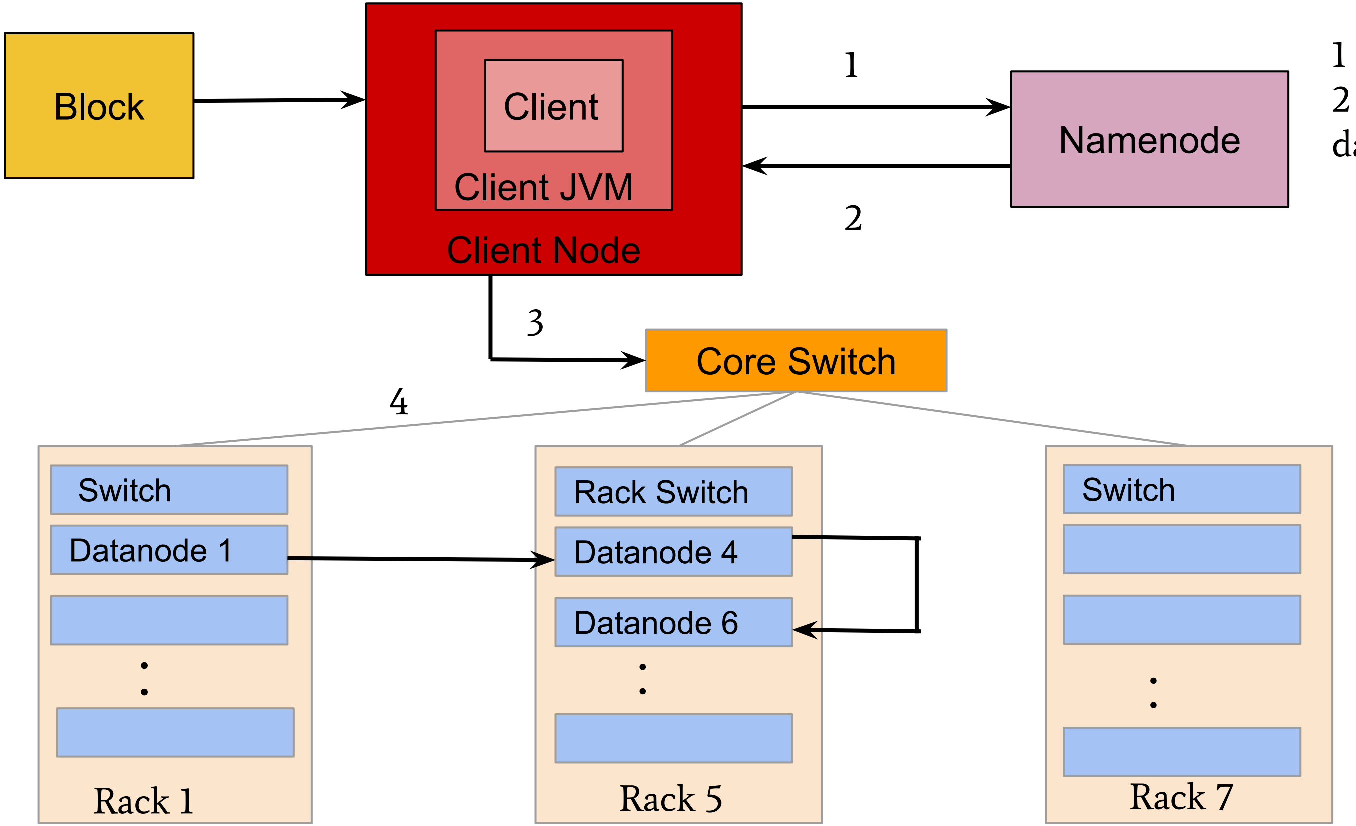
1. **Set up of pipeline:**
   Before writing client confirm whether datanode ready to receive data or not
   client creates pipeline for each block by connecting data nodes
   Whatever client contain that list will connect

Block A List A= {IP Address of Datanodes 1, 4, 6}
Block B List B= {IP Address of Datanodes 3, 7, 9}



HDFS Write Architecture

Setting up HDFS Write pipeline:



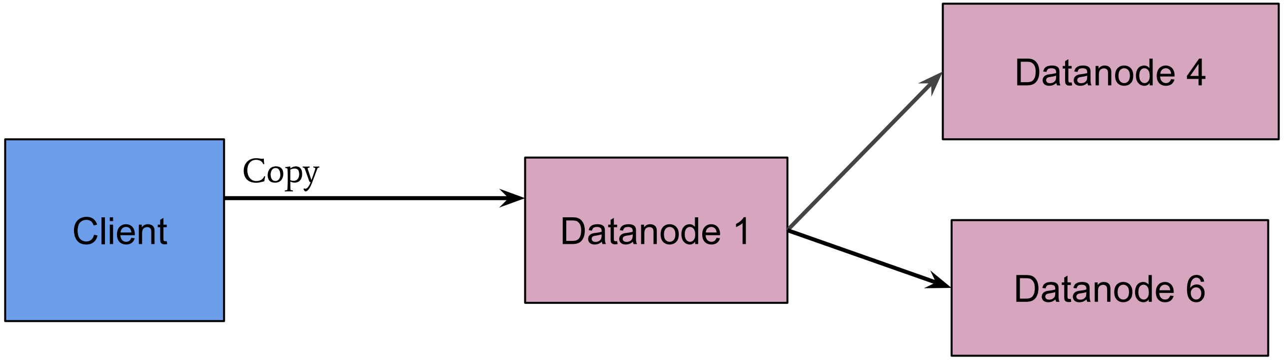
1 Write request for Block A
2 Send Ip Address of datanodes to client

HDFS Write Architecture

2. **Data Streaming:**

After creating pipeline client push data into pipeline, HDFS data is replicated based on replication factor

Block A stores 3 Datanodes as replication factor is 3

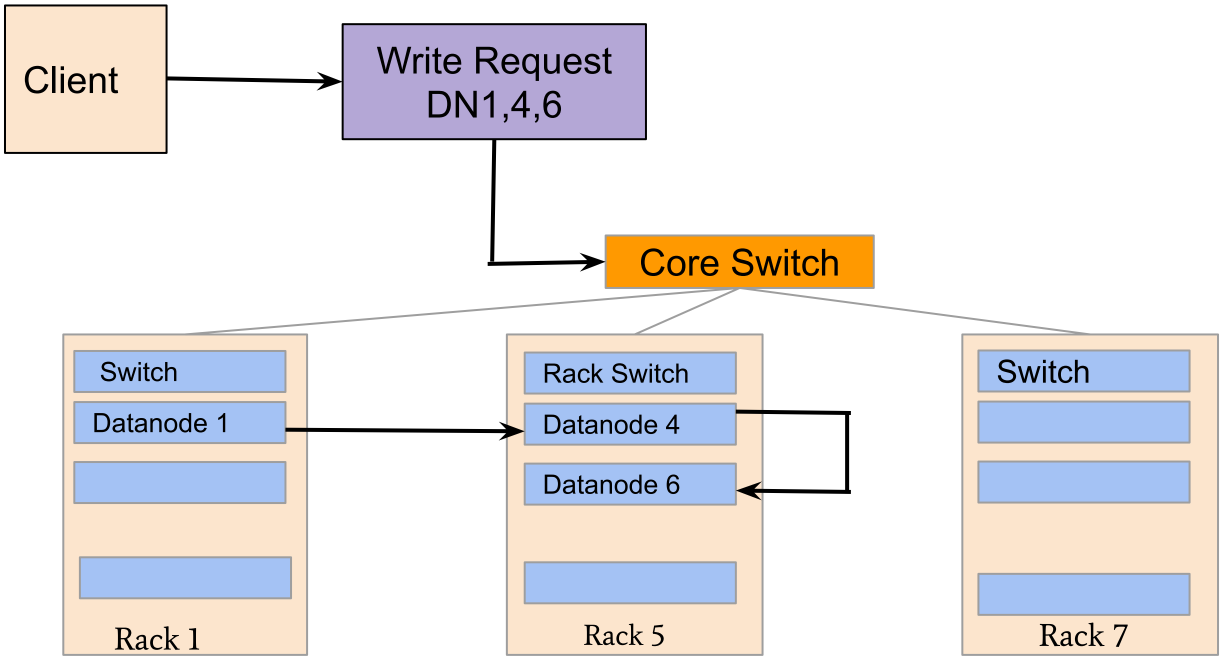


HDFS Write Architecture

2. **Data Streaming:**

Datanode 1 push block in pipeline data copied DN4

DN4 connect to DN6 last replica block

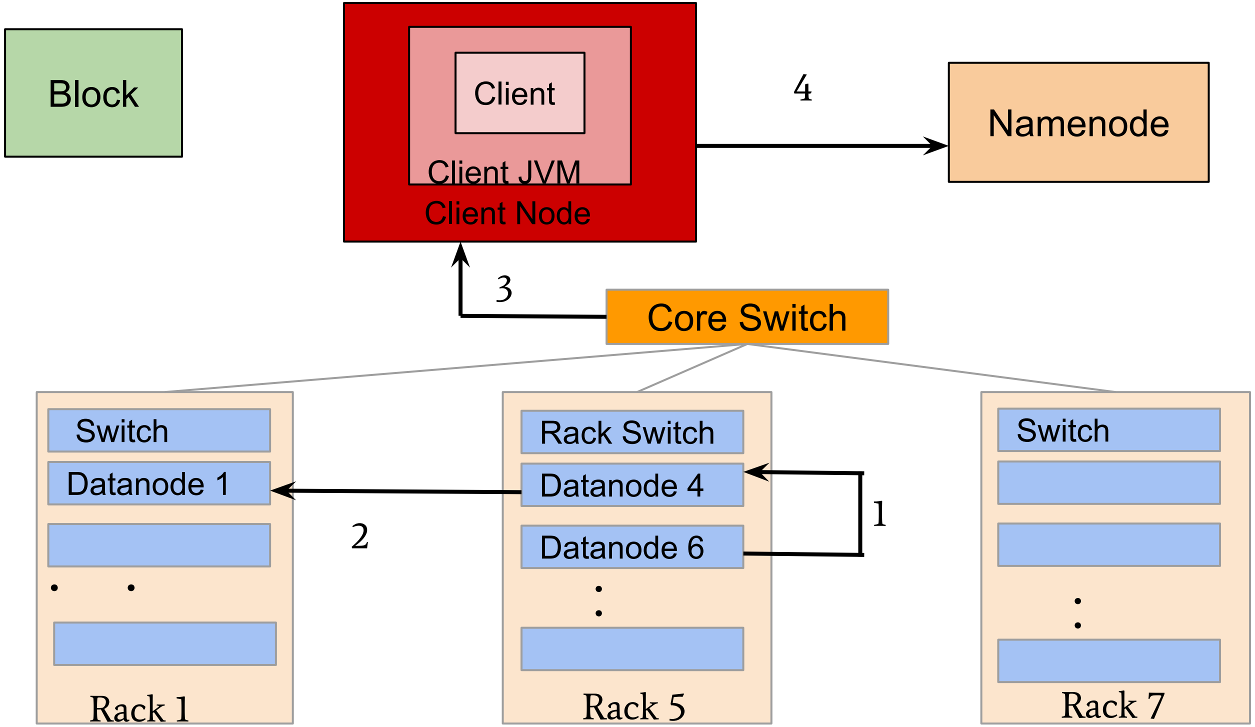


HDFS Write Architecture

3. **Shutdown of pipeline or Acknowledge Stage:**

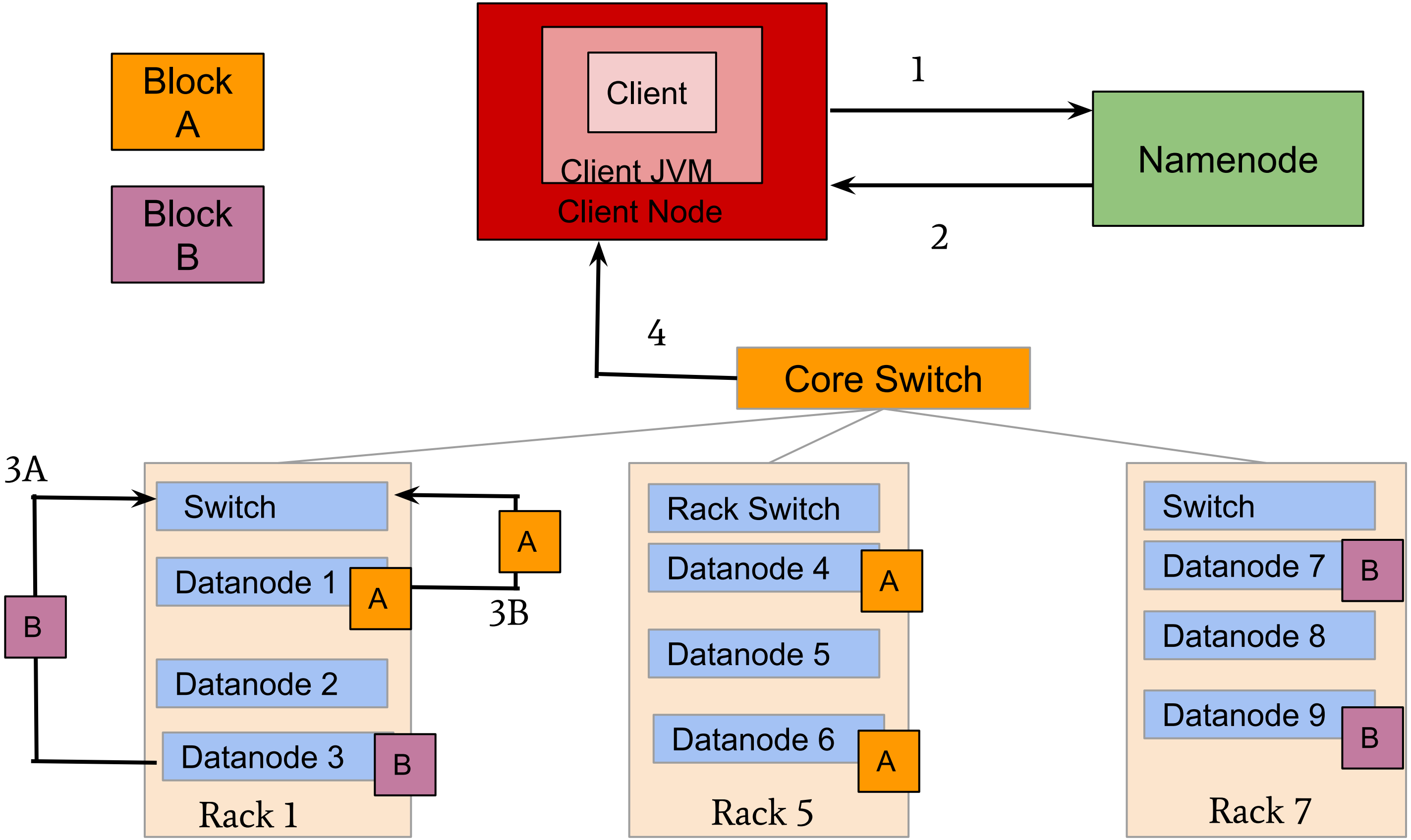
Once block has been copied 3 Datanodes series of Acknowledge takes place.

Acknowledgement—> Data Written Successful



1. Acknowledgment Datanode 6 to Datanode 4
2. Acknowledgment Datanode 4 to Datanode 1
3. Send Ack to Client
4. Write Successful

HDFS Read Architecture



1. Read Request for Bock A,B
2. Send Ip Address of DN1, DN3
3. Reading Data from block A, B
4. Read from DN1, DN3

hadoop

YARN

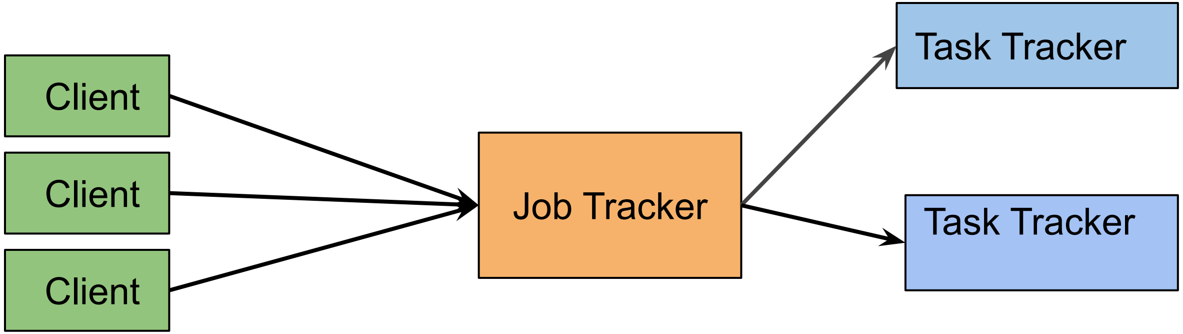
YARN

Yarn- Yet Another Resource Negotiator
Map reduce performed both processing & resource management functions

Job Tracker [Single Master]

1. Allocated the resources
2. Performed Scheduling
3. Monitor the processing jobs

If assigned Map & Rescue task on subordinate process called Task Trackers
Task Trackers: Report their programs to job tracker



YARN

Yarn allows different data processing methods

1. Graph processing
2. Interactive processing
3. Stream processing
4. Batch processing

To run and process Data stored in HDFS

YARN

Hadoop 1.0

Map Reduce Data Processing & Resource Management

HDFS

Hadoop 2.0

Map Reduce

Other Data Processing frameworks

YARN
Resource Manager

HDFS

YARN

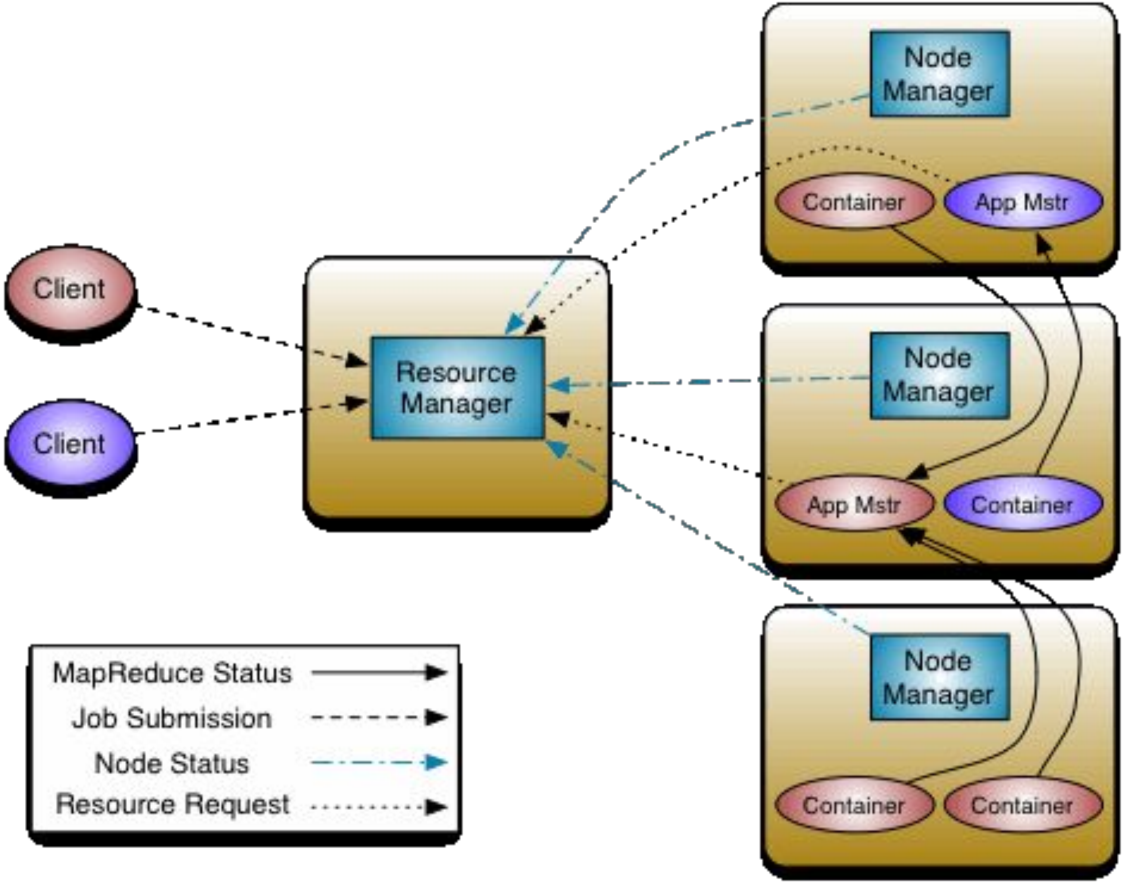
Yarn enable users to perform operations as per requirement

Spark → Real Time Processing
Hive → SQL
HBase → NoSQL

We can use all these in Same cluster

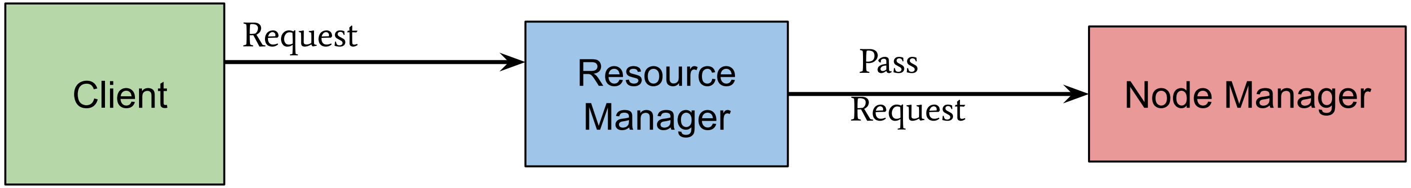
Yarn Performs resource management , Job Scheduling

Main Components of YARN



Main Components of YARN

1. **Resource Manager:**
   Allocate cluster resources using scheduler & Application Manager



Check Available resources to complete applications

Main Components of YARN

1. **Resource Manager:**

   Scheduler :

1. Allocate resources to various running applications
2. Only schedule applications so it called as pure scheduler
3. If any application failure scheduler not guarantee because it doesn’t track applications
4. This scheduling performed based on resource requirement
5. **Pluggable Policy:** Responsible for partitioning the cluster resources on various applications.

Main Components of YARN

1. **Resource Manager:**

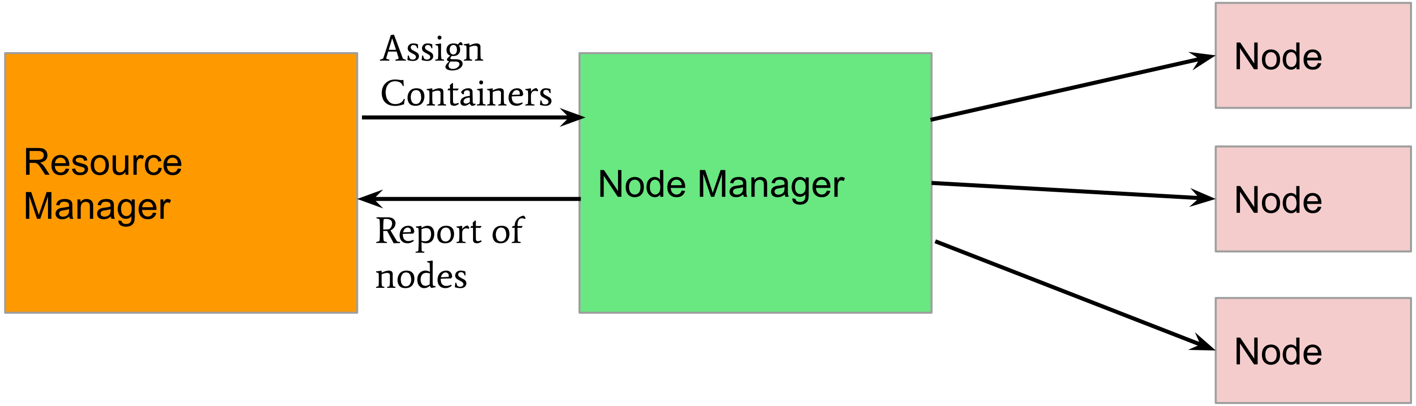
   Application Manager :

1. Responsible for accepting job submission.
2. Take first container from resource manager for executing applications.
3. Manage running application master in a cluster & provide service for restarting application master container on failure.
4. Resource manager optimize cluster utilization like keeping all resources in use.

Main Components of YARN

2. Node Manager:

1. Takes care of all individual nodes in a cluster and manages user jobs



Primary goal to manage application containers assigned by resource manager

Main Components of YARN

2. Node Manager:

Application Master requests the assigned container from the Node Manager by sending it a Container Launch Context(CLC)

Node manager creates requested container process and starts it

Perform log management

It kills container as directed by resource manager

Main Components of YARN

3. Application Master:

1. Application Master is a Single Job submitted to framework
2. Every Application have a unique application master associated with framework
3. Manage faults in applications execution in cluster

Main Tasks:

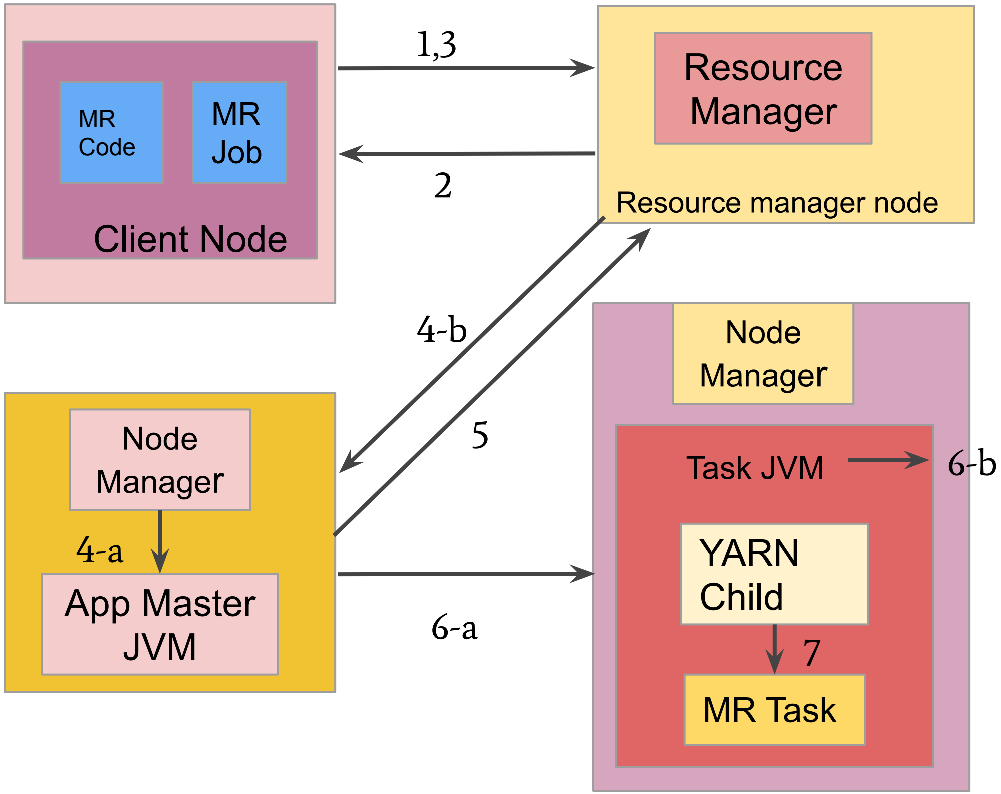
- Negotiate resources from Resource manager and works with node manager & monitor tasks
- Its should take appropriate containers from Resource Manager
- Track & monitor containers status
- Updates pass to resource manager

Main Components of YARN

4. Container:

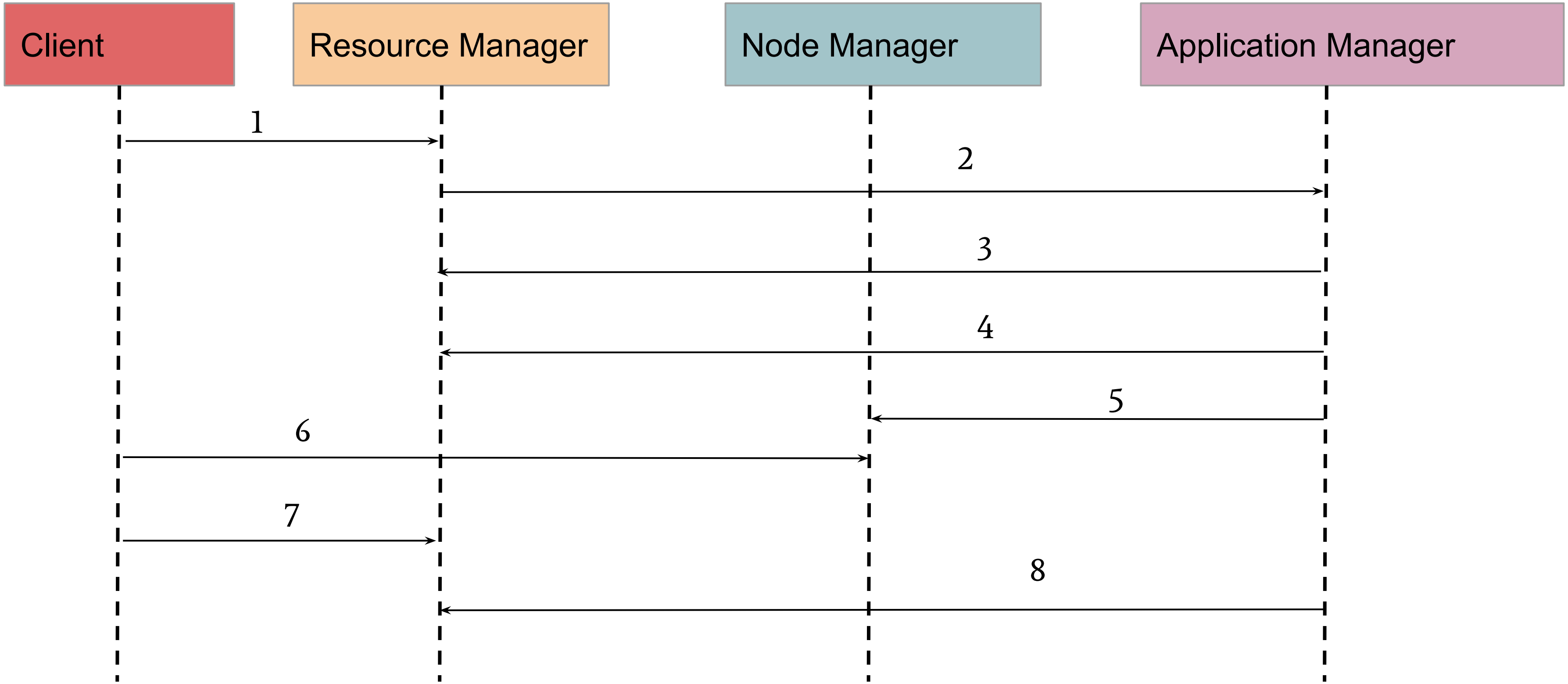
1. Collection of physical resources[RAM, CPU, Disks on Single node]
2. Yarn containers managed by CLC(Container Launch Context or Container Life Cycle)
3. CLC contains map of environment variable dependencies stored in remotely accessible storage, Security tokens, Payloads for Node manager.
4. Grant rights to application to use specific amount of resources on specific host

Application submission steps in YARN



1. Submit the Job
2. Get Application ID
3. Application submission context
4. (a) Start Container launch
   (b) Launch Application Master
5. Allocate Resources
6. (a) Container
   (b) Launch
7. Execute

Application Workflow of YARN



Application Workflow of YARN

1. Client submit an application
2. Resource Manager allocates a container to start application manager
3. Application manager register with resource manager
4. Application manager ask container from resource manager
5. Application manager notifies Node manager to launch containers
6. Application code executed in container
7. Client contact resource manager [Application manager to monitor application status]
8. Application manager unregister with resource manager

Hadoop Ecosystem

Big Data

01 HDFS

02 YARN

03 Map Reduce

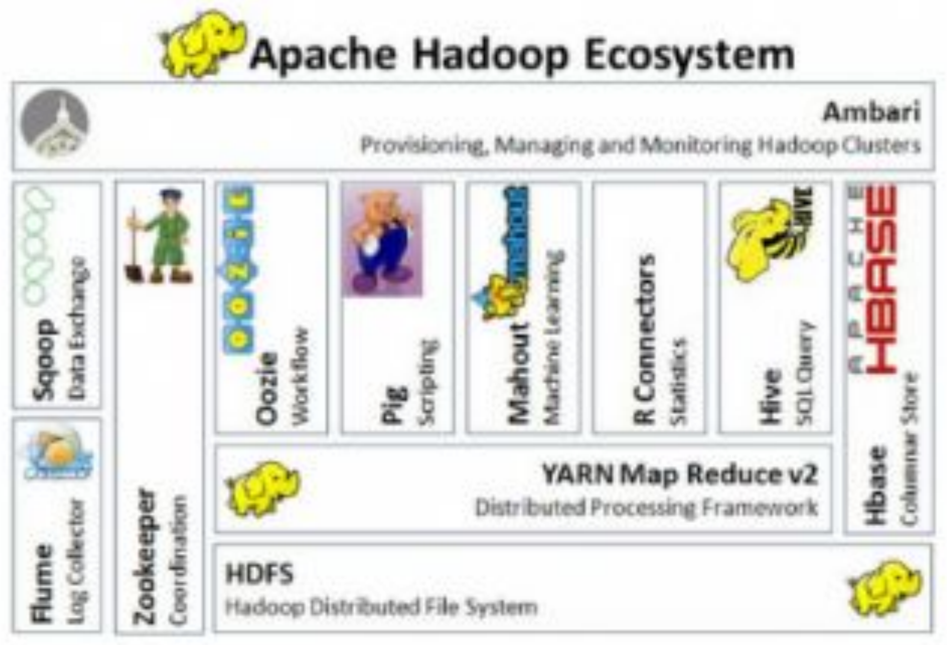
04 Apache Pig, Apache Spark, Hive

05 Other Tools of Hadoop

Hadoop Ecosystem

Hadoop is is a platform with various integral components that enable distributed data storage and processing.

These components together form the Hadoop ecosystem.



Hadoop Core Components

1. HDFS
2. MapReduce
3. YARN
4. Hive
5. Apache Pig
6. Apache HBase
7. Sqoop,
8. Apache Flume
9. Apache Drill
10. Apache Mahout
11. Zookeeper

hadoop

HDFS

HDFS

Distributed File System:
Managing Data [Files, Folders across multiple servers, Computers]

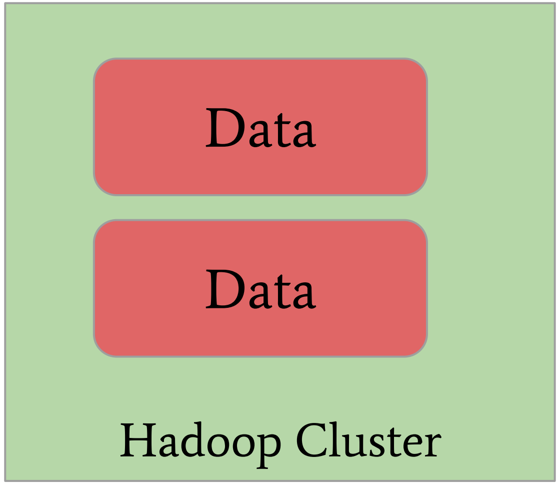
DFS allow us to store data in multiple nodes or multiple machines in a cluster, multiple users can access data.

Ex: Works like File System
Windows—> NTFS [New Technology File System]
Mac —>HFS [Hierarchical File System]
NTFS, HFS Stores data in single machine
DFS Stores data in multiple machines

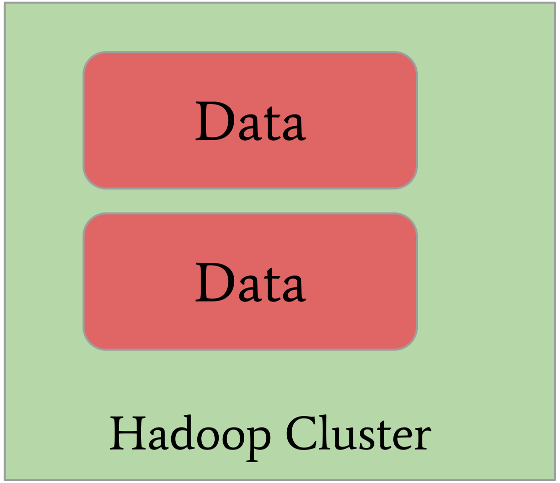
HDFS

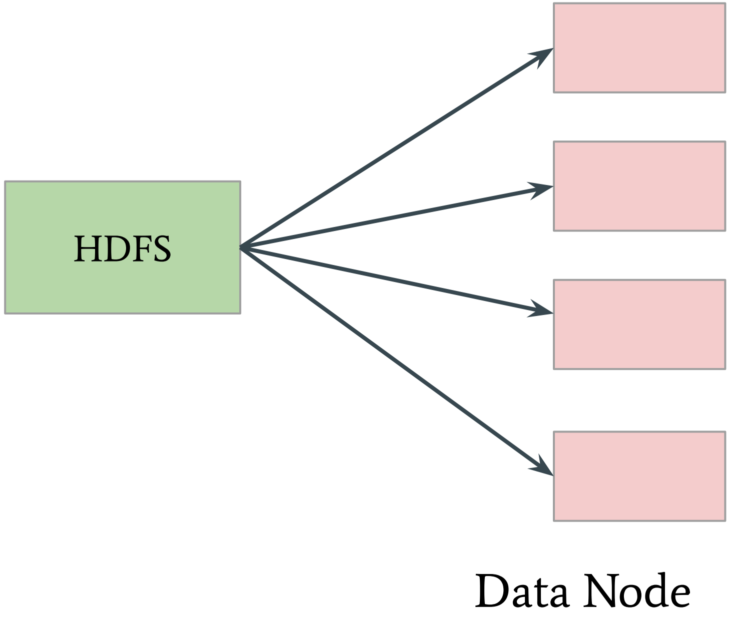
Hadoop Cluster

HDFS



HDFS



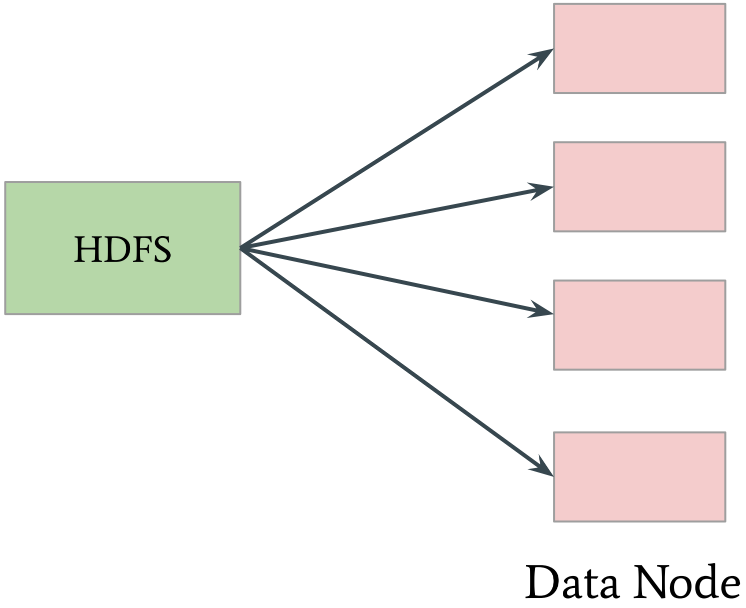


HDFS

Advantages of Hadoop:

1. **Distributed Storage:**

Example:
10TB data stored in 10 machines
But it looks like all 10 TB data in Single machine



HDFS

Advantages of Hadoop:

2. Distributed & Parallel Computation:

Example:

30 mins to process 1TB data in machine

Taking 1 TB Data and processing in 10 machines

1 Machine -30 mins

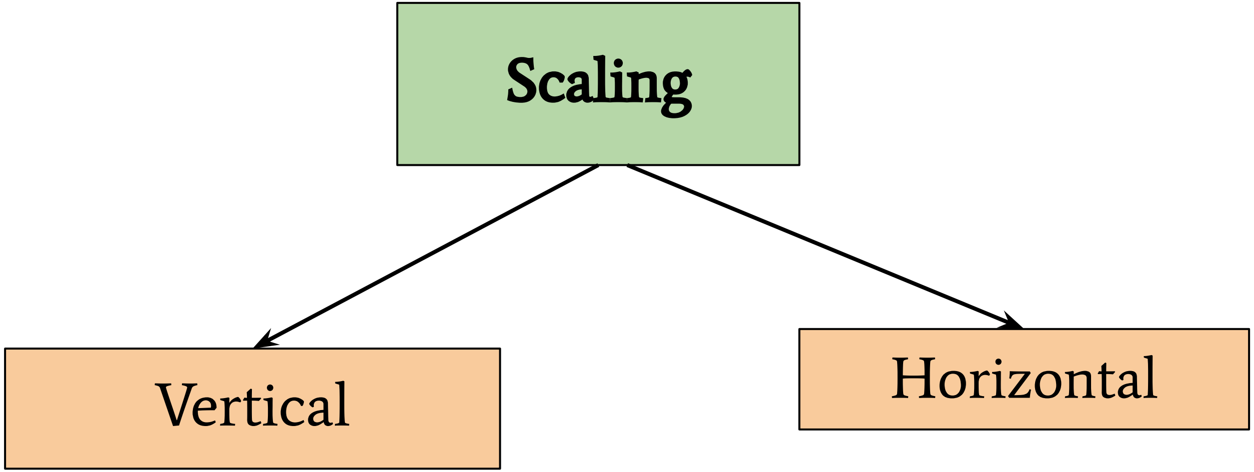
10 machines-? -----> 3 mins

Data Divided and processed over 10 machines parallely.

HDFS

Advantages of Hadoop:
3. Horizontal Scalability:

Vertical , Horizontal



HDFS

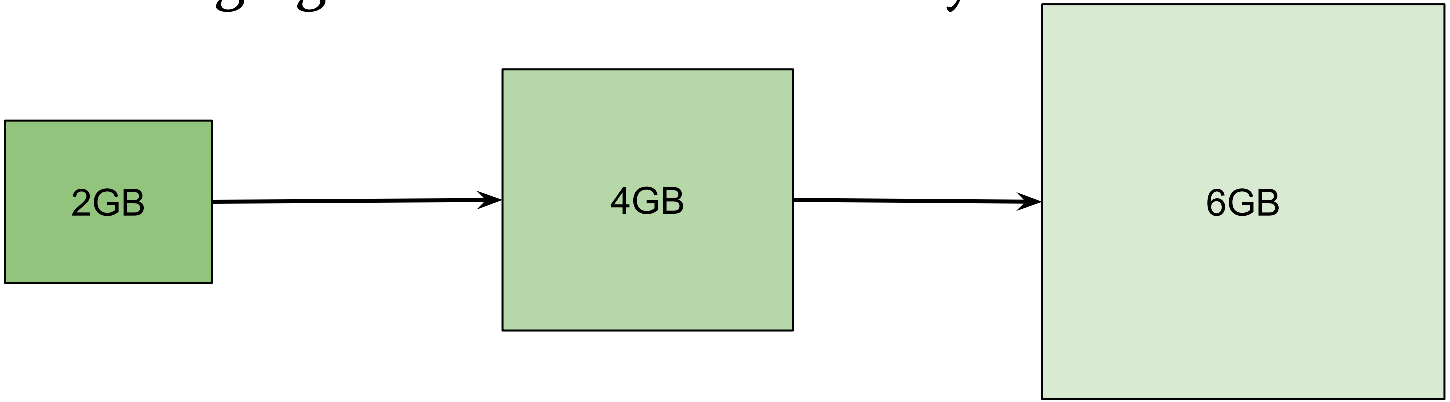
Advantages of Hadoop:

Vertical Scalability[Scale up]:

Increase hardware capacity of system

There is a limit to increase hardware capacity

After changing size we have to restart system



HDFS

Advantages of Hadoop:

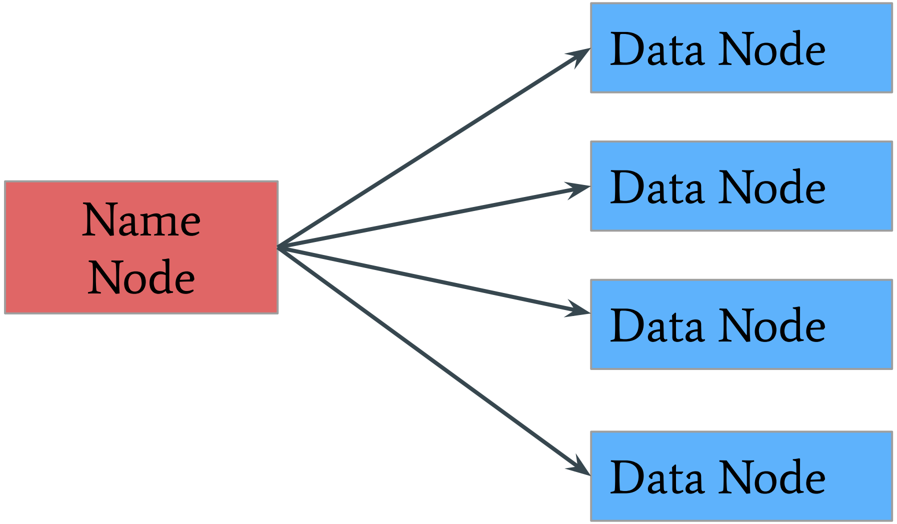
Horizontal Scalability:

Add more nodes to existing cluster instead of increasing hardware capacity

We can add more machines without stopping them

At last all machines run parallel as per requirement

HDFS Architecture



HDFS is Block Structured File System
Each file divided into particular size

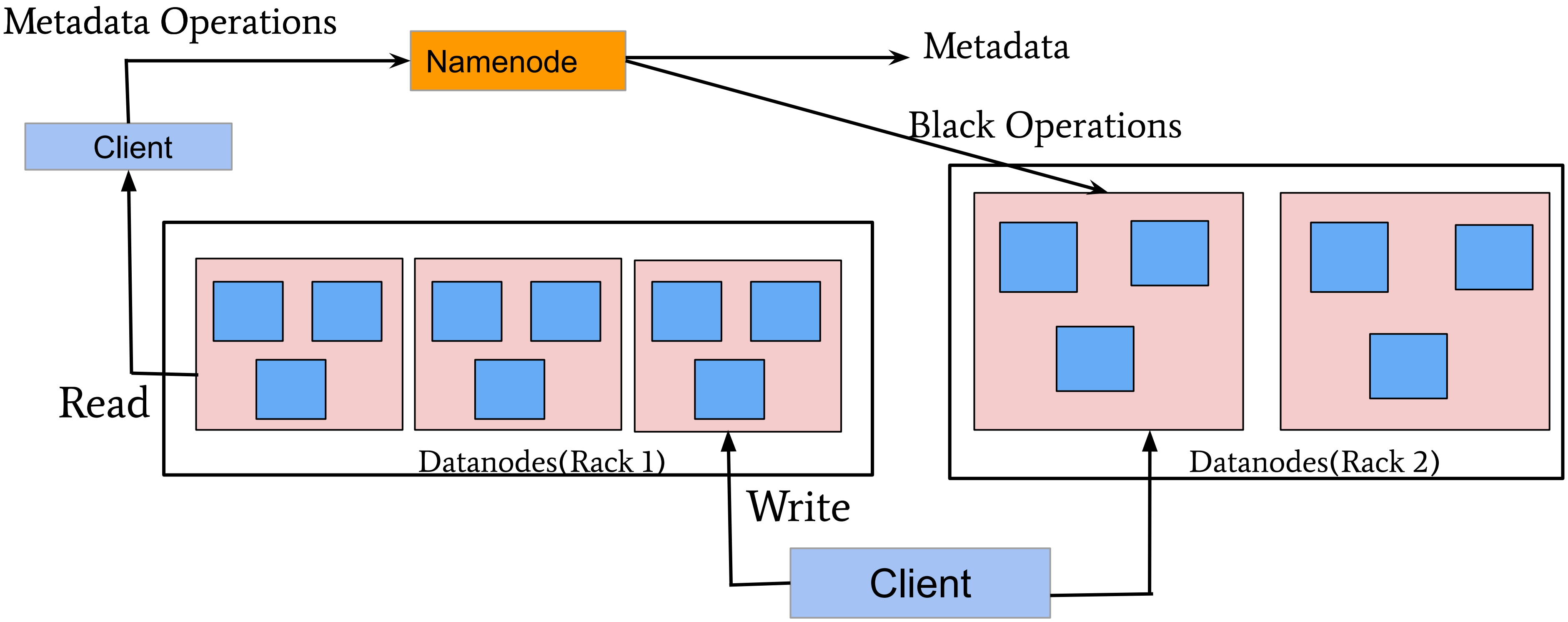
These blocks stored across cluster one or more machines

Follow Master/ Slave Architecture
Master- Namenode
Slave - Multiple Datanodes

Datanodes can run on single machine or several machines

HDFS Architecture

Name Node



HDFS Architecture

Functions of Namenode:

1. Maintain & manages Data Nodes
2. Records Metadata

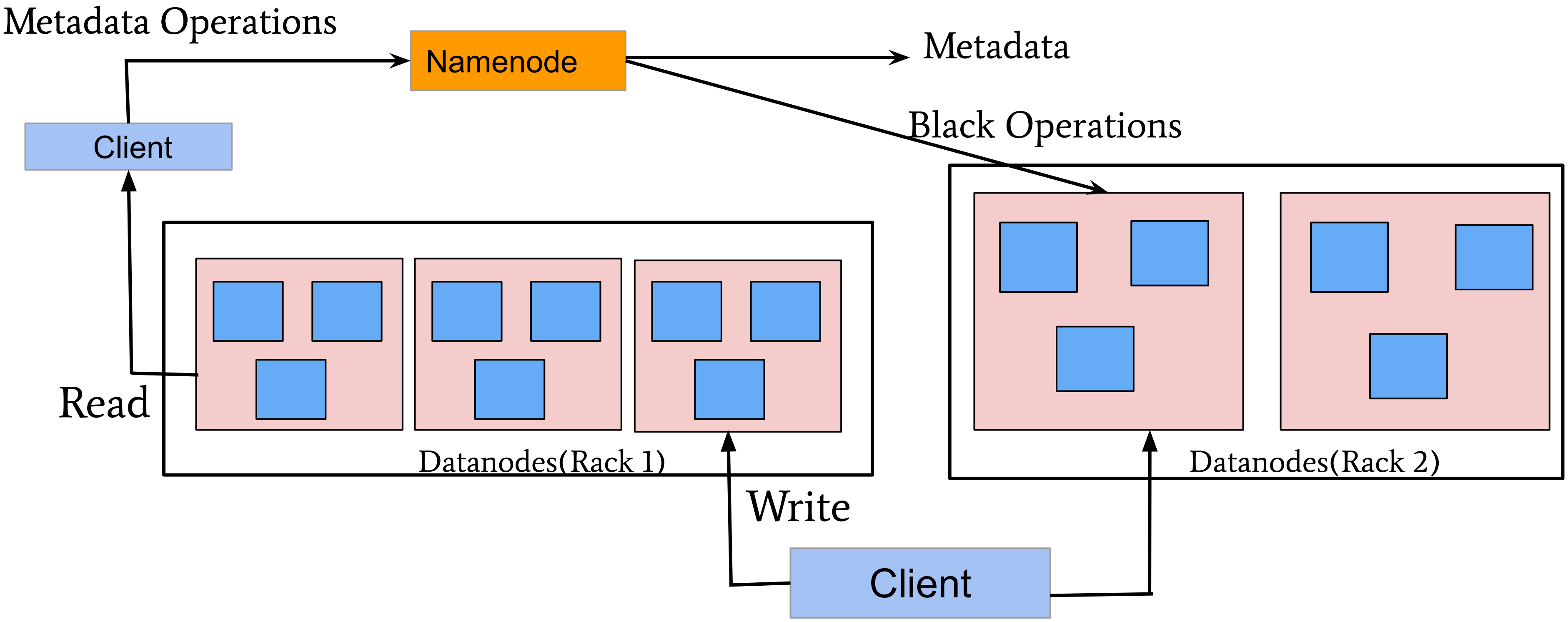
   Metadata contains 2 components

   FsImage: Complete state of file system stores from start

   Edit logs: All recent modifications with respect to FsImage
3. Records all Changes
4. Receives regular report of all datanodes, blocks
5. Also responsible for replication factor.
6. If Datanode fails Namenode Choose new datanode for new replicas

HDFS Architecture

Data Node



HDFS Architecture

Datanode:

Slave nodes in HDFS

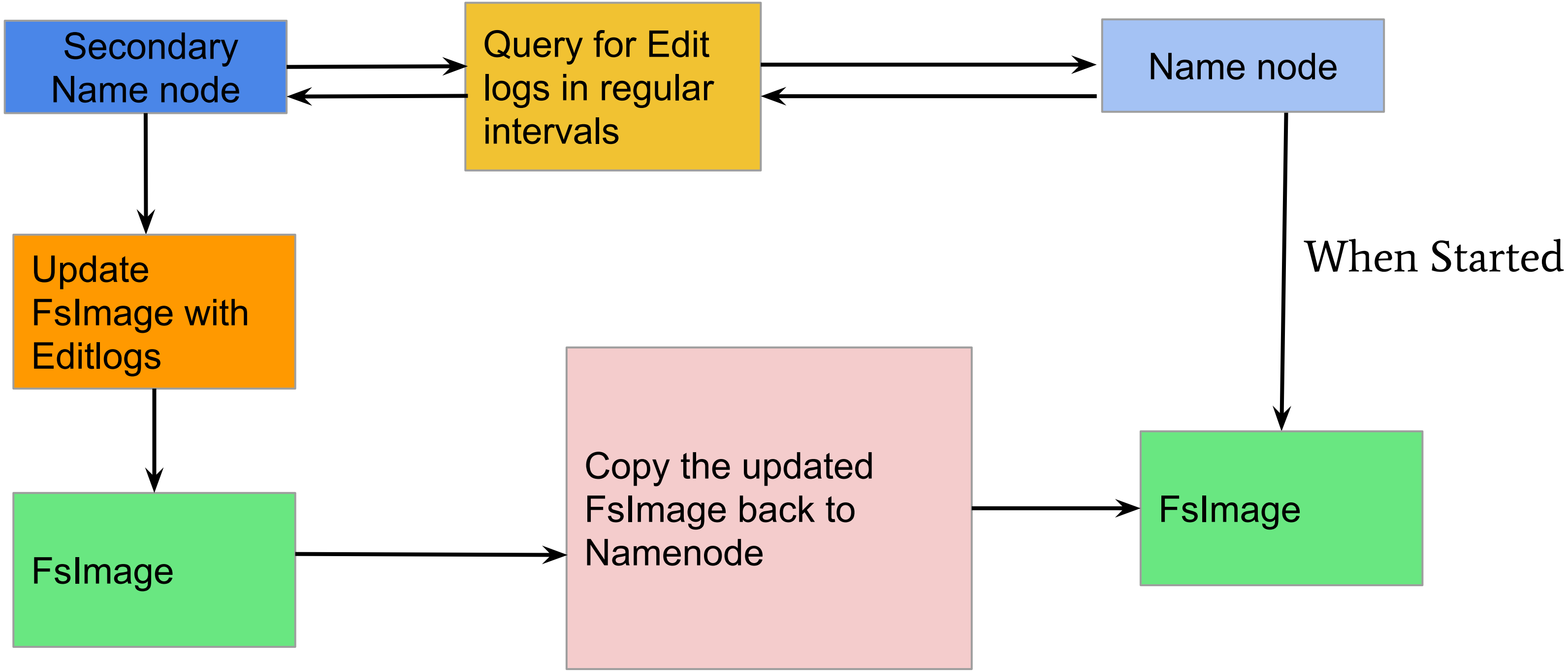
Datanodes is Block server stores data in local file.

Functions of Datanodes:

1. Each run on slave machine.
2. Actual Data is stored in datanodes
3. Performs low level read & write requests from file system’s client
4. Send heartbeat to Namenode periodically.

HDFS Architecture

Secondary Namenode: [Not backup of Namenode]



HDFS Architecture

Secondary Namenode:

Works Currently with primary namenode

Functions of Secondary Namenode:

1. Constantly reads all files systems & metadata from RAM of Namenode and write to hard disk or file system.
2. Responsible for combining edit log with FsImage from Namenode.
3. Secondary namenode regularly checkpoint in HDFS, So it called as Checkpoint Node

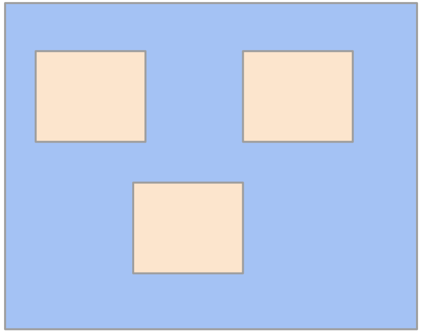
HDFS Architecture

Datanode, Blocks

DataNode

HDFS Architecture

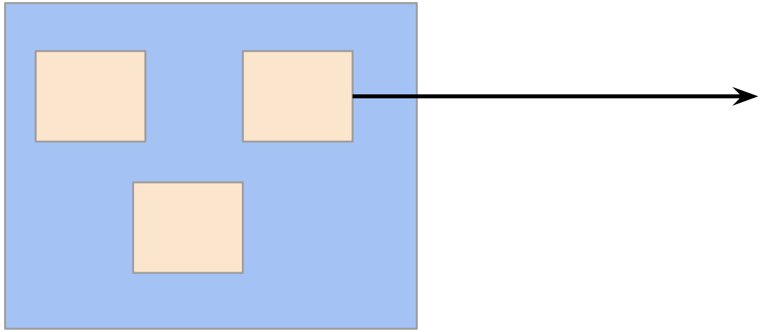
Block:



DataNode

HDFS Architecture

Block:



DataNode

Blocks:
Smallest continuous location on your hard drive where data stored

Default Size of Block -128MB
HDFS Stores each file as Block

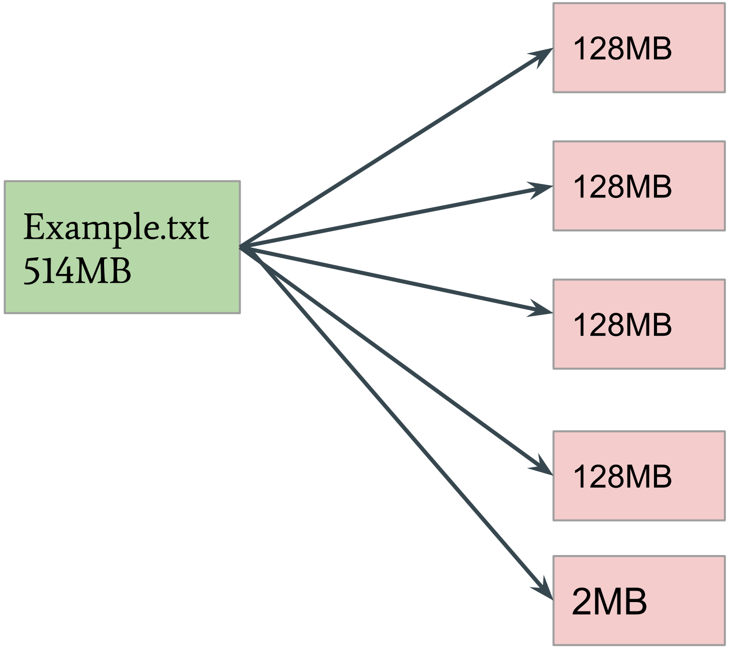
HDFS Architecture

Block:

Example.txt
514MB

HDFS Architecture

Block:



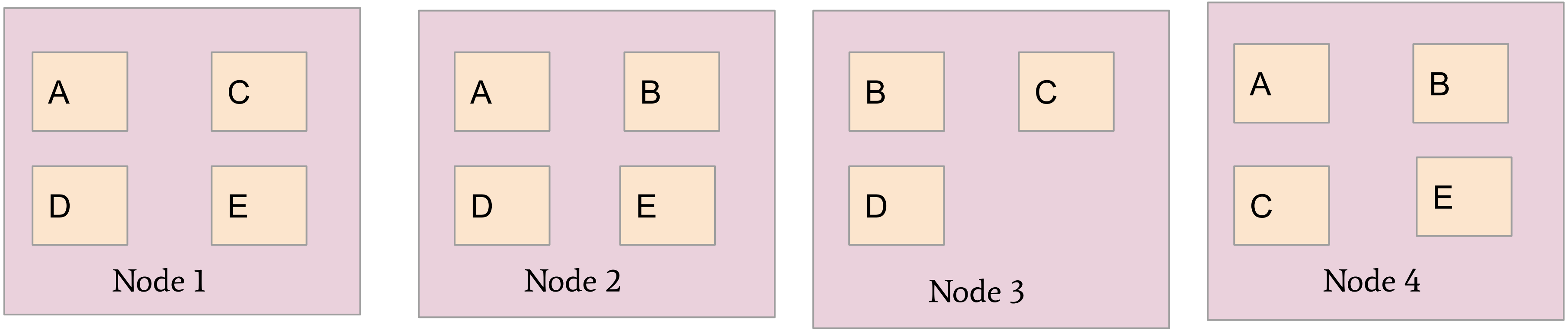
HDFS Architecture

Replication Management:

Data Blocks replicated to provide fault tolerance

Default Replication Factor 3 (We can change replication factor)

Every node having 3 times



Name node collects block report from datanode periodically to maintain the replication factor.

If block is over replicated or under replicated the name node deletes or add replicas as needed.

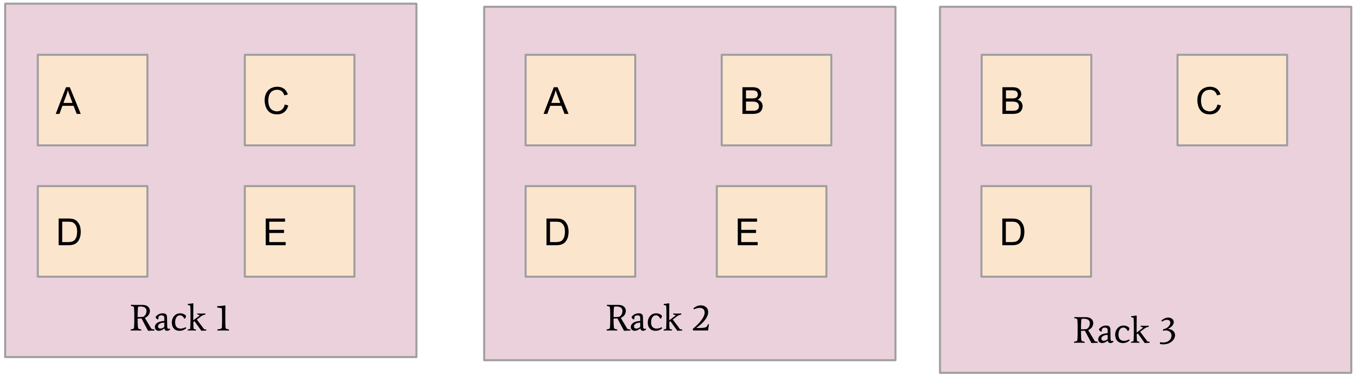
HDFS Architecture

Rack Awareness:

Name node takes place as all replicas not stored in same rack or single rack

It follows in-built rack awareness to reduce latency to provide fault tolerance.

First replica stored in one block and next to replicas stored in another block



HDFS Architecture

Hadoop Production Cluster

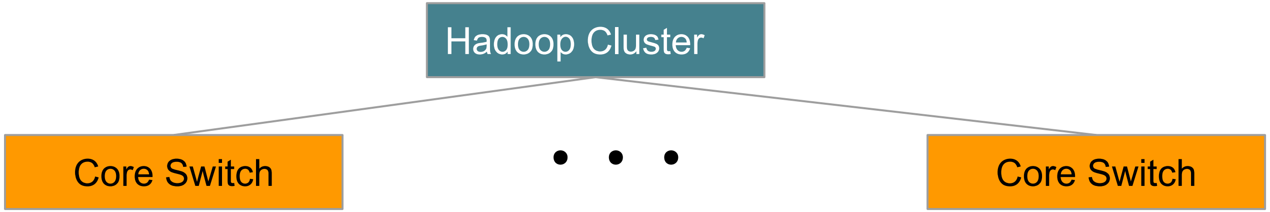
Multiple Racks Populated with data Node

Hadoop Cluster

HDFS Architecture

Hadoop Production Cluster

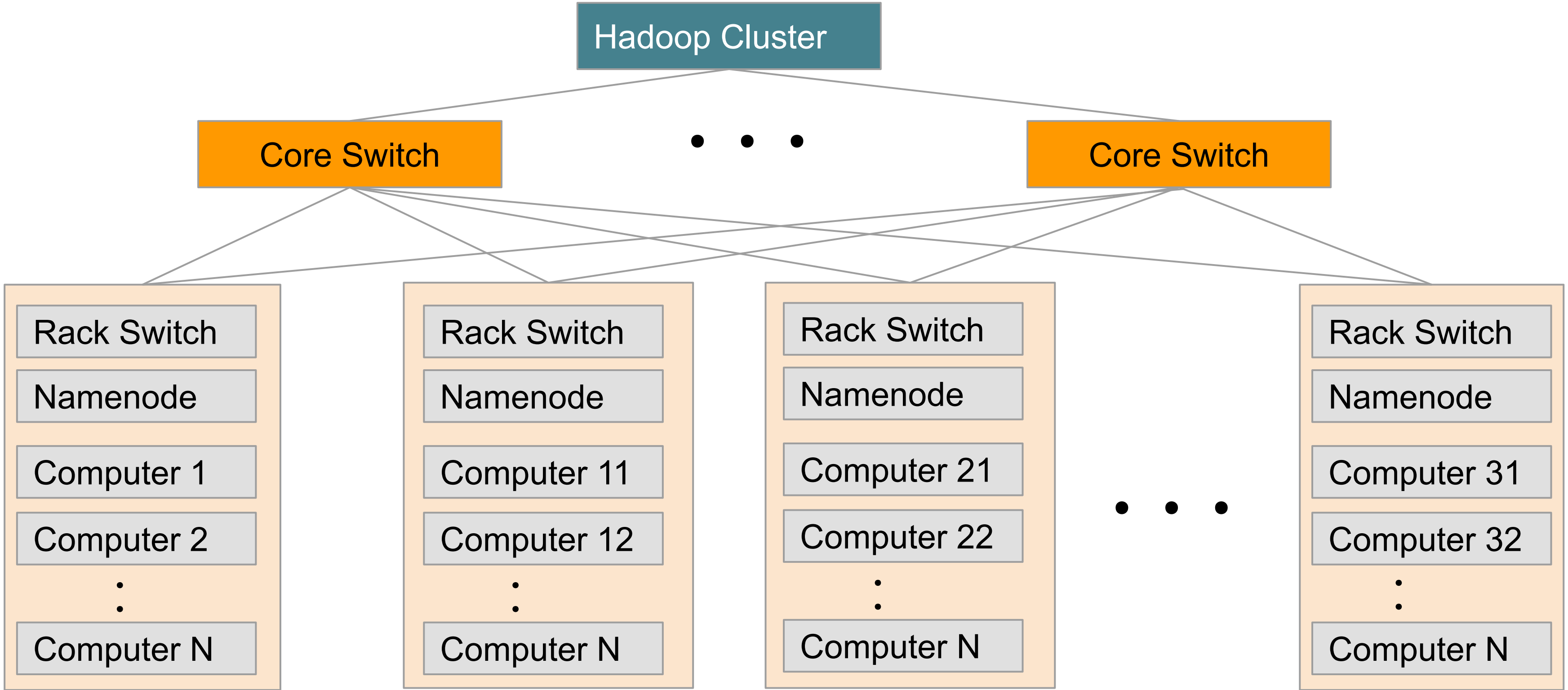
Multiple Racks Populated with data Node



HDFS Architecture

Hadoop Production Cluster

Multiple Racks Populated with data Node



HDFS Architecture

Advantages of Rack Awareness:

1. To improve Network Performance:

   All racks connected with core switch, Same rack having bandwidth comparing to different rack.

2. To prevent loss of data:

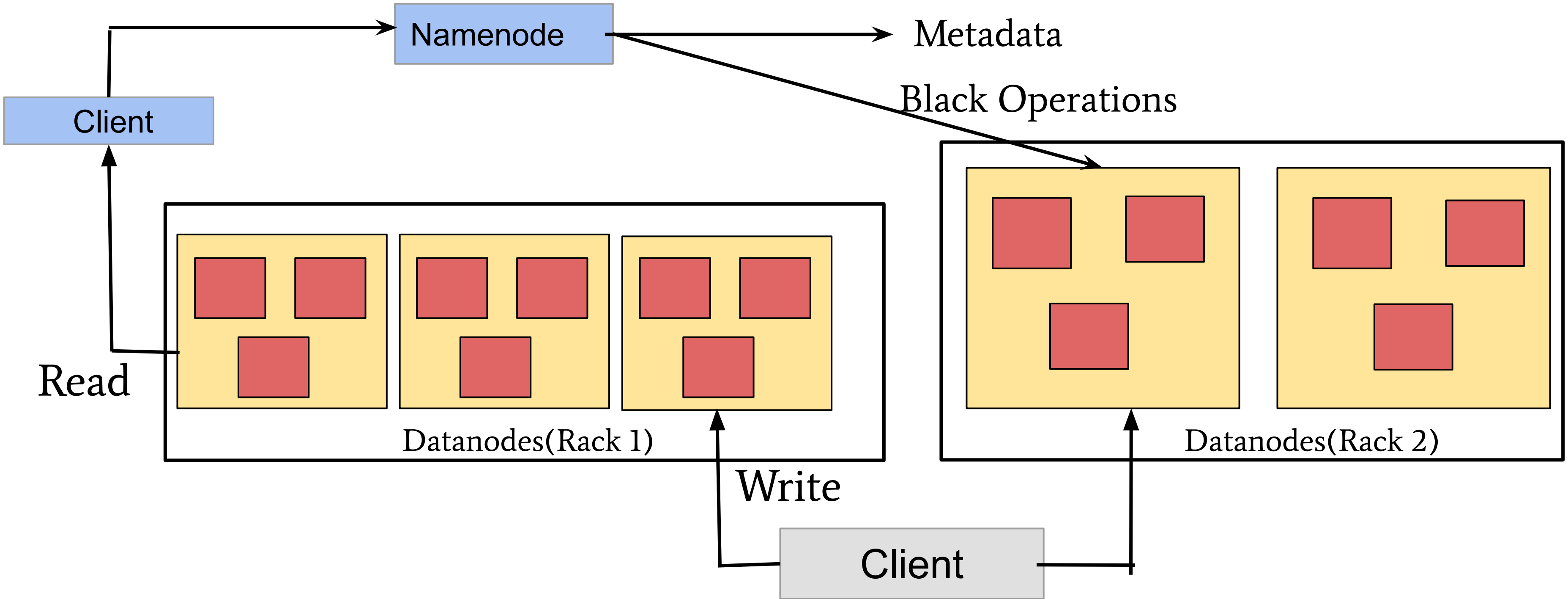
   If one rack fails but all data stored in different rack so data may not loss.

HDFS Read/Write Architecture

HDFS follow write once read many philosophy

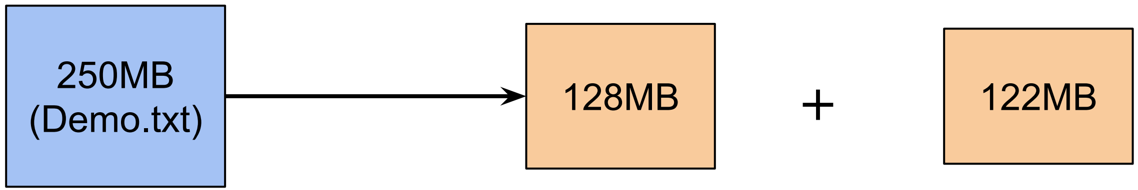
We can't edit files stored in HDFS.

We can append by reopening file.



HDFS Write Architecture

Ex: Client wants to write a file "Demo.txt" with size 250MB



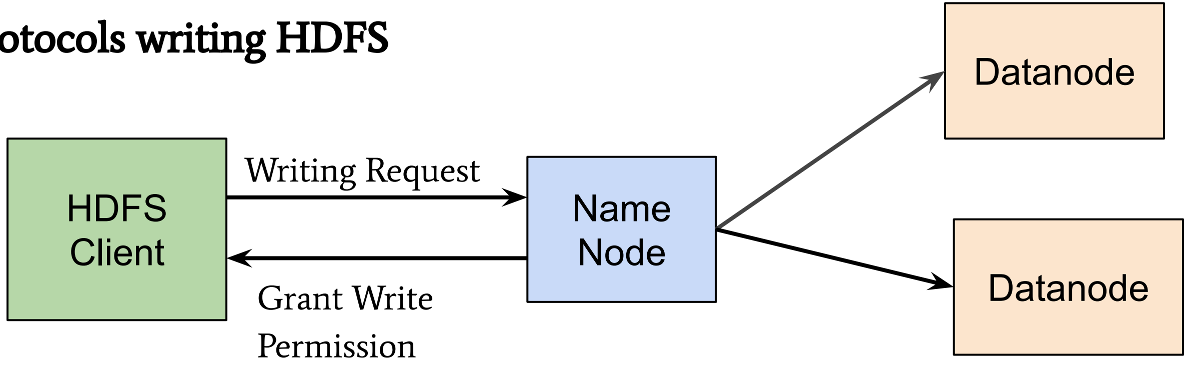
Default Size 128MB

A–128MB

Remaining Size 122MB

HDFS Write Architecture

Protocols writing HDFS



And shares IP Address of Data nodes

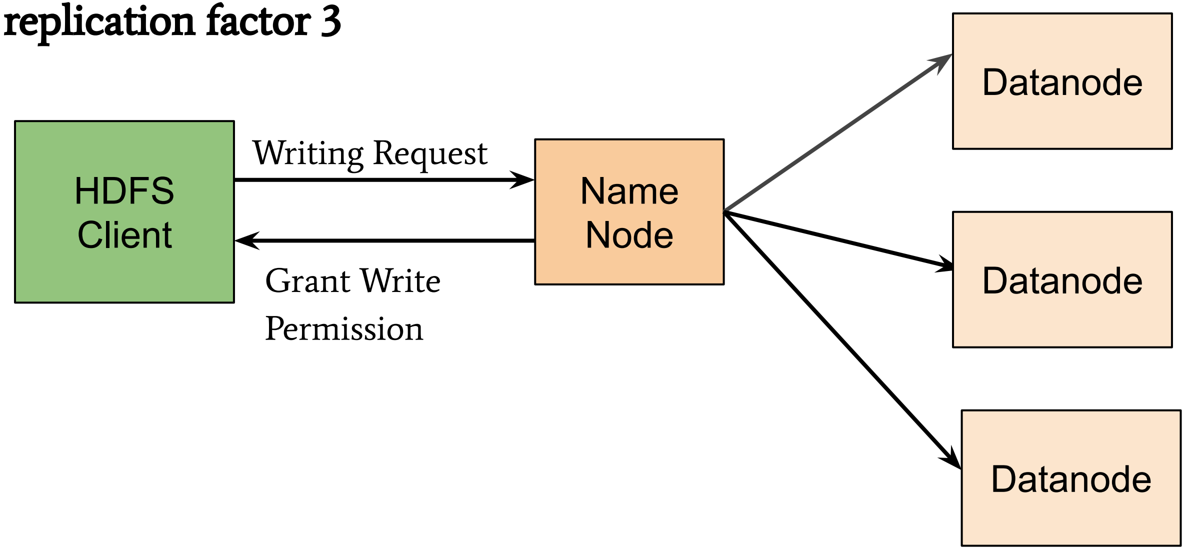
Now Selection of ip address purely randomized based on availability, replication factor , rack awareness

Ex: If replication factor 3

Only replicate 3 —-> Provide 3 Ip addresses of Data nodes

HDFS Write Architecture

If replication factor 3



Namenode provide list to client of 3 Ip address Datanodes

List will be unique for each block

Ex:

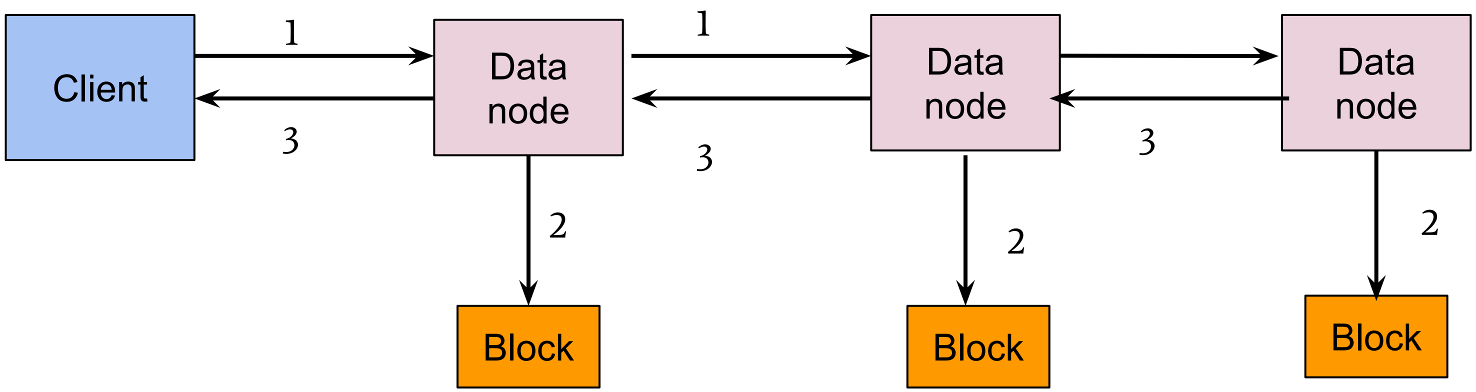
For Block A List A= {IP Address of Datanodes 1, 4, 6}

Block B List B= {IP Address of Datanodes 3, 7, 9}

HDFS Write Architecture

Each block copied in 3 different datanodes, to maintain replication factor.

1. Set up of pipeline
2. Data Streaming & replication
3. Shutdown of pipeline (Acknowledgment)



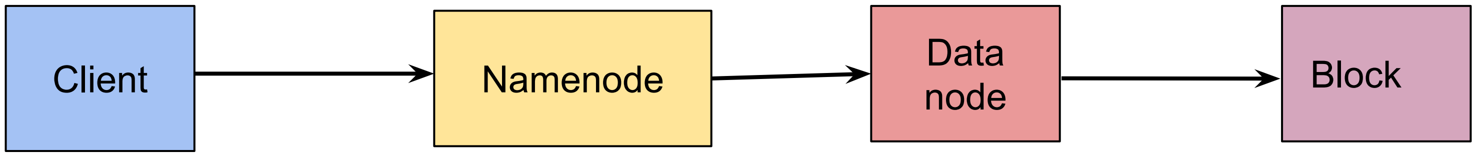
HDFS Write Architecture

Each block copied in 3 different datanodes, to maintain replication factor.

1. **Set up of pipeline:**
   Before writing client confirm whether datanode ready to receive data or not
   client creates pipeline for each block by connecting data nodes
   Whatever client contain that list will connect

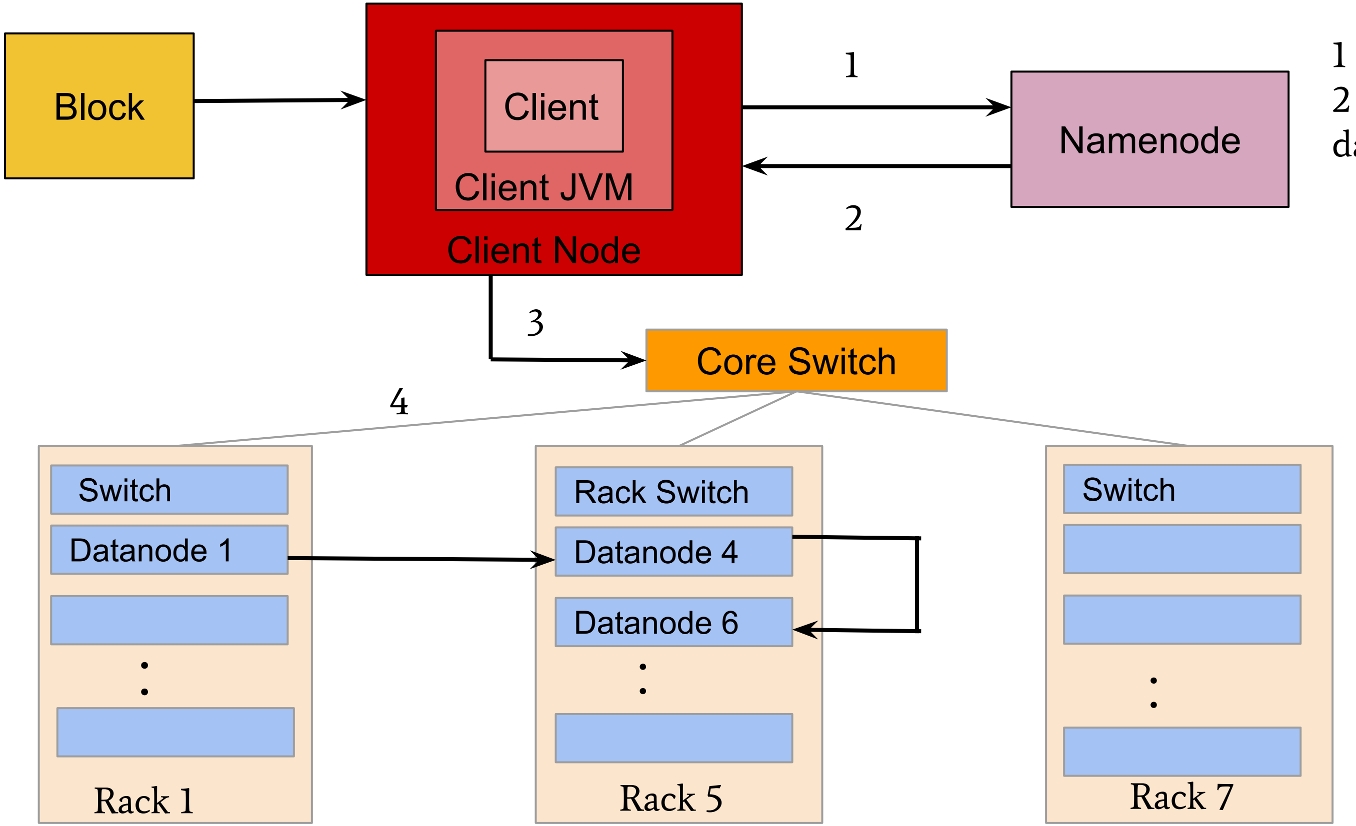
Block A List A= {IP Address of Datanodes 1, 4, 6}

Block B List B= {IP Address of Datanodes 3, 7, 9}



HDFS Write Architecture

Setting up HDFS Write pipeline:



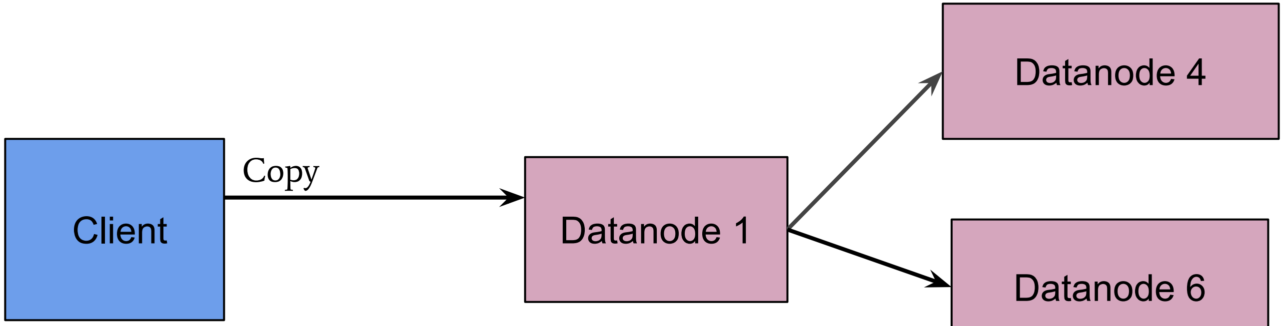
1 Write request for Block A
2 Send Ip Address of datanodes to client

HDFS Write Architecture

2. **Data Streaming:**

After creating pipeline client push data into pipeline, HDFS data is replicated based on replication factor

Block A stores 3 Datanodes as replication factor is 3

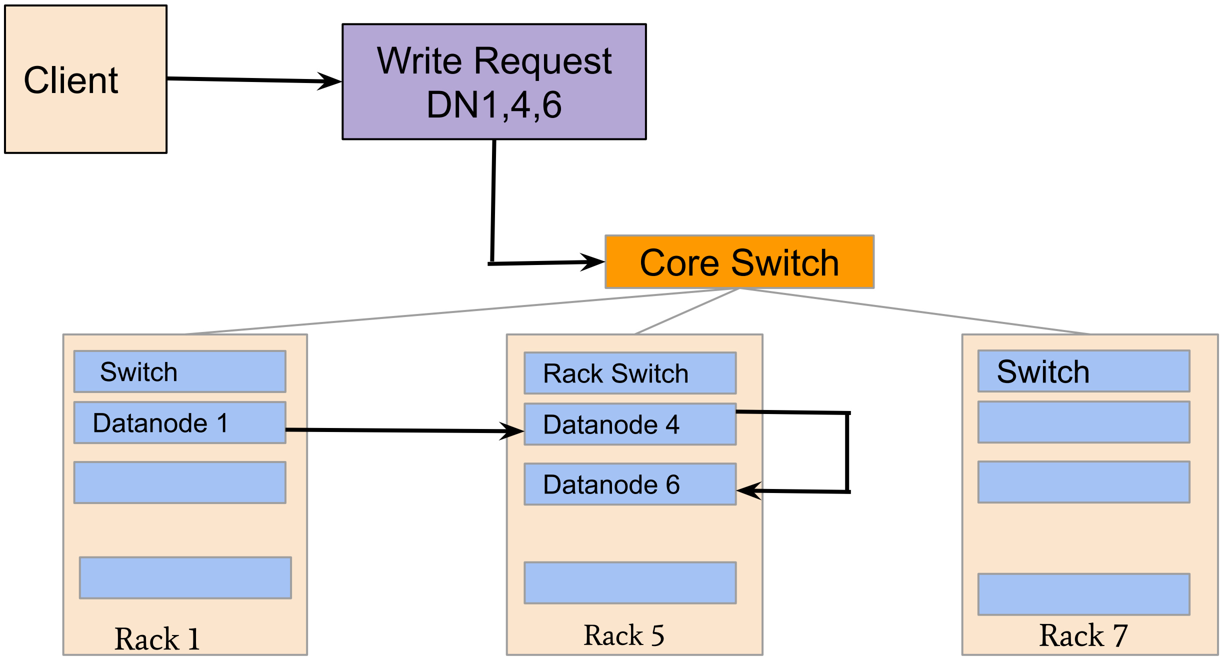


HDFS Write Architecture

2. **Data Streaming:**

Datanode 1 push block in pipeline data copied DN4

DN4 connect to DN6 last replica block

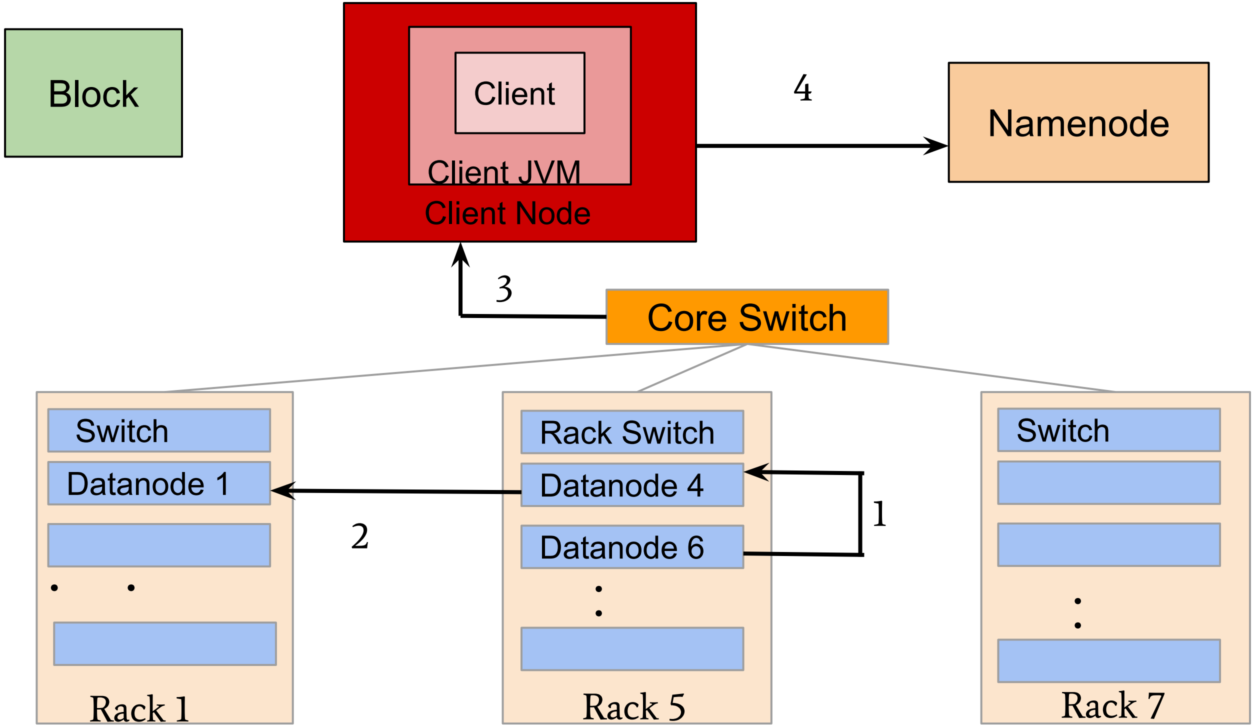


HDFS Write Architecture

3. **Shutdown of pipeline or Acknowledge Stage:**

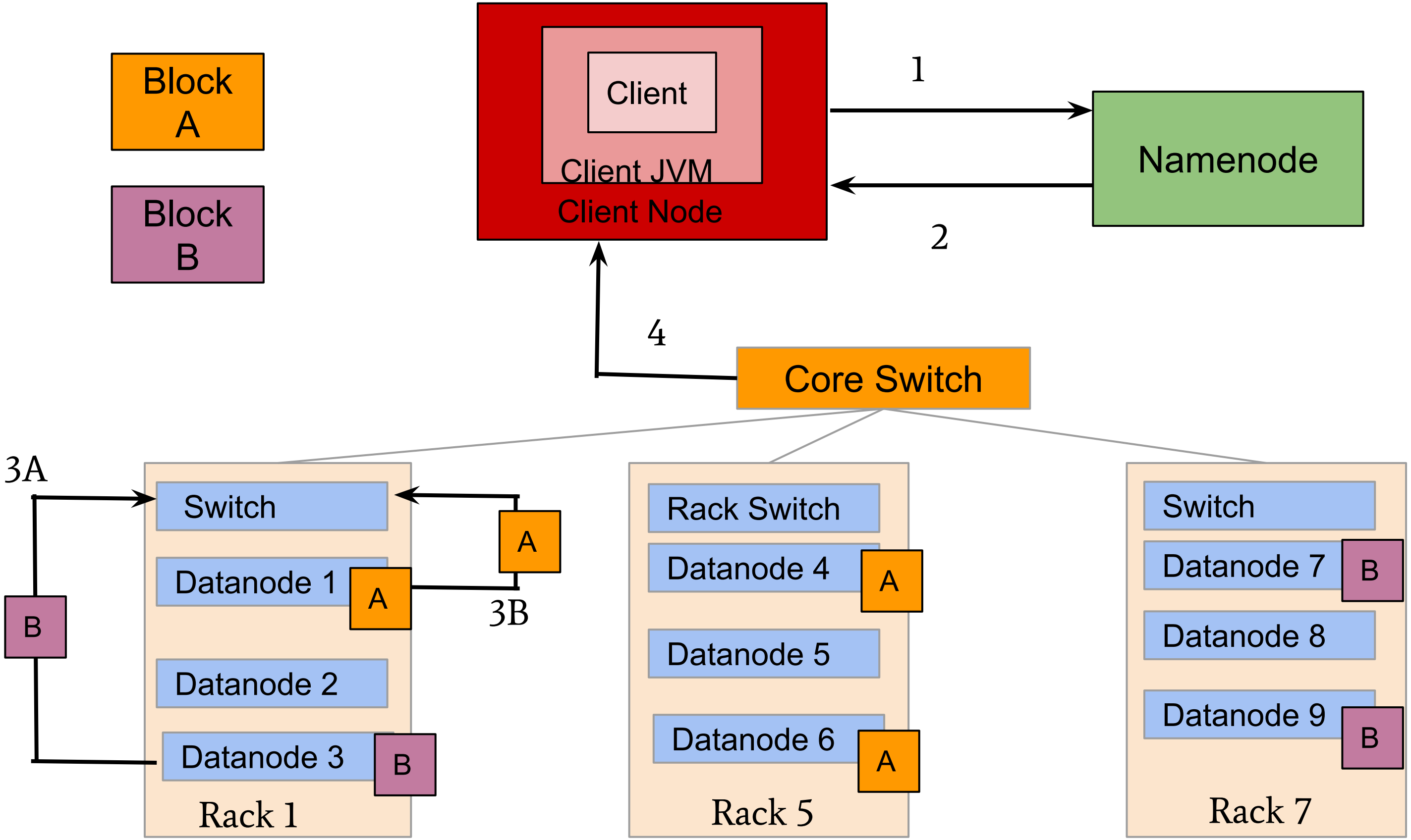
Once block has been copied 3 Datanodes series of Acknowledge takes place.

Acknowledgement—> Data Written Successful



1. Acknowledgment Datanode 6 to Datanode 4
2. Acknowledgment Datanode 4 to Datanode 1
3. Send Ack to Client
4. Write Successful

HDFS Read Architecture



1. Read Request for Bock A,B
2. Send Ip Address of DN1, DN3
3. Reading Data from block A, B
4. Read from DN1, DN3

YARN

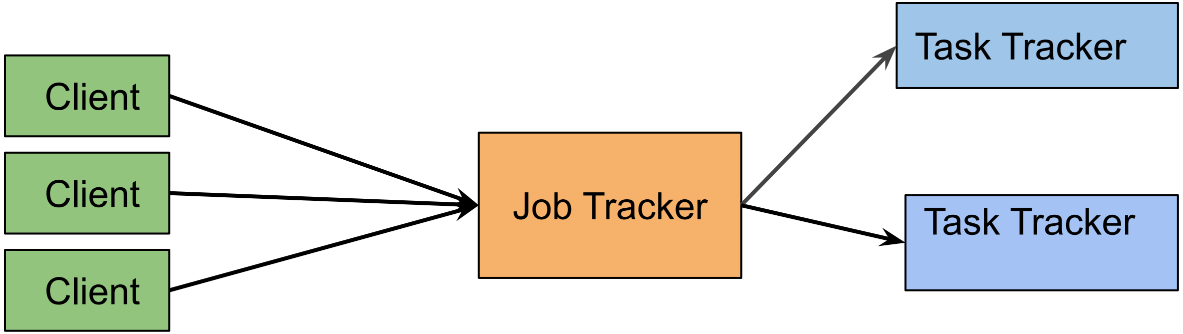
YARN

Yarn- Yet Another Resource Negotiator
Map reduce performed both processing & resource management functions

Job Tracker [Single Master]

1. Allocated the resources
2. Performed Scheduling
3. Monitor the processing jobs

If assigned Map & Rescue task on subordinate process called Task Trackers
Task Trackers: Report their programs to job tracker



YARN

Yarn allows different data processing methods

1. Graph processing
2. Interactive processing
3. Stream processing
4. Batch processing

To run and process Data stored in HDFS

YARN

Hadoop 1.0

Map Reduce Data Processing & Resource Management

HDFS

Hadoop 2.0

Map Reduce

Other Data Processing frameworks

YARN Resource Manager

HDFS

YARN

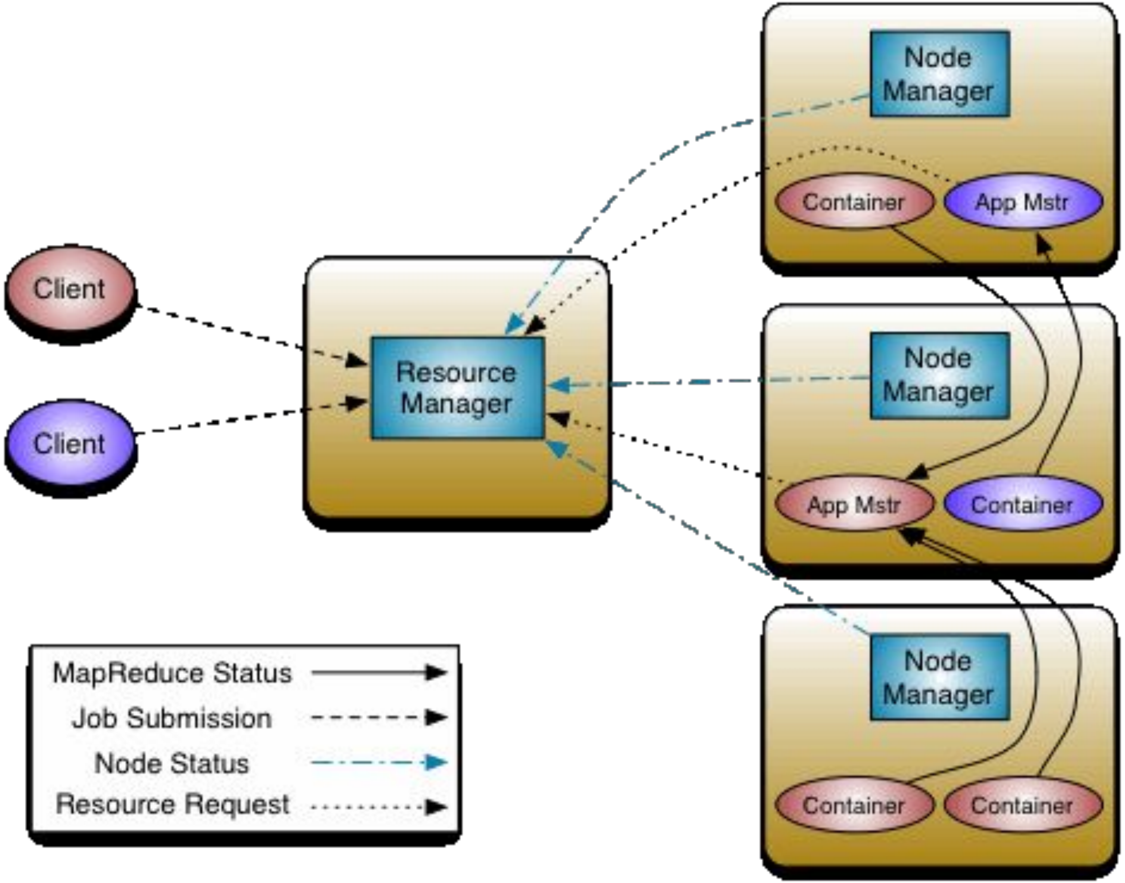
Yarn enable users to perform operations as per requirement

Spark → Real Time Processing
Hive → SQL
HBase → NoSQL

We can use all these in Same cluster

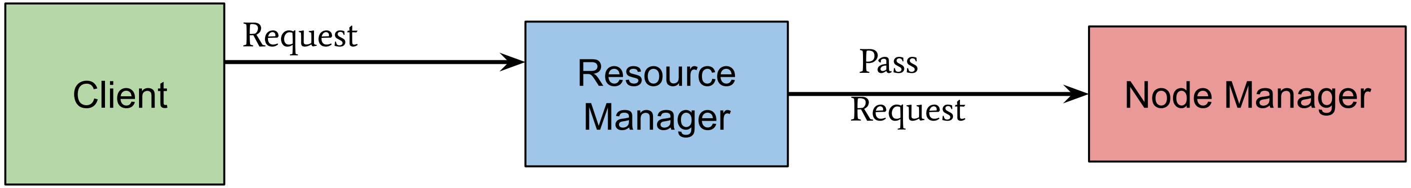
Yarn Performs resource management , Job Scheduling

Main Components of YARN



Main Components of YARN

1. **Resource Manager:**
   Allocate cluster resources using scheduler & Application Manager



Check Available resources to complete applications

Main Components of YARN

1. **Resource Manager:**

   Scheduler :

1. Allocate resources to various running applications
2. Only schedule applications so it called as pure scheduler
3. If any application failure scheduler not guarantee because it doesn't track applications
4. This scheduling performed based on resource requirement
5. **Pluggable Policy:** Responsible for partitioning the cluster resources on various applications.

Main Components of YARN

1. **Resource Manager:**

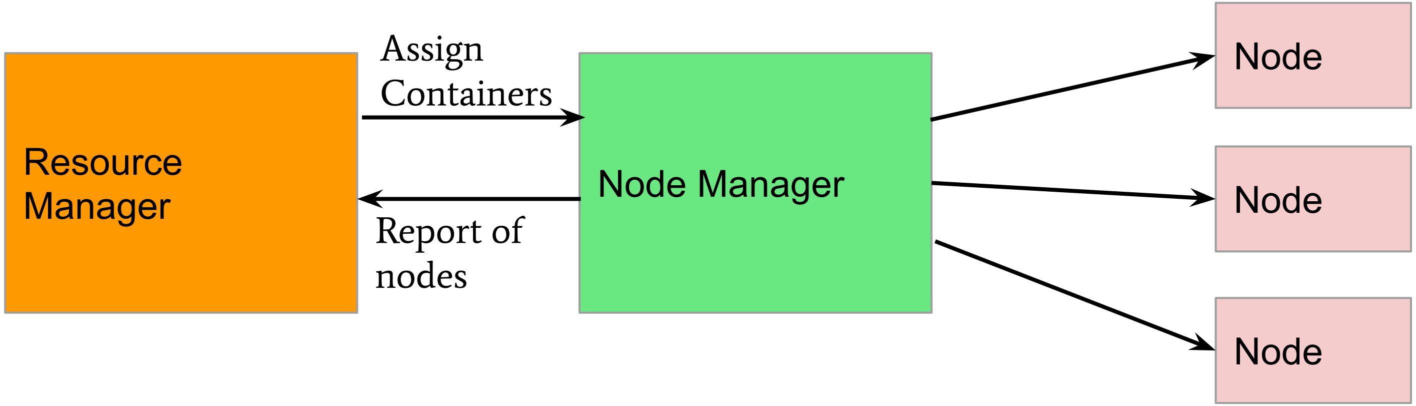
   Application Manager :

1. Responsible for accepting job submission.
2. Take first container from resource manager for executing applications.
3. Manage running application master in a cluster & provide service for restarting application master container on failure.
4. Resource manager optimize cluster utilization like keeping all resources in use.

Main Components of YARN

2. Node Manager:

1. Takes care of all individual nodes in a cluster and manages user jobs



Primary goal to manage application containers assigned by resource manager

Main Components of YARN

2. Node Manager:

Application Master requests the assigned container from the Node Manager by sending it a Container Launch Context(CLC)

Node manager creates requested container process and starts it

Perform log management

Ìt kills container as directed by resource manager

Main Components of YARN

3. Application Master:

1. Application Master is a Single Job submitted to framework
2. Every Application have a unique application master associated with framework
3. Manage faults in applications execution in cluster

Main Tasks:

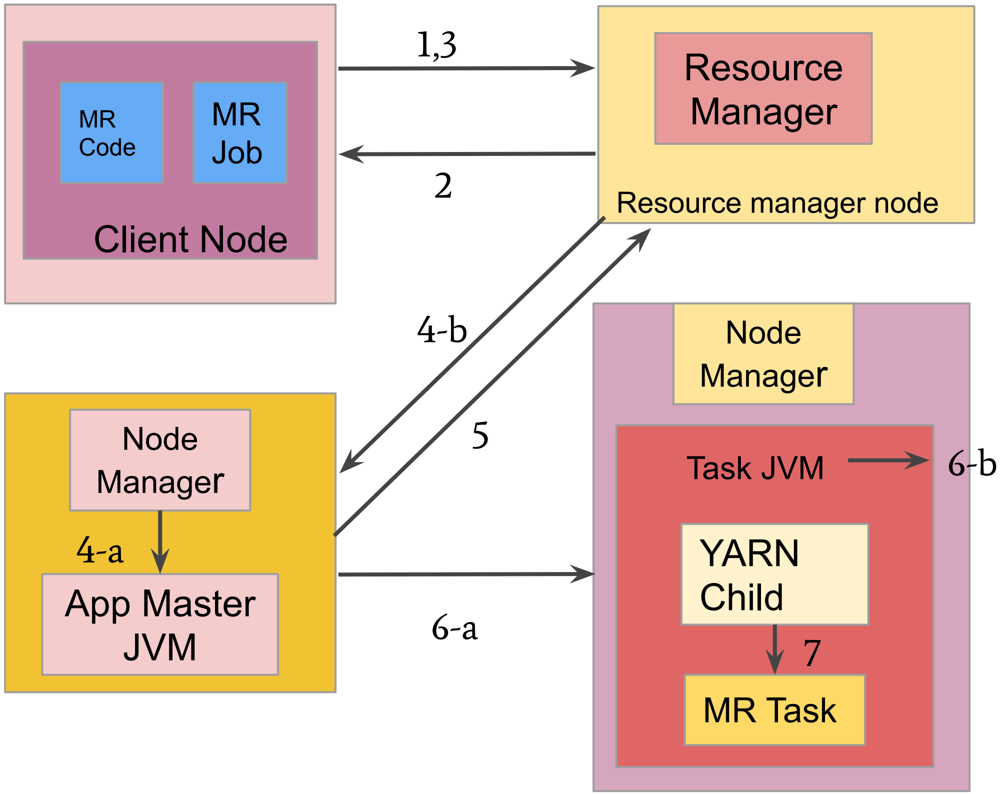
- Negotiate resources from Resource manager and works with node manager & monitor tasks
- Its should take appropriate containers from Resource Manager
- Track & monitor containers status
- Updates pass to resource manager

Main Components of YARN

4. Container:

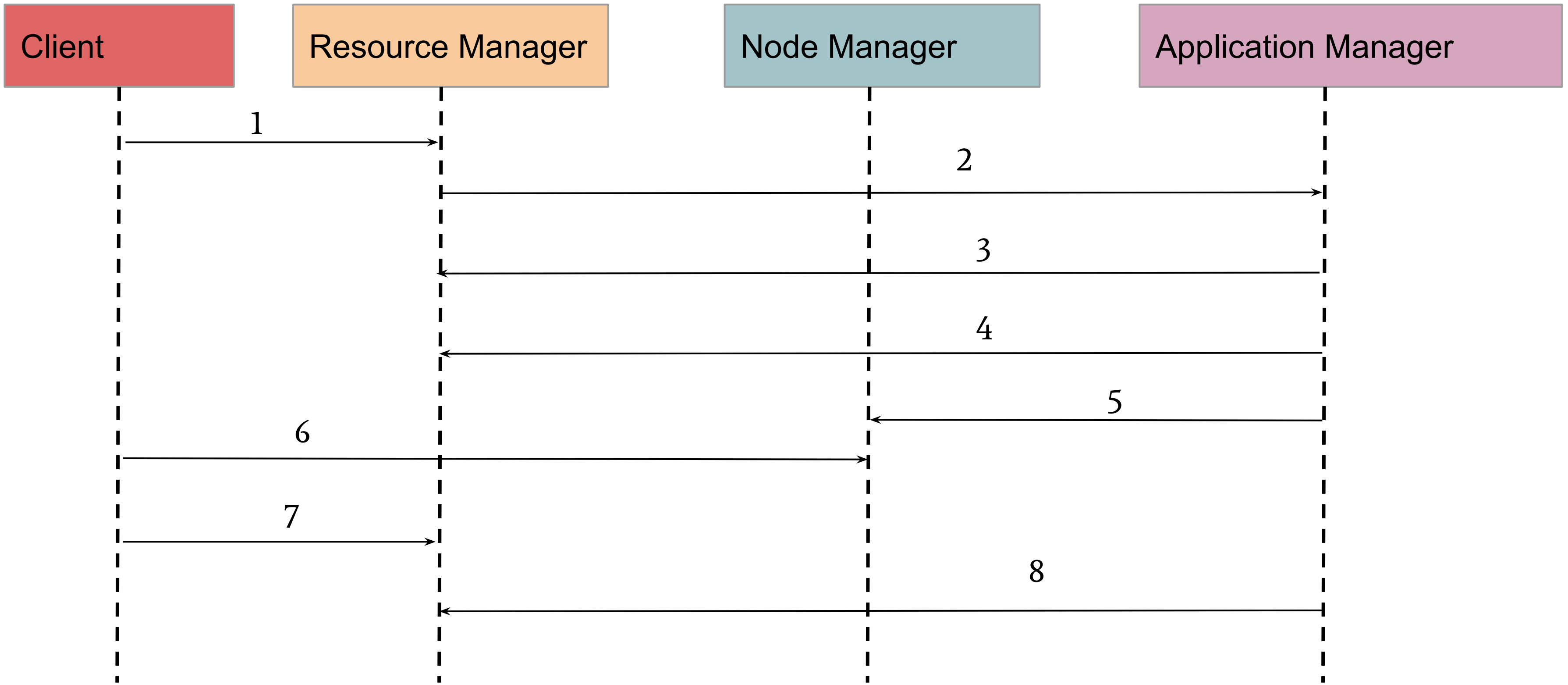
1. Collection of physical resources[RAM, CPU, Disks on Single node]
2. Yarn containers managed by CLC(Container Launch Context or Container Life Cycle)
3. CLC contains map of environment variable dependencies stored in remotely accessible storage, Security tokens, Payloads for Node manager.
4. Grant rights to application to use specific amount of resources on specific host

Application submission steps in YARN



1. Submit the Job
2. Get Application ID
3. Application submission context
4. (a) Start Container launch
   (b) Launch Application Master
5. Allocate Resources
6. (a) Container
   (b) Launch
7. Execute

Application Workflow of YARN



Application Workflow of YARN

1. Client submit an application
2. Resource Manager allocates a container to start application manager
3. Application manager register with resource manager
4. Application manager ask container from resource manager
5. Application manager notifies Node manager to launch containers
6. Application code executed in container
7. Client contact resource manager [Application manager to monitor application status]
8. Application manager unregister with resource manager

hadoop

Map Reduce

Map Reduce

Core building block of processing in hadoop.

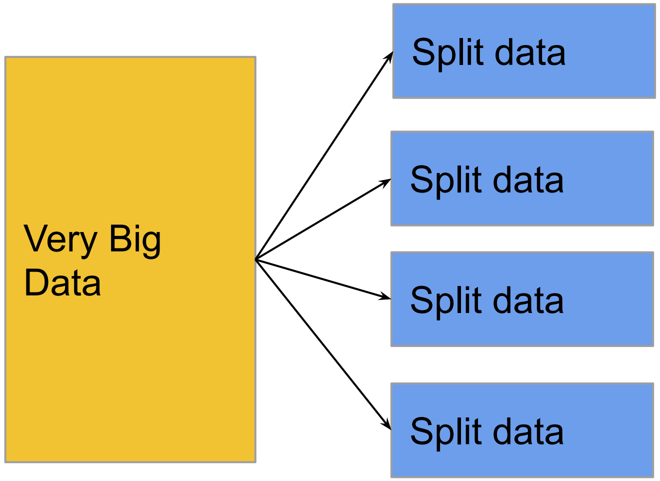
Traditional Way:

Very Big Data

Map Reduce

Core building block of processing in hadoop.

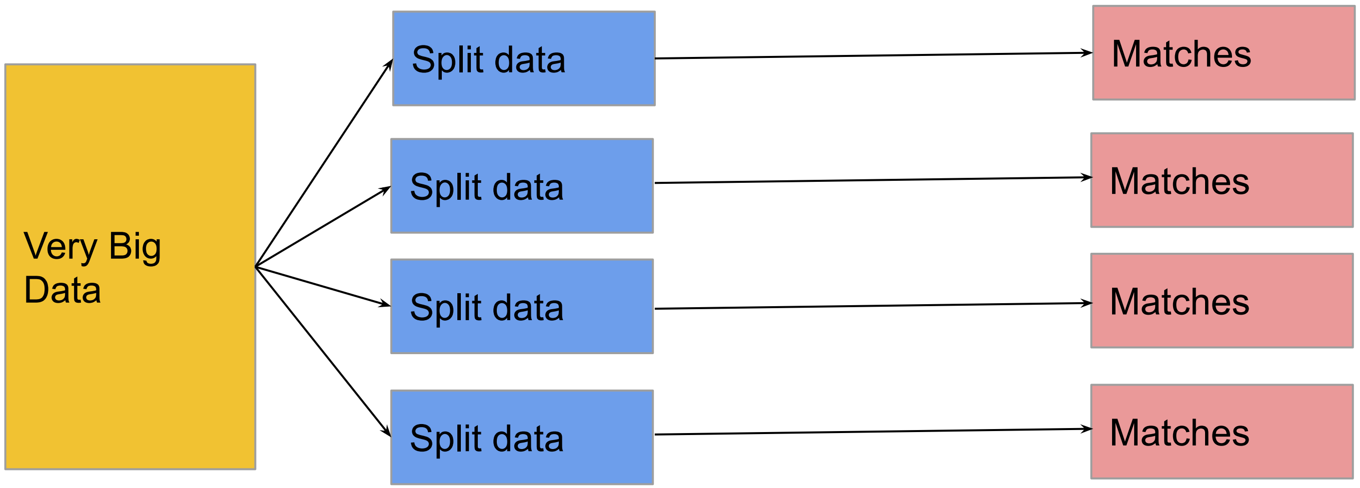
Traditional Way:



Map Reduce

Core building block of processing in hadoop.

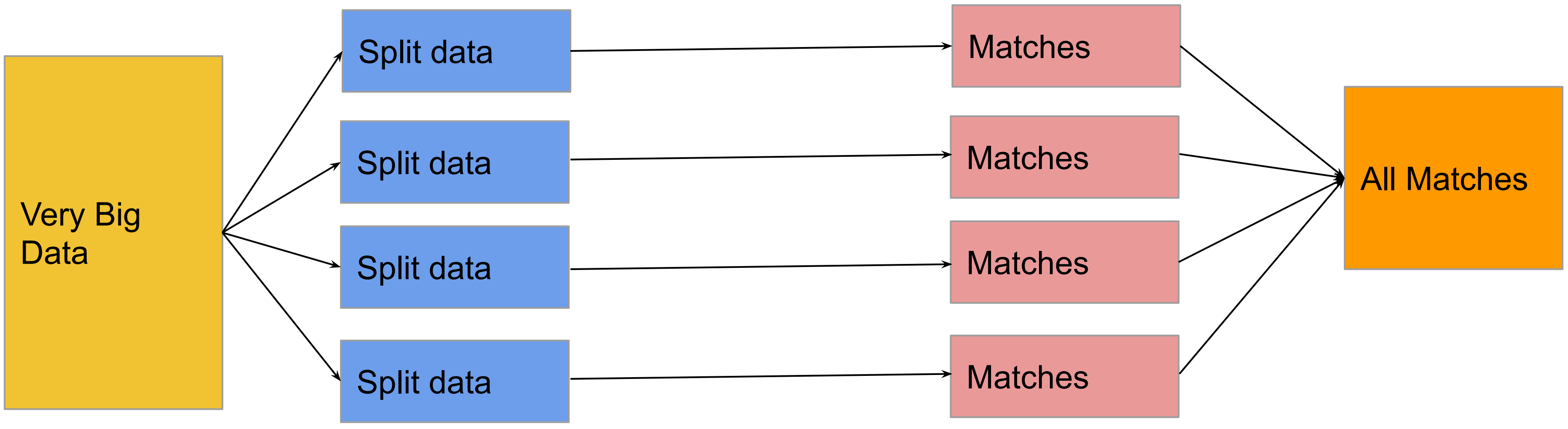
Traditional Way:



Map Reduce

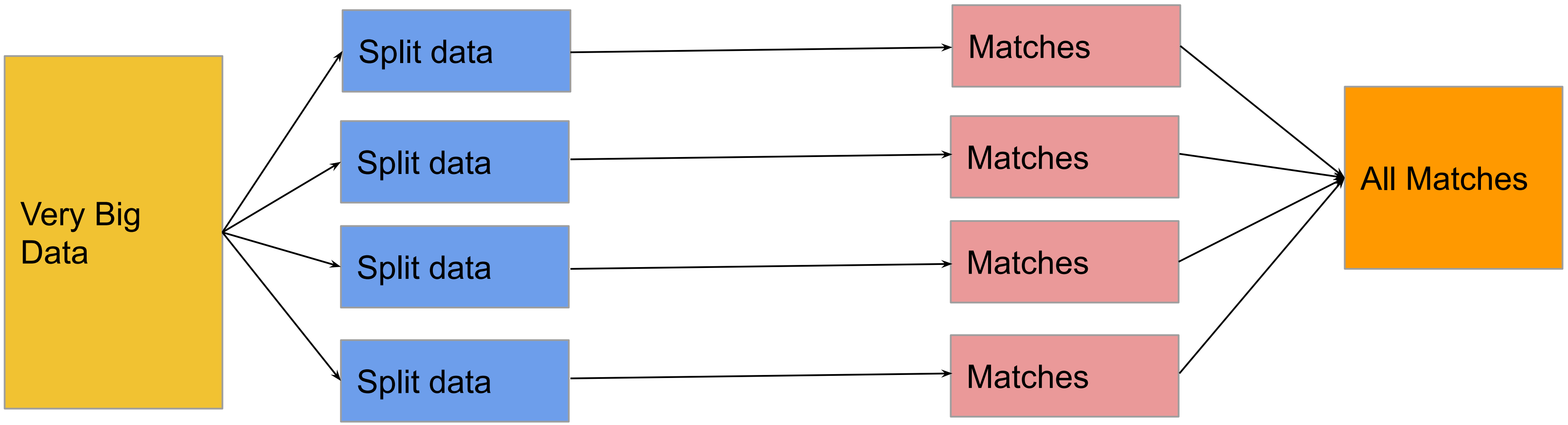
Core building block of processing in hadoop.

Traditional Way:



Map Reduce

Example:
2000-2015 Weather Report
Qn: Find Highest Temperature day in year

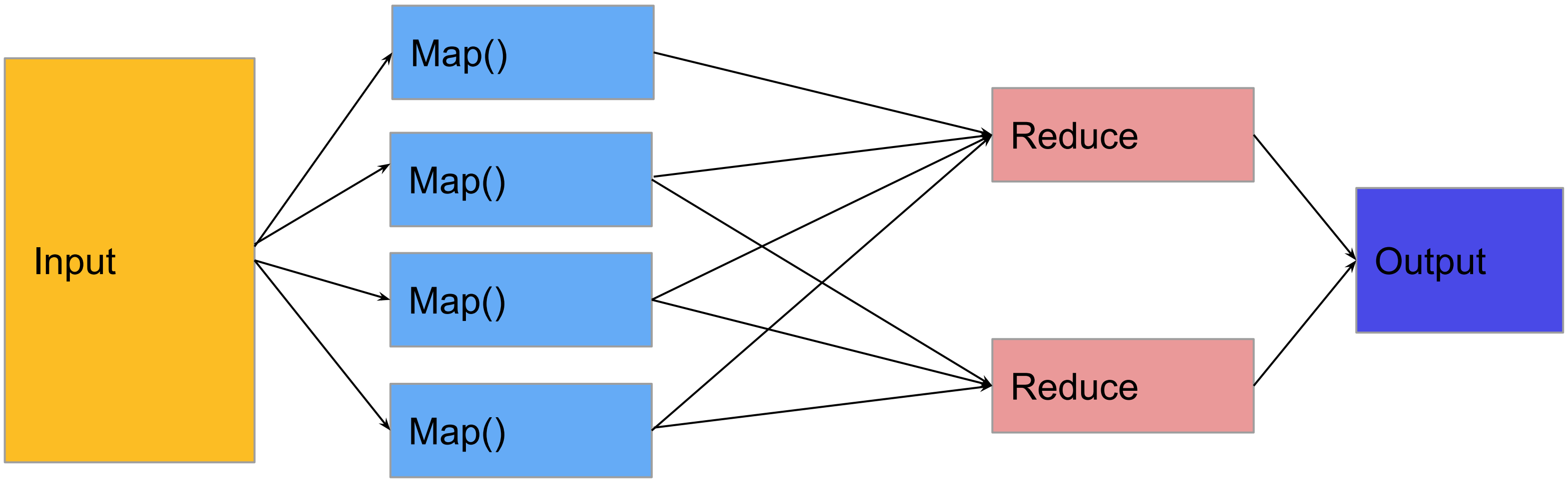


Map Reduce

Challenges with traditional way:

1. **Critical Path Problem:**
   Time Delay [Time to take to complete job]
2. **Reliability problem:**
   If any machine worked on any fail management of failure become challenge.
3. **Equal Split issue:**
   Each machine gets equal parts of data. Sometimes machines overloaded or underloaded.
4. **The single split may fail:**
   If one machine fails to provide output we can't calculate result
5. **Aggregate of Result:**
   To aggregate final result generated by each machine to produce final output.

MapReduce in Detail

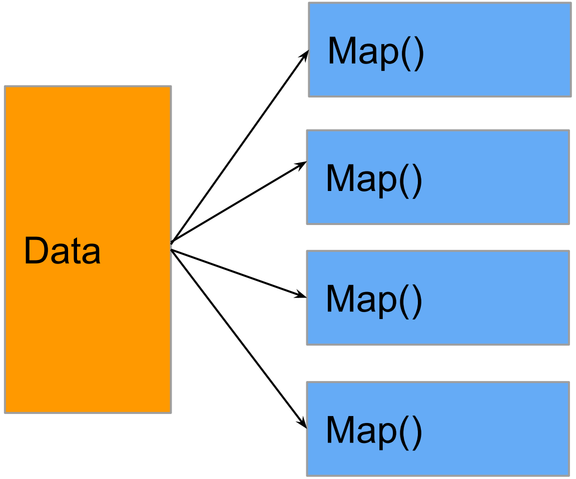


Map Reduce:
Perform Distributed & parallel processing on large data.
Map Reduce: 2 Taks Map, Reduce

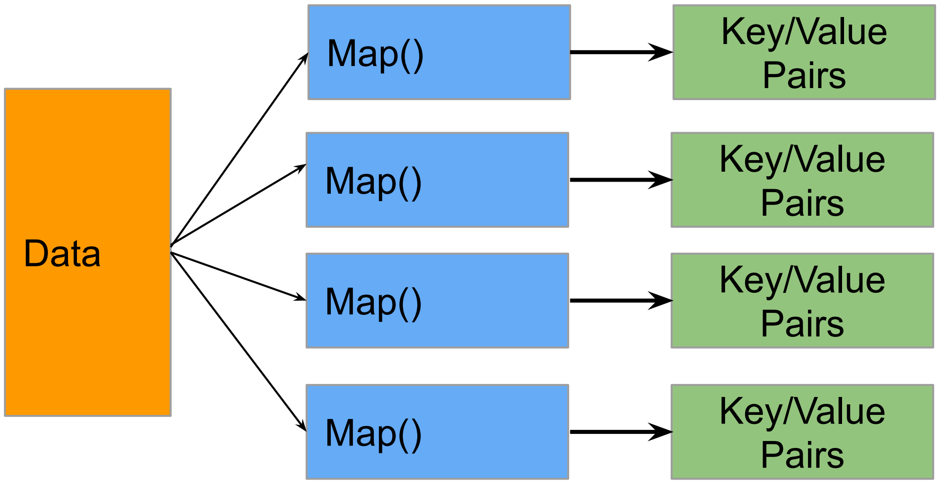
MapReduce in Detail

Data

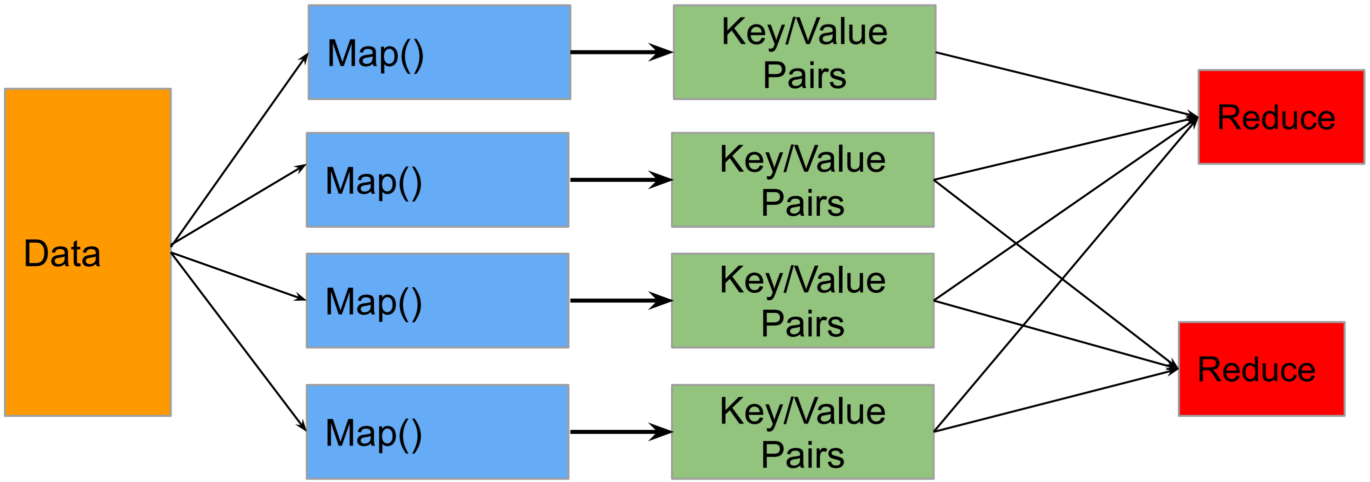
MapReduce in Detail



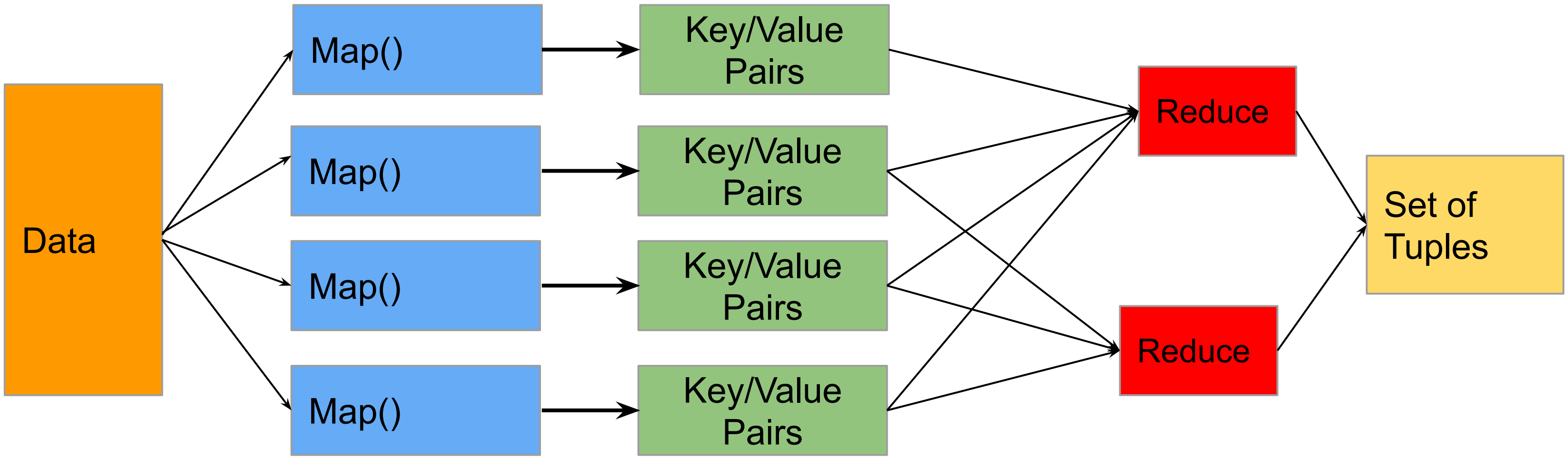
MapReduce in Detail



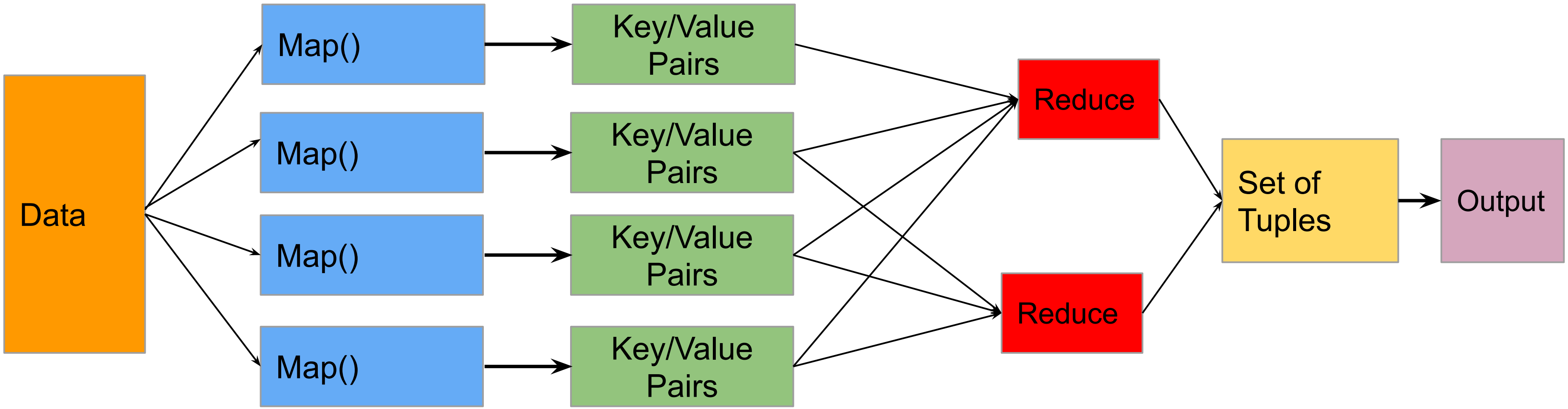
MapReduce in Detail



MapReduce in Detail

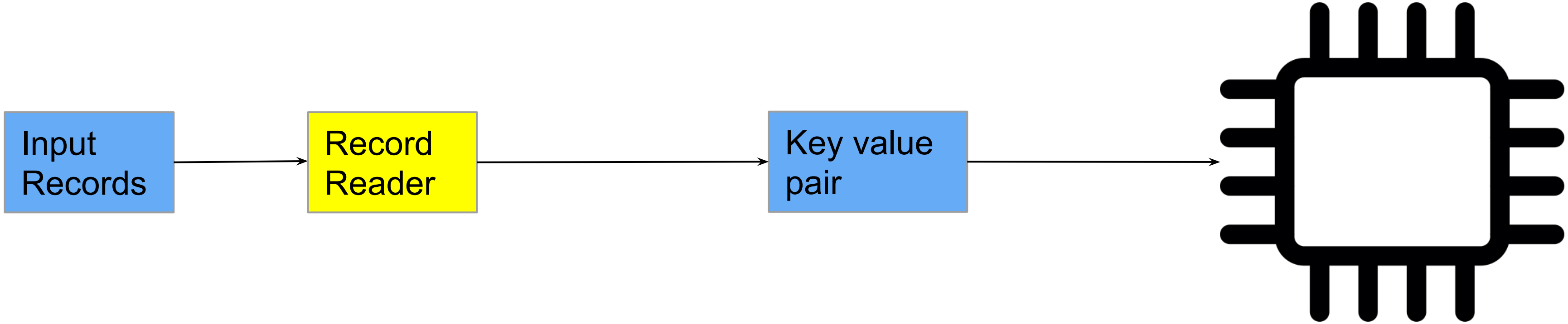


MapReduce in Detail



MapReduce in Detail

Mapper Class():
First stage in Data processing



MapReduce in Detail

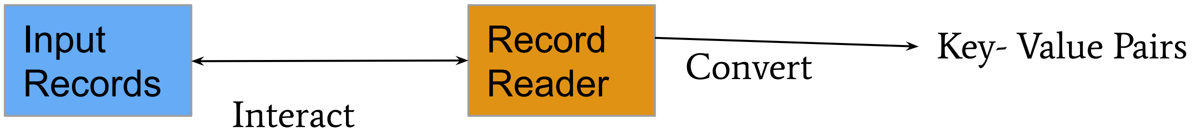
Input Split

It is the logical representation of data.

It represents a block of work that contains a single map task in the MapReduce Program.

RecordReader

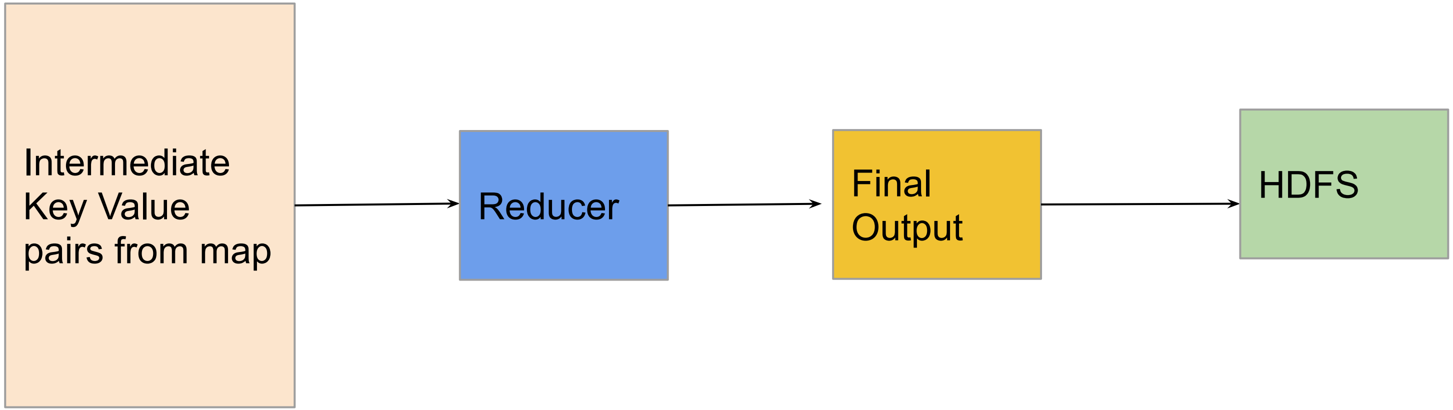
It interacts with the Input split and converts the obtained data in the form of Key-Value Pairs.



MapReduce in Detail

Reducer Class():

First stage in Data processing



MapReduce in Detail

Driver Class():

1. Major Component of Map Reduce
2. Set Mapreduce job to run in hadoop
3. We specify all data related to Mapper & reducer

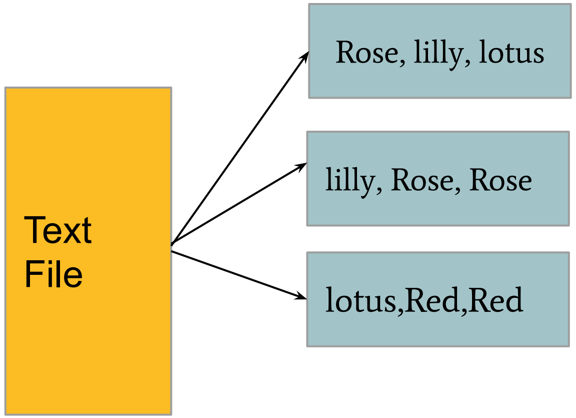
MapReduce in Detail

Example: Word Count using Mapreduce
Text File —--> Rose, lilly, lotus, lilly, Rose, Red,lotus,Rose,Red
Goal: Perform Word Count on Text File

Text File

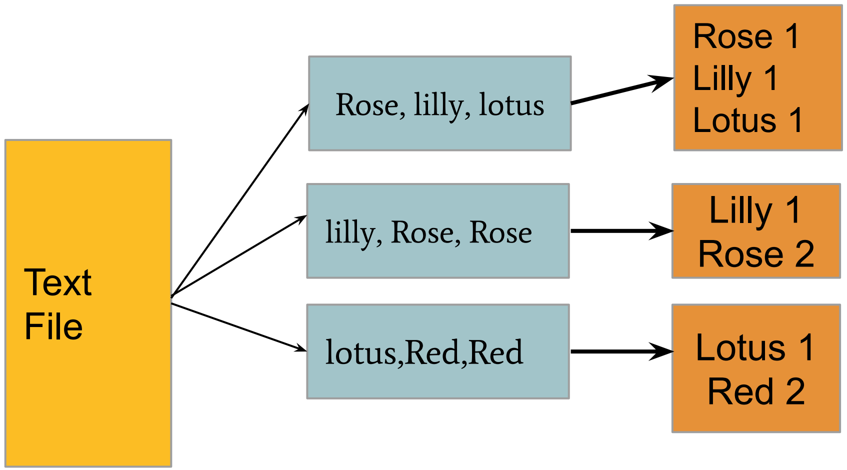
MapReduce in Detail

Example: Word Count using Mapreduce
Text File —--> Rose, lilly, lotus, lilly, Rose, Red,lotus,Rose,Red
Goal: Perform Word Count on Text File



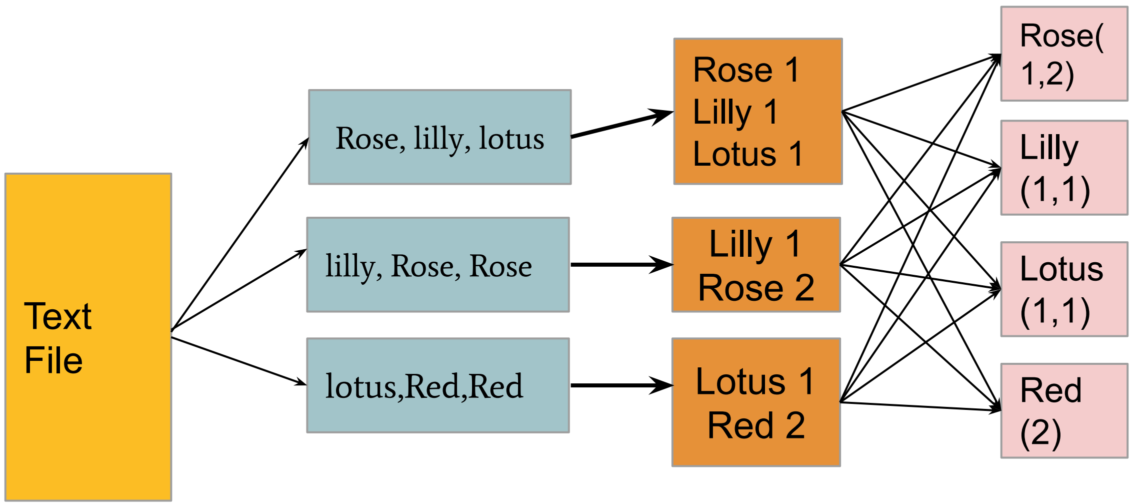
MapReduce in Detail

Example: Word Count using Mapreduce
Text File —--> Rose, lilly, lotus, lilly, Rose, Red,lotus,Rose,Red
Goal: Perform Word Count on Text File



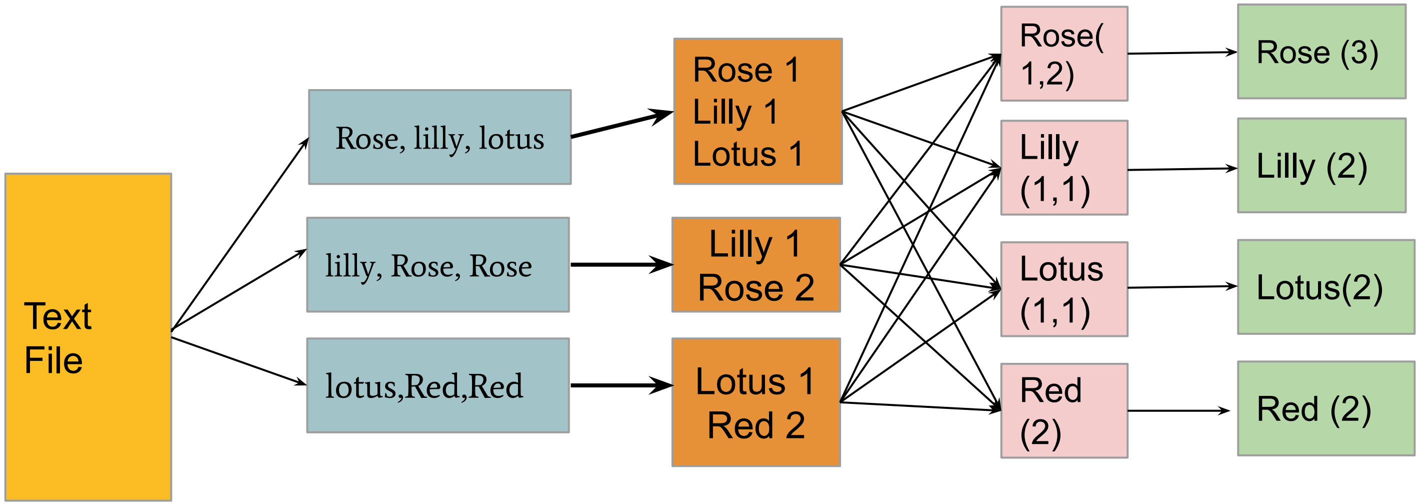
MapReduce in Detail

Example: Word Count using Mapreduce
Text File —--> Rose, lilly, lotus, lilly, Rose, Red,lotus,Rose,Red
Goal: Perform Word Count on Text File



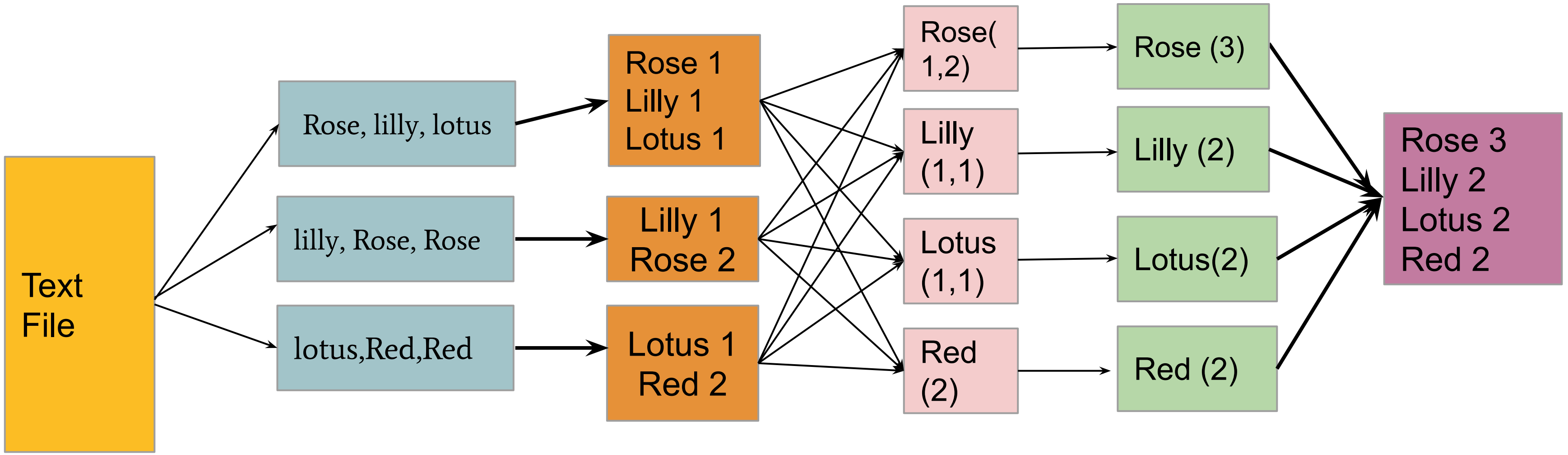
MapReduce in Detail

Example: Word Count using Mapreduce
Text File —--> Rose, lilly, lotus, lilly, Rose, Red,lotus,Rose,Red
Goal: Perform Word Count on Text File



MapReduce in Detail

Example: Word Count using Mapreduce
Text File —--> Rose, lilly, lotus, lilly, Rose, Red,lotus,Rose,Red
Goal: Perform Word Count on Text File



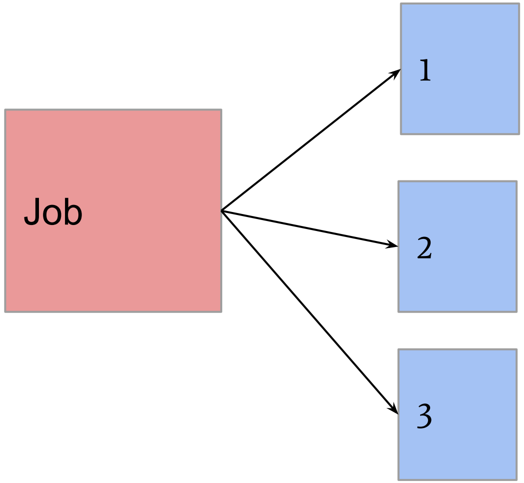
Advantages of MapReduce

1. **Parallel Processing:**

   Dividing job along multiple nodes

   Mapreduce based on divide and conquer method it helps process data on different machines

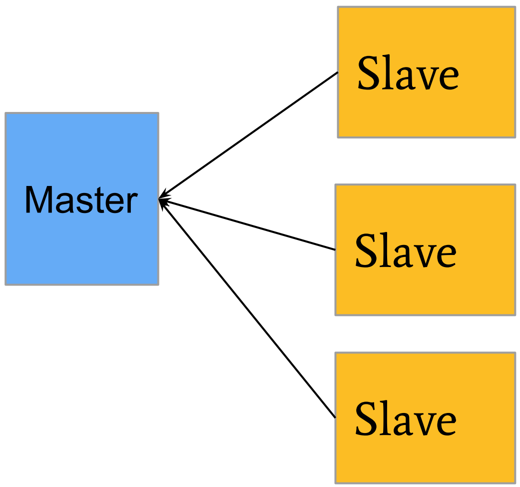
   Data running on different machines instead of single machine it won't take much time.



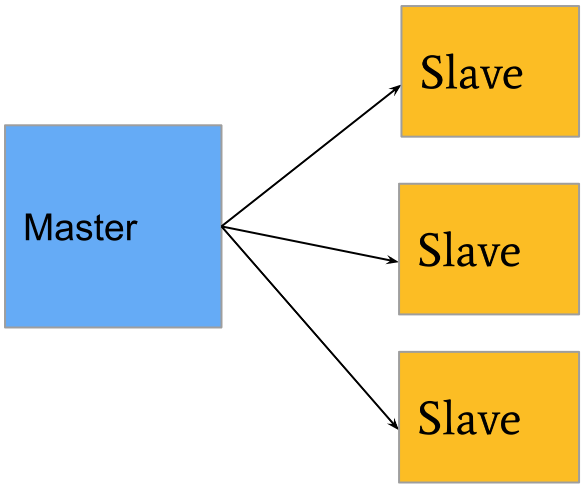
Advantages of MapReduce

Difference Between Traditional & Map Reduce

Moving Data to Processing Unit (Traditional)



Moving Processing Unit to Data (Mapreduce)

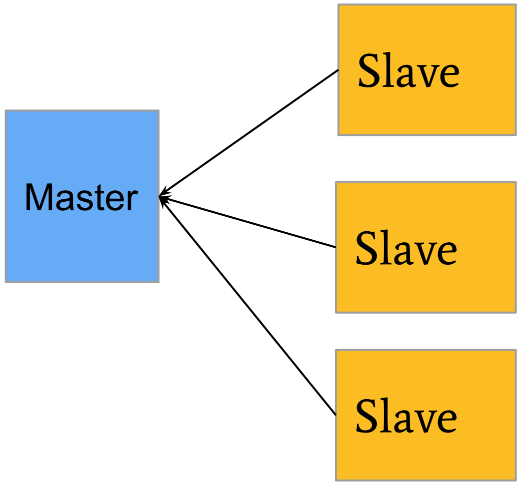


Advantages of MapReduce

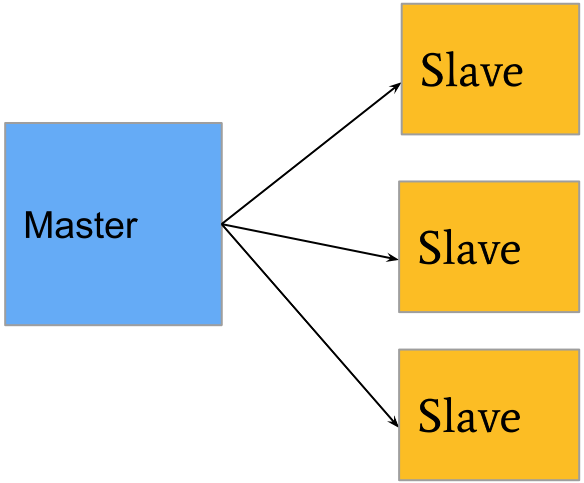
2. Data Locality:

Huge Data cannot bring to processing unit

Moving Data to Processing Unit (Traditional)



Moving Processing Unit to Data (Mapreduce)



Advantages of MapReduce

2. Data Locality:

Issues with processing unit

1. Moving data to processing unit is costly it damage network performance
2. Processing takes time if data processed by single unit
3. Master node can get burdened may fail

Advantages of MapReduce

2. Data Locality:

Mapreduce Overcome these issues:

Data distributed among multiple nodes. Each node process the part of data

Advantages:

1. Cost Effective to move processing unit to data
2. Processing time reduce as all nodes working
3. Every node get part of data no chance to overburdened

APACHE PIG

APACHE PIG

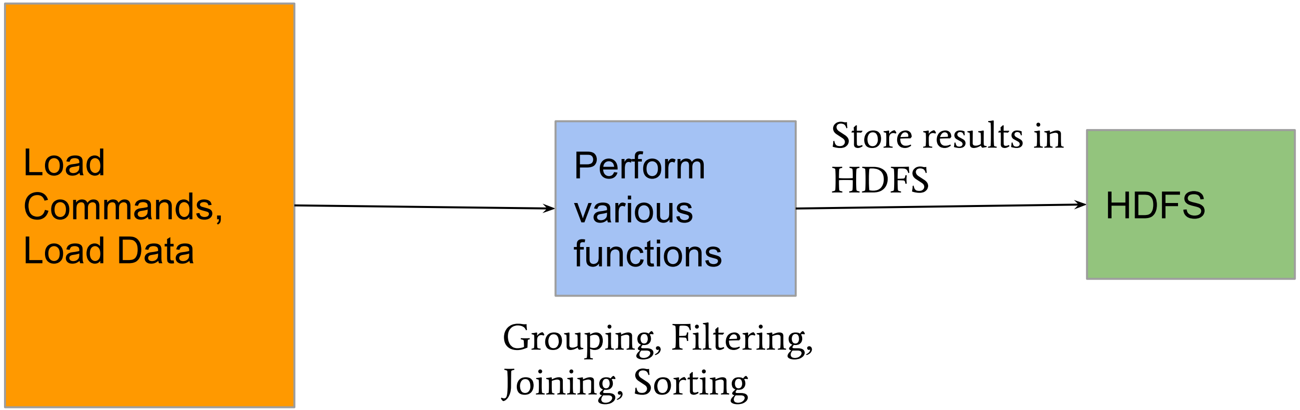
Apache Pig

- 10 lines of pig code= 200 lines of mapreduce code
- Mapreduce written in java
- Pig is not related to python or java
- It is easy to learn
- Pig is 2 parts
  - Pig Latin →Language[Like SQL]
  - Pig runtime→ Every backend of pig job mapreduce job executes
- Compiler internally convert pig latin to mapreduce
- Pig Developed by Yahoo
- It gives platform for building data flow for ETL (Extract, Transform, Load) Processing & analysing huge datasets

APACHE PIG

Apache Pig

How Pig Works:



- We can use pig for huge data, streaming, online data
- We can use for data processing for search platforms
- We can use to process sensitive data

HIVE

APACHE HIVE

Facebook created hive
Hive written in SQL
Easy with Hive while working in hadoop

What is HIVE?
Data warehouse component
Performs Reading,Writing, Managing large datasets in distributed environment using SQL-like interface.

HIVE + SQL = HQL [Hive Query Language]

APACHE HIVE

HIVE have 2 components:

1. HIVE Command Line
2. JDBC/ ODBC Driver– Used to establish connection from database storage

JDBC→ Java Database Connectivity

ODBC→ Object Database Connectivity

Hive is highly scalable

It can serve for large datasets processing (Batch Query Processing)& Real time processing (Intermediate Query Processing)

APACHE Spark™

APACHE SPARK

- Framework for real time processing data analytics in a distributed computing
- Spark written in scala
- Executes in memory computation to increase speed of data processing over mapreduce
- It is 100x faster because of In-memory computation while comparing hadoop
- Spark is pack of all libraries -R, SQL, Python, Scala,Java, Spark SQL+Dataframes,Streaming, MiLib, GraphX

APACHE SPARK

Q) Apache Spark Killer or Saviour of Hadoop
Spark→ Real time processing
Hadoop→ Designed to store unstructured data and processing over it.

Spark Features + Hadoop Features = Low cost Hardware

So this reason so many companies using these technologies together.

APACHE
HBASE

APACHE HBASE

Open Source, Non relational distributed database, NoSQL database.
Support all types of data
Capable to handle anything in hadoop
Modelled after google big table (Distributed Storage System)
HBase was designed to run on top of HDFS
It store small data without fault.
HBase written in java

Ex: Email –Billions of users

APACHE
DRILL

APACHE DRILL

1. Drill to any kind of data
2. Open source application worked on distributed environment to analyze large datasets
3. Replica of Google Dremel
4. Support NoSQL database, File Systems
5. **Ex:** Azure Blob Storage, Google Cloud Storage, HBase, MongoDB, MapR-DB HDFS, MapR-FS, Amazon S3, Swift, NAS and local files.

Drill Scalable

1. We can process petabytes, exabytes of data in minutes
2. Combine various data stores just by single Query.
3. Follow ANSI SQL
4. Powerful scalability factor, serve their query request's over large data

APACHE

ZooKeeper™

APACHE ZOOKEEPER

APACHE
ZooKeeper™

- Coordinates of any hadoop job
- Coordinates with various services
- To coordinate before zookeeper it is time consuming
- Services earlier we have so many issues like common configurations (Services with same names)
- Grouping & Naming also time consuming factor

Main Goals:

1. Synchronization
2. Configuration
3. Maintenance
4. Grouping
5. Naming

OOZiE

APACHE OOZIE

Clock & Alarm service inside hadoop for apache jobs. Oozie is like scheduler
It schedules hadoop jobs and binds them together as on logical work.

2 OOzie Jobs:

1. Oozie workflow:

   Sequence set of actions to executed assume it to relay race

2. Oozie Coordinator:

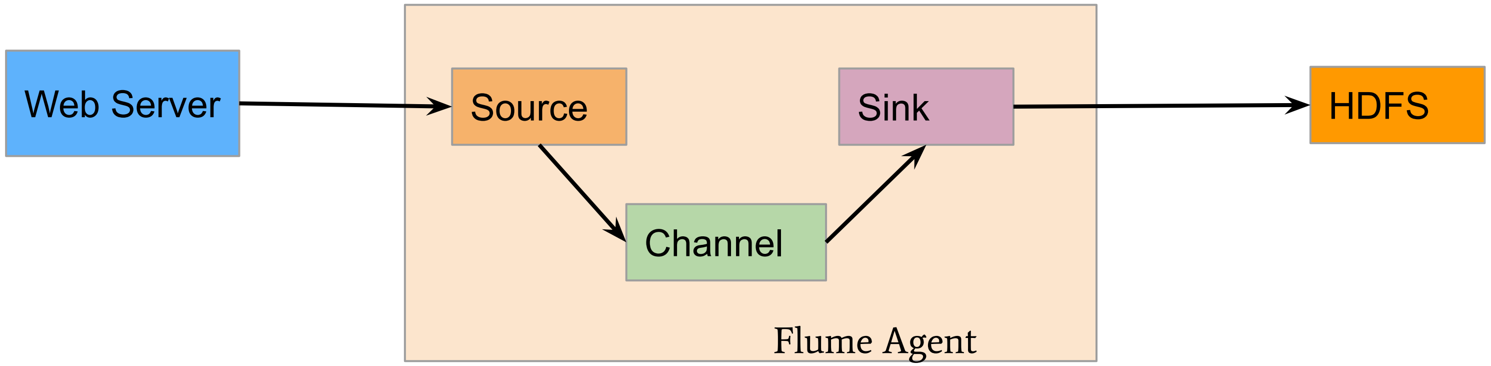
   Jobs triggered when data is made available

FLUME

APACHE FLUME

- Ingesting Data is an important art of hadoop
- Flume helps in ingesting unstructured , semi structured data into HDFS
- It helps in collecting, aggregating, moving large datasets
- Ingest online streaming data from various sources like network traffic social media, email, msgs in HDFS

APACHE FLUME



Flume Agent: Ingests streaming Data from various sources to HDFS
It have 3 components

- **Source:** Accept Data from incoming stream line & stores in channel
- **Channel:** Local Storage or primary storage, Channel is temporary storage between source data & persisted HDFS data
- **Sink:** Collects data from channel & commits or writes the data in HDFS

SQOOP

APACHE SQOOP

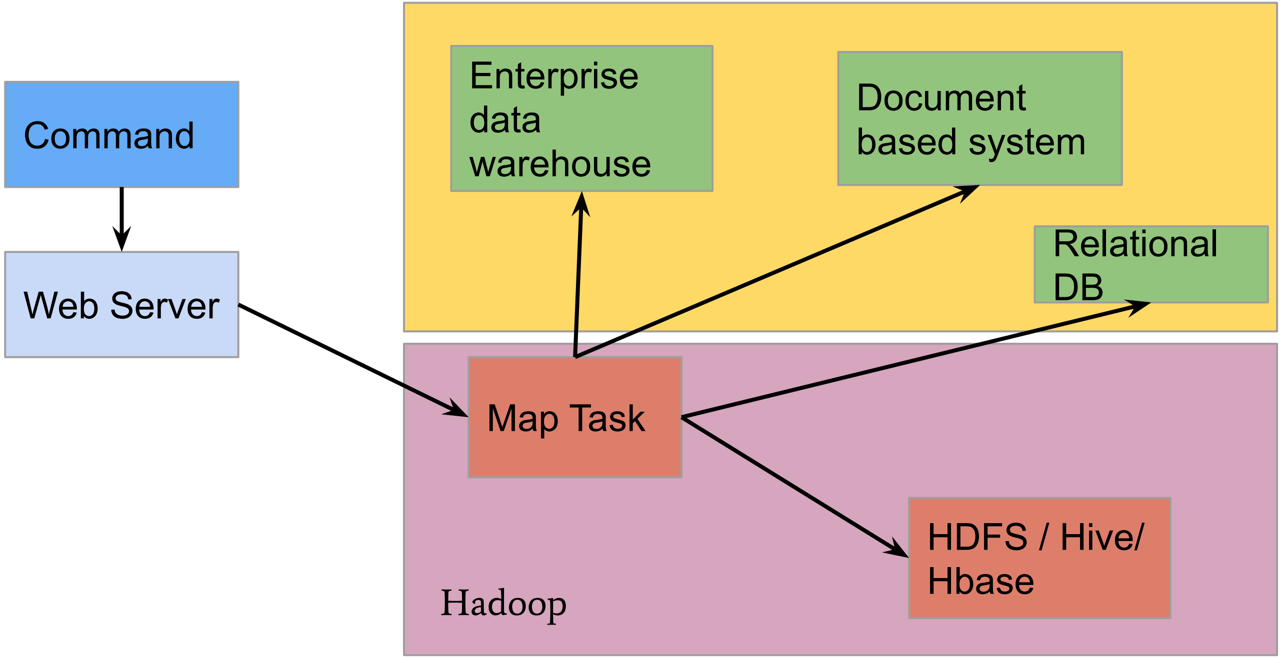
Data ingest service

Major Difference between Flume & Sqoop:

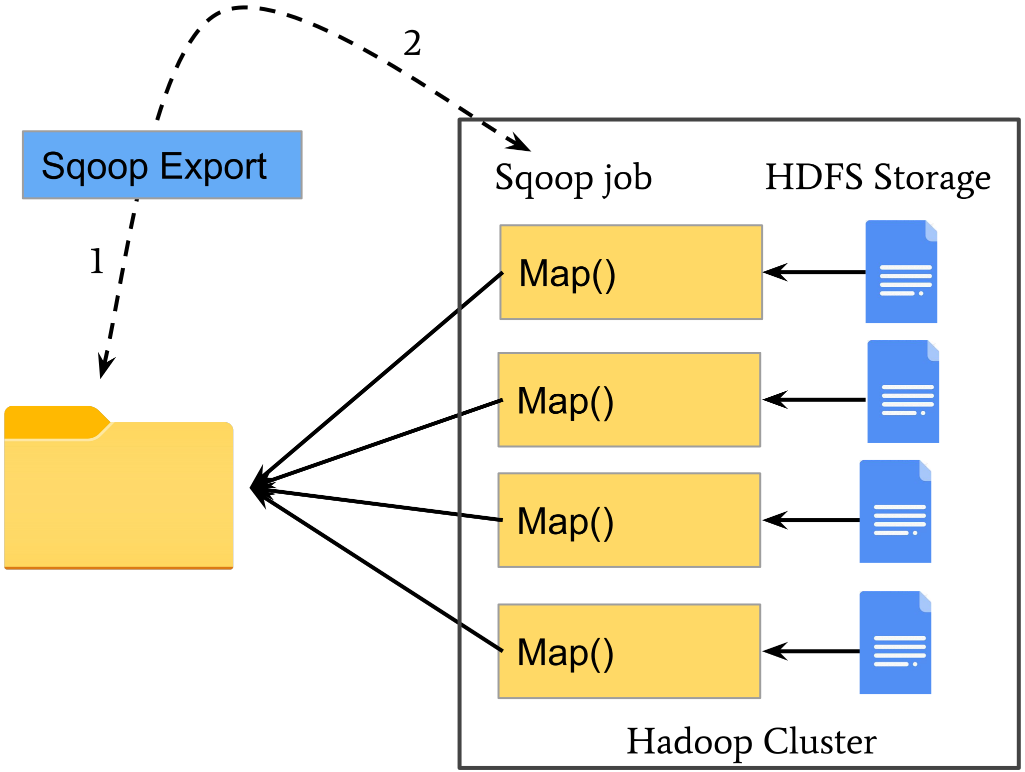
1. Flume only ingest Unstructured, Semi-structured data into HDFS
2. SQOOP can import as well as export structured data from RDBMS or Enterprise data warehouse to HDFS (vice versa)

APACHE SQOOP

When we submit Sqoop command, our main task gets divided into sub tasks which is handled by individual Map Task internally. Map Task is the sub task, which imports part of data to the Hadoop Ecosystem. Collectively, all Map tasks imports the whole data.



APACHE SQOOP



1 Gather Metadata
2. Submit map only job

mahout

APACHE MAHOUT

Renowned for Machine Learning
Provides environment for creating machine learning applications
Mahout performs

1. **Collaborative filtering:**
   Mines user behaviour,pattern, character based on this predicts and make recommendations to users
   Ex : E-commerce Websites
2. **Clustering:**
   Organize similar group of data together
   Ex: Blogs, Articles, News
3. **Classifier:**
   Classifying and categorizing data
4. **Frequent item set missing:**
   Which object likely appearing & make suggestions, if they are missing.
   Ex: Search for Mobile, suggests back cases

THANK YOU