Data Analytics (IT-3006)

Kalinga Institute of Industrial Technology
Deemed to be University
Bhubaneswar-751024

School of Computer Engineering

3 Credit

Lecture Note – Unit 5

Course Contents

Sr #	Major and Detailed Coverage Area	Hrs
5	**Frameworks and Visualization**	8
	Introduction to framework and Visualization, Introduction to Hadoop, Core Components of Hadoop, Hadoop Ecosystem, Physical Architecture, Hadoop Limitations, ~~Hive~~, ~~MapReduce and The New Software Stack~~, MapReduce, ~~Algorithms Using MapReduce~~, NOSQL, NoSQL Business Drivers, NoSQL Case Studies, NoSQL Data Architectural Patterns, Variations of NoSQL Architectural Patterns, Using NoSQL to Manage Big Data, Visualizations	

Introduction

- Huge volume of unstructured data are produced by heterogeneous scenarios in various applications from scientific computing to social networks, from e-government applications to medical information systems and so on, have to be stored in big data stores and analyzed in order to derive intelligence and extract useful knowledge from them.
- There is a need for good visualization of the results produced from analytic engines.
- Visualization issues are a big problem in data warehousing and OLAP research.
- These issues get multifold in the context of big data analytics, as visualization is expected to give a stronger decision-support value.
- Data visualization helps to tell stories by curating data into a form easier to understand, highlighting the trends and outliers.

Hadoop

- Hadoop is an open-source project of the Apache Foundation.
- Apache Hadoop is written in Java and a collection of open-source software utilities that facilitate using a network of many computers to solve problems involving massive amounts of data and computation.
- It provides a software framework for distributed storage and processing of big data and uses Google's MapReduce and Google File System as its foundation.

Hadoop
Apache open-source software framework

Inspired by:
- Google MapReduce
- Google File System

Hadoop provides various tools and technologies, collectively termed as Hadoop ecosystem, to enable development and deployment of Big Data solutions. It accomplishes two tasks namely i) Massive data storage, and ii) Faster data processing.

Data Challenges

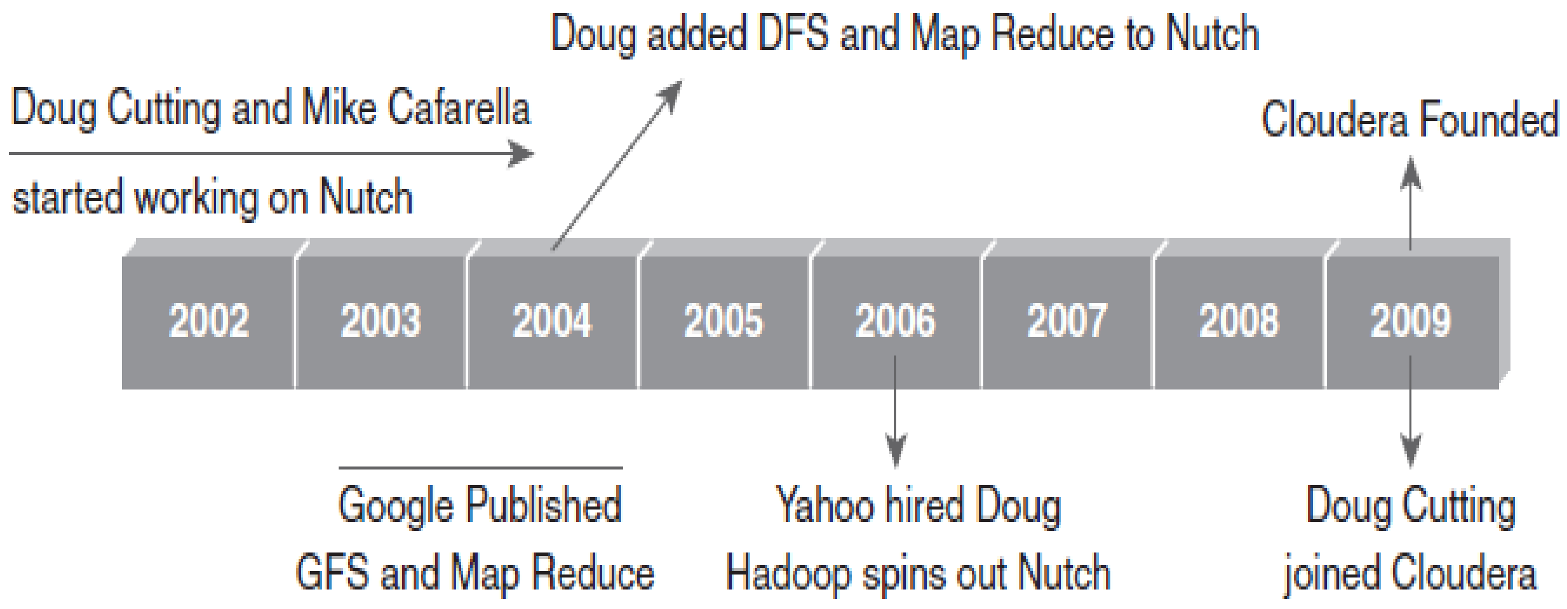
To process, analyze and made sense of these different kinds of data, a system is needed that scales and address the challenges as shown:

"I am flooded with data". How to store terabytes of mounting data?

"I have data in various sources. I have data that rich in variety – structured, semi-structured and unstructured". How to work with data that is so very different?

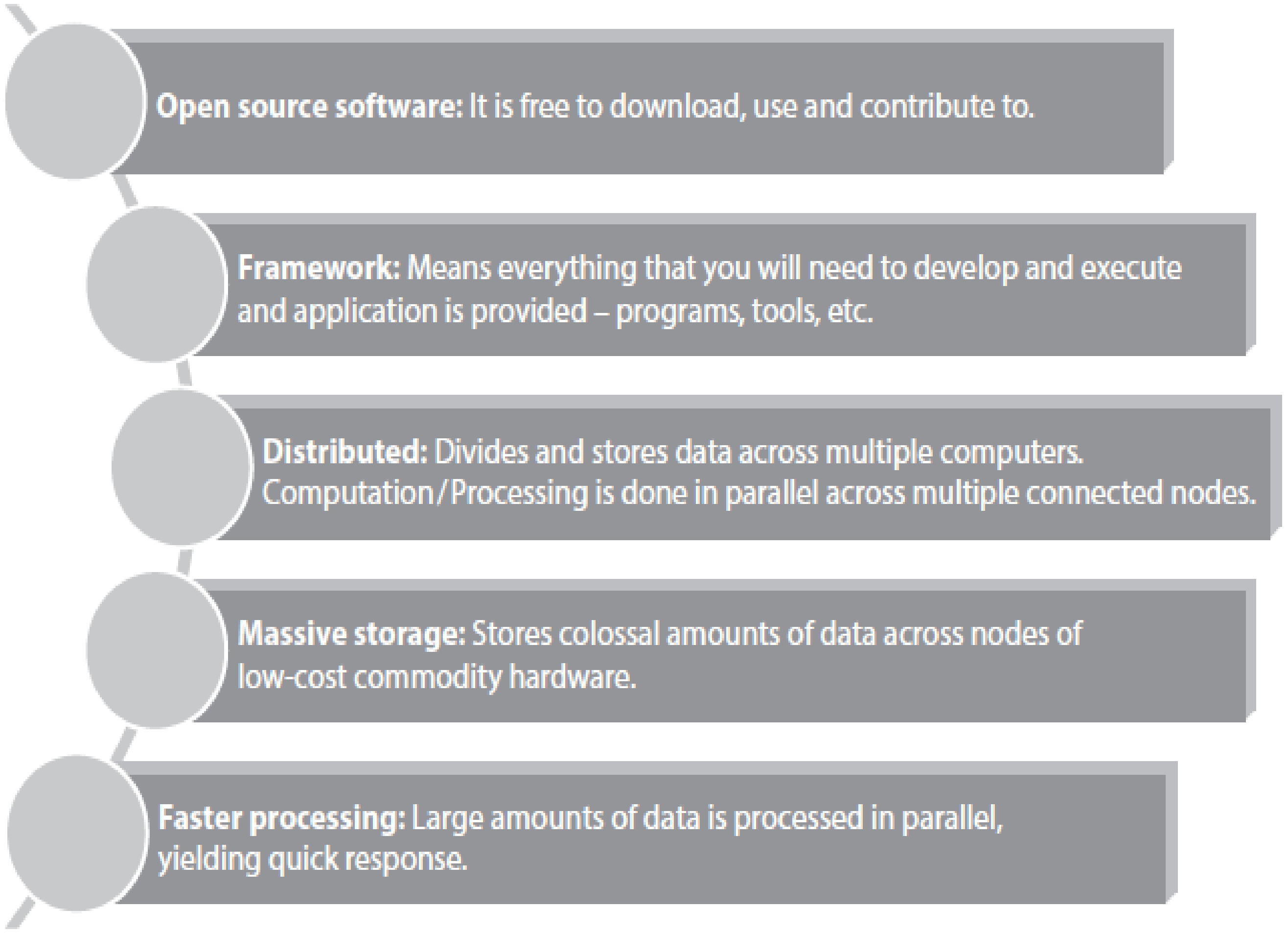
"I need this data to be proceed quickly. My decision is pending". How to access the information quickly?

Hadoop History



Hadoop was created by Doug Cutting, the creator of Apache Lucene (text search library). Hadoop was part of Apace Nutch (open-source web search engine of Yahoo project) and also part of Lucene project. The name Hadoop is not an acronym; it's a made-up name.

Key Aspects of Hadoop



Open source software: It is free to download, use and contribute to.

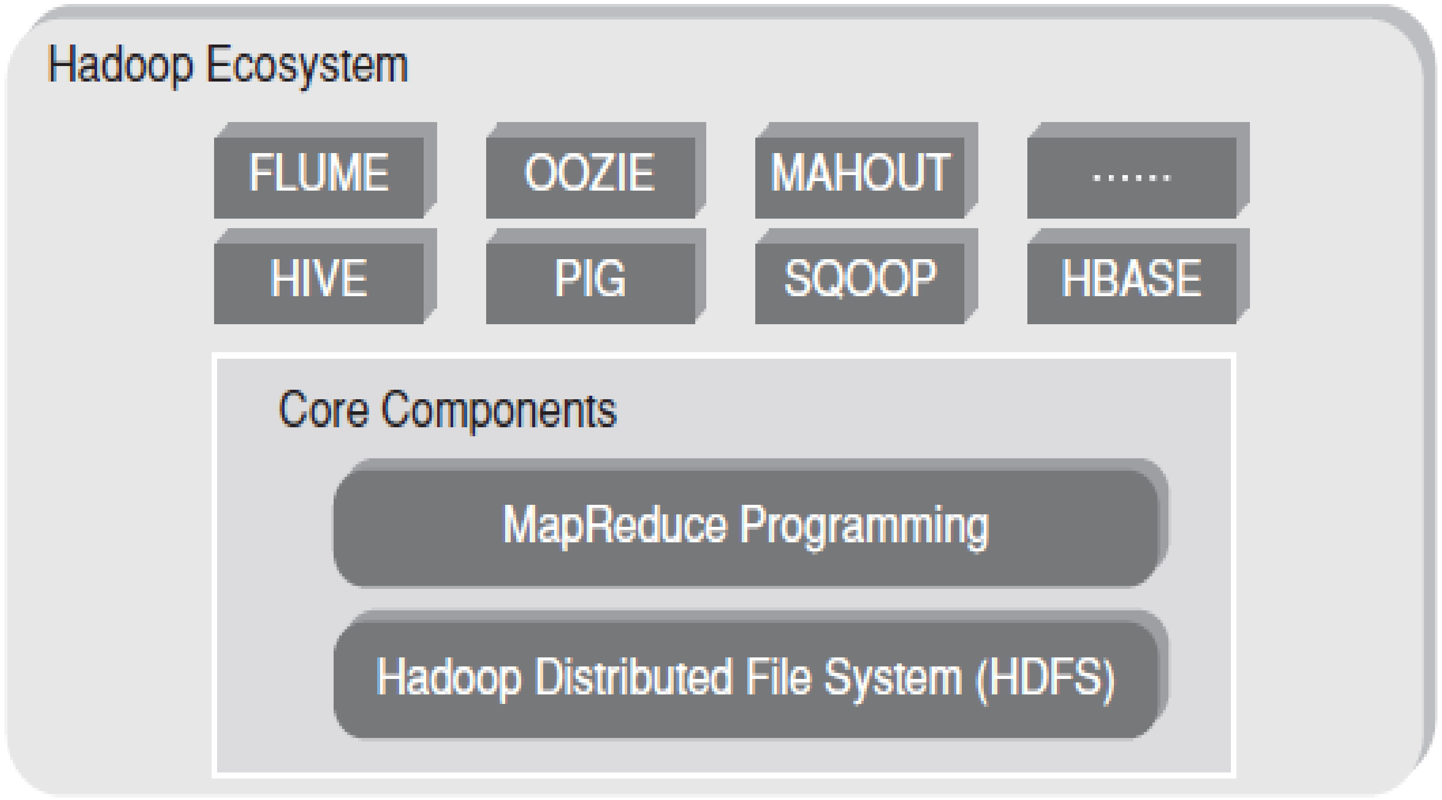
Framework: Means everything that you will need to develop and execute and application is provided – programs, tools, etc.

Distributed: Divides and stores data across multiple computers. Computation/Processing is done in parallel across multiple connected nodes.

Massive storage: Stores colossal amounts of data across nodes of low-cost commodity hardware.

Faster processing: Large amounts of data is processed in parallel, yielding quick response.

Hadoop Components



Hadoop Components cont’d

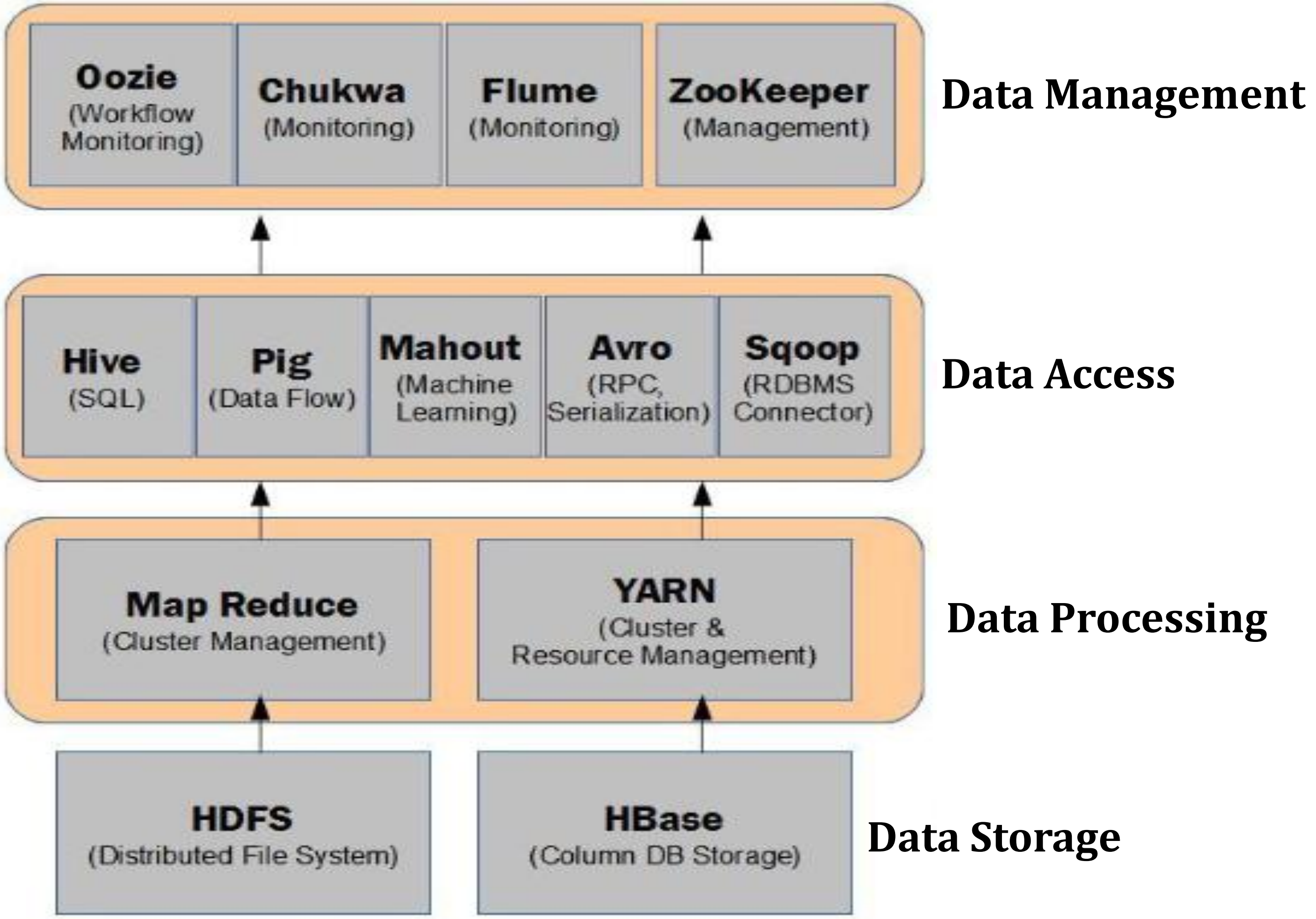
Hadoop Core Components:

- HDFS
  - Storage component
  - Distributed data across several nodes
  - Natively redundant
- MapReduce
  - Computational Framework
  - Splits a task across multiple nodes
  - Process data in parallel

Hadoop Ecosystems: These are support projects to enhance the functionality of Hadoop Core components. The projects are as follows:

- Hive
- Pig
- Sqoop
- Flume
- Oozie
- Mahout
- HBase

Hadoop Ecosystem



Daemons in Hadoop Core

- NameNode
- DataNode
- Secondary NameNode*

HDFS

- JobTracker*
- TaskTracker*

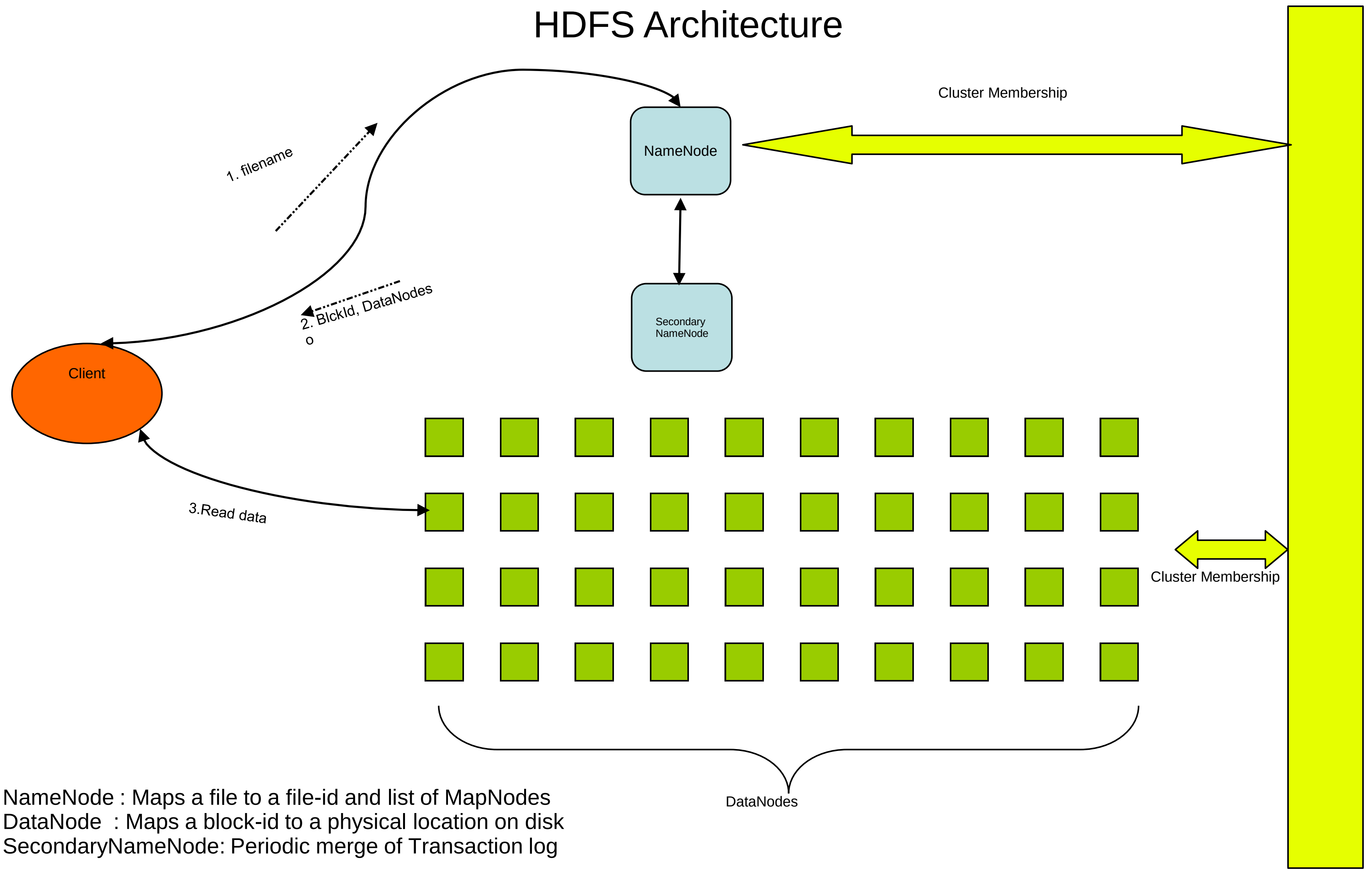
MR

- Daemon Process
  - process which runs in background and has no controlling terminal.

How are Files Stored

- Files are split into blocks
- Blocks are split across many machines at load time
  - Different blocks from the same file will be stored on different machines
- Blocks are replicated across multiple machines
- The NameNode keeps track of which blocks make up a file and where they are stored

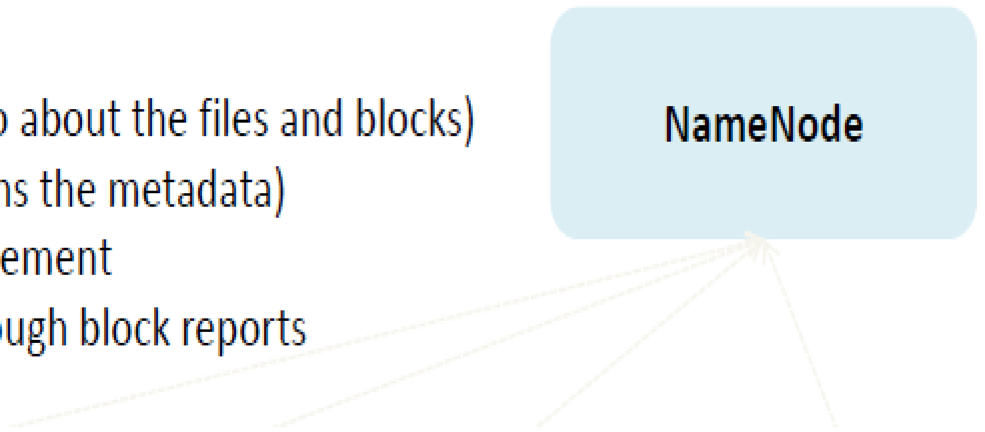
HDFS Architecture



NameNode : Maps a file to a file-id and list of MapNodes
DataNode : Maps a block-id to a physical location on disk
SecondaryNameNode: Periodic merge of Transaction log

Role of NameNode

- Stores the metadata (info about the files and blocks)
- File Management(contains the metadata)
- Block and Replica Management
- Health of datanodes through block reports



Secondary NameNode*

- A helper node for NameNode
- performs memory-intensive administrative functions for the
- NameNode
- Have a checkpoint for the file system (HDFS)
- Not a Backup Node

- Running on a single machine, the **NameNode daemon determines and tracks where the various blocks of a data file are** stored.

- The **DataNode daemon manages the data stored on each machine.**

- **If a client** application wants to access a particular file stored in HDFS, the **application contacts the NameNode.**

- **NameNode provides the application with the locations of the various blocks for that file.**

- The application then communicates with the appropriate DataNodes to access the file.

- Each DataNode periodically builds a report about the blocks stored on the DataNode and sends the report to the NameNode.

- For performance reasons, the NameNode resides in a machine's memory.

- Because the NameNode is critical to the operation of HDFS, any unavailability or corruption of the NameNode results in a data unavailability event on the cluster.

- Thus, the NameNode is viewed as a single point of failure in the Hadoop environment.

- To minimize the chance of a NameNode failure and to improve performance, the NameNode is typically run on a dedicated machine.

- A third daemon, the **Secondary NameNode.**

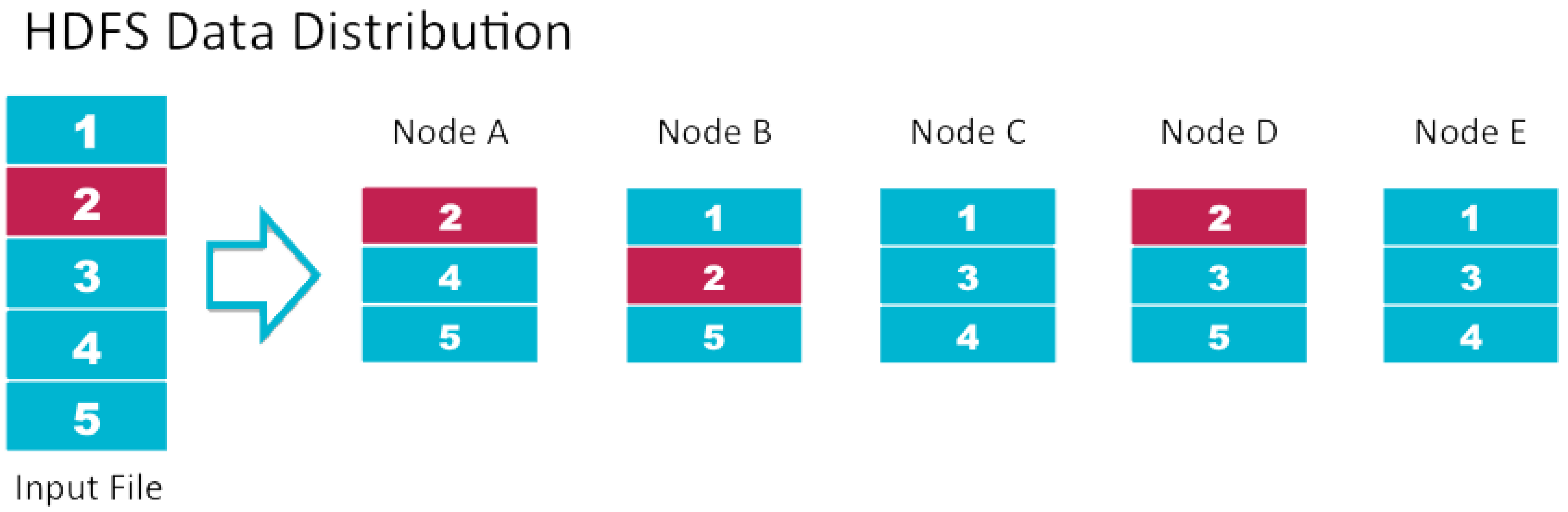
- **It provides the capability to perform some of** the NameNode tasks to reduce the load on the NameNode.

- Such tasks include updating the file system image with the contents of the file system edit logs.

- In the event of a NameNode outage, the NameNode must be restarted and initialized with the last file system image file and the contents of the edits logs.

Data Replication

- Default replication is 3-fold



Block Concept

Files are splitted into number of chunks(Blocks) of pre-defined size

TestFile1.txt	->	1GB
Block Size	->	64 MB

No of Blocks = 1GB / 64MB = 16 blocks

Blocks are B1,B2,.....B16

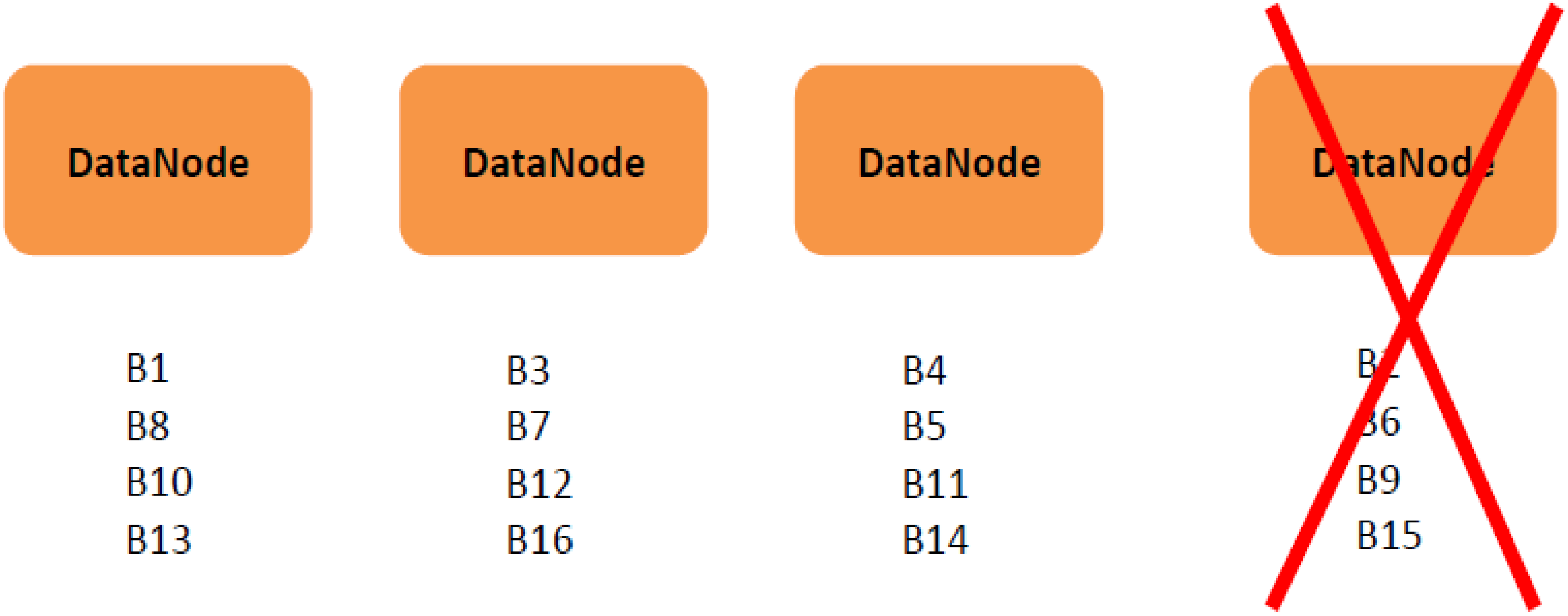
DataNode	DataNode	DataNode	DataNode
B1	B3	B4	B2
B8	B7	B5	B6
B10	B12	B11	B9
B13	B16	B14	B15

Block Concept

What happens to my data if node 4 goes down??

TestFile1.txt -> 1GB
Block Size -> 64 MB

No of Blocks = 1GB / 64MB = 16 blocks
Blocks are B1,B2,.....B16



Data Retrieval

- When a client wants to retrieve data
  - Communicates with the NameNode to determine which blocks make up a file and on which data nodes those blocks are stored
  - Then communicated directly with the data nodes to read the data

Hadoop HDFS

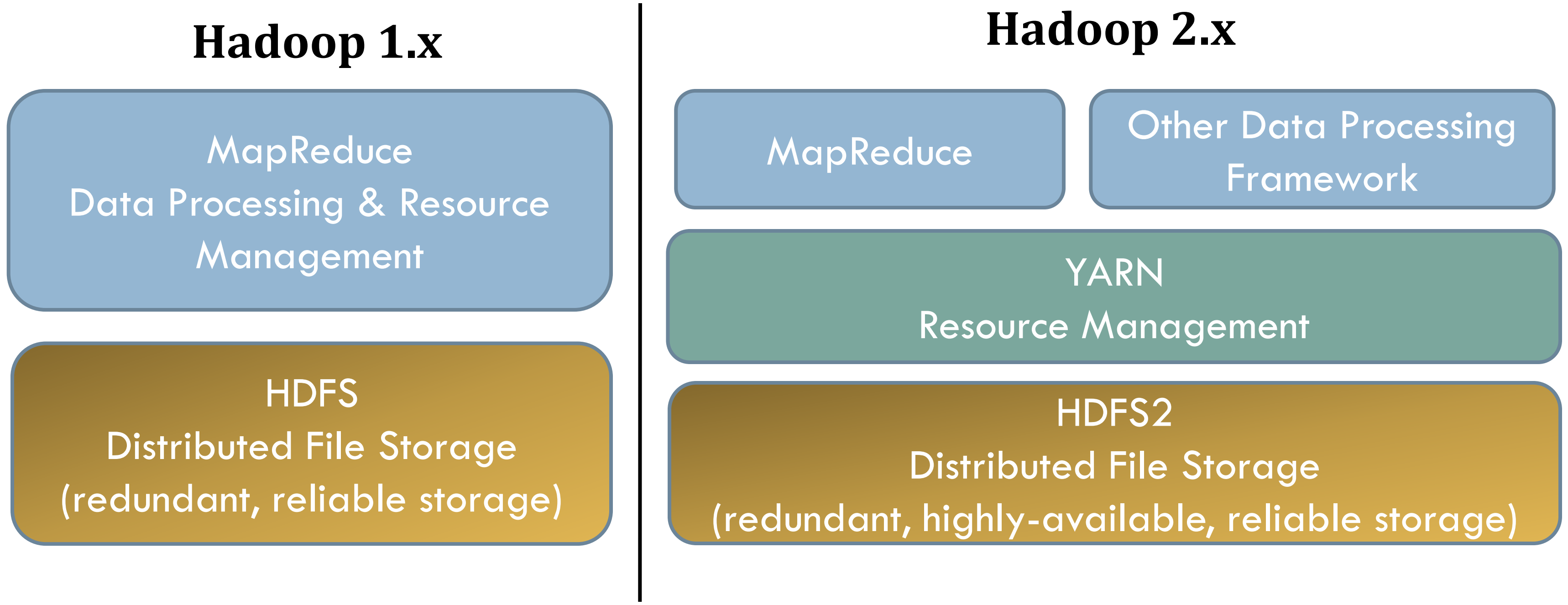
- The Hadoop Distributed File System (HDFS) is the primary data storage system used by Hadoop applications.
- HDFS holds very large amount of data and employs a NameNode and DataNode architecture to implement a distributed file system that provides high-performance access to data across highly scalable Hadoop clusters.
- To store such huge data, the files are stored across multiple machines.
- These files are stored in redundant fashion to rescue the system from possible data losses in case of failure.
- It's run on commodity hardware.
- Unlike other distributed systems, HDFS is highly fault-tolerant and designed using low-cost hardware.

Version of Hadoop

There are 3 versions of Hadoop available:

- ❑ Hadoop 1.x
- ❑ Hadoop 2.x
- ❑ Hadoop 3.x

Hadoop 1.x vs. Hadoop 2.x

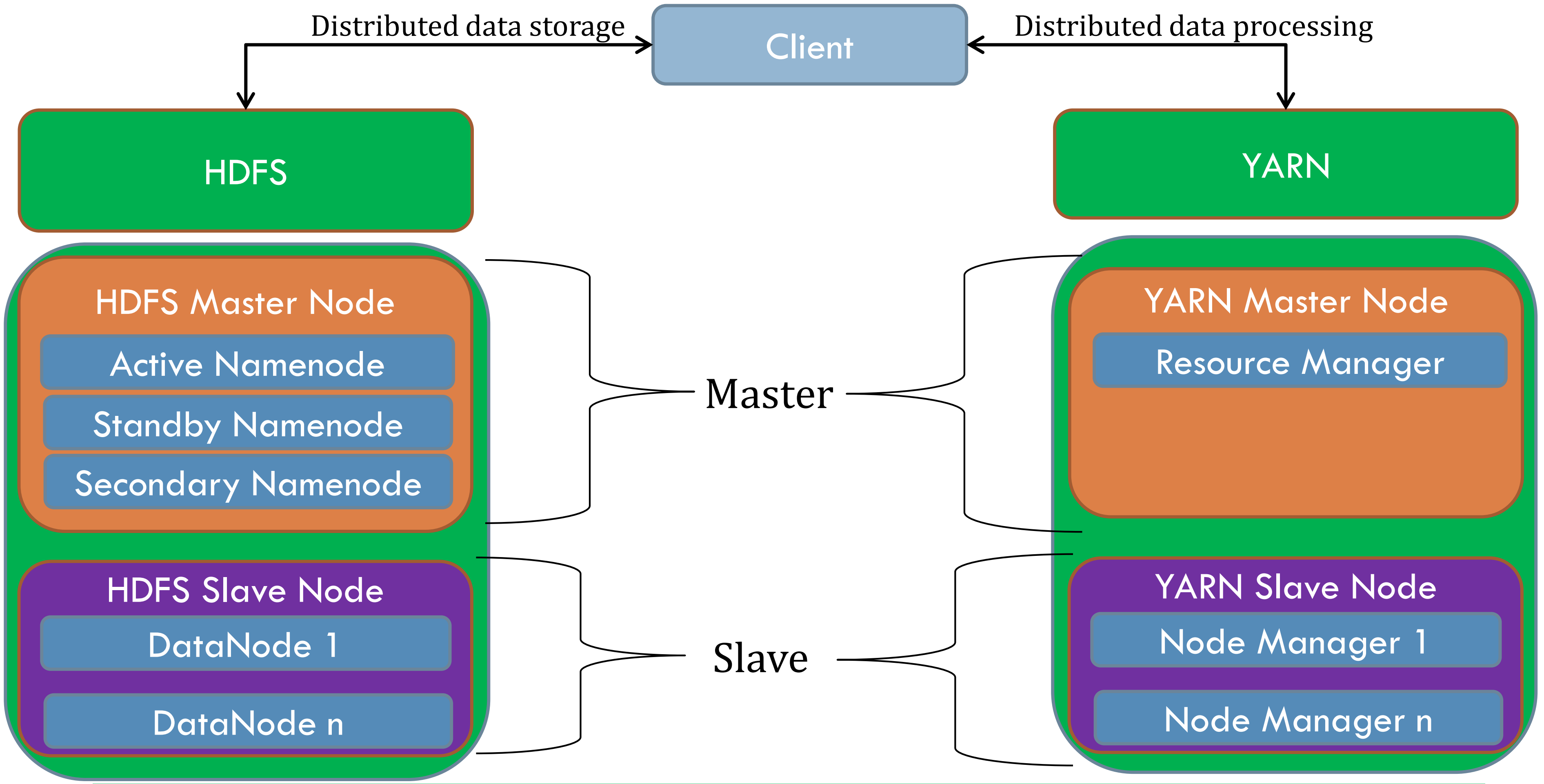

Hadoop 1.x
MapReduce
Data Processing & Resource Management
HDFS
Distributed File Storage
(redundant, reliable storage)
Hadoop 2.x
MapReduce
Other Data Processing Framework
YARN
Resource Management
HDFS2
Distributed File Storage
(redundant, highly-available, reliable storage)

Hadoop 2.x vs. Hadoop 3.x

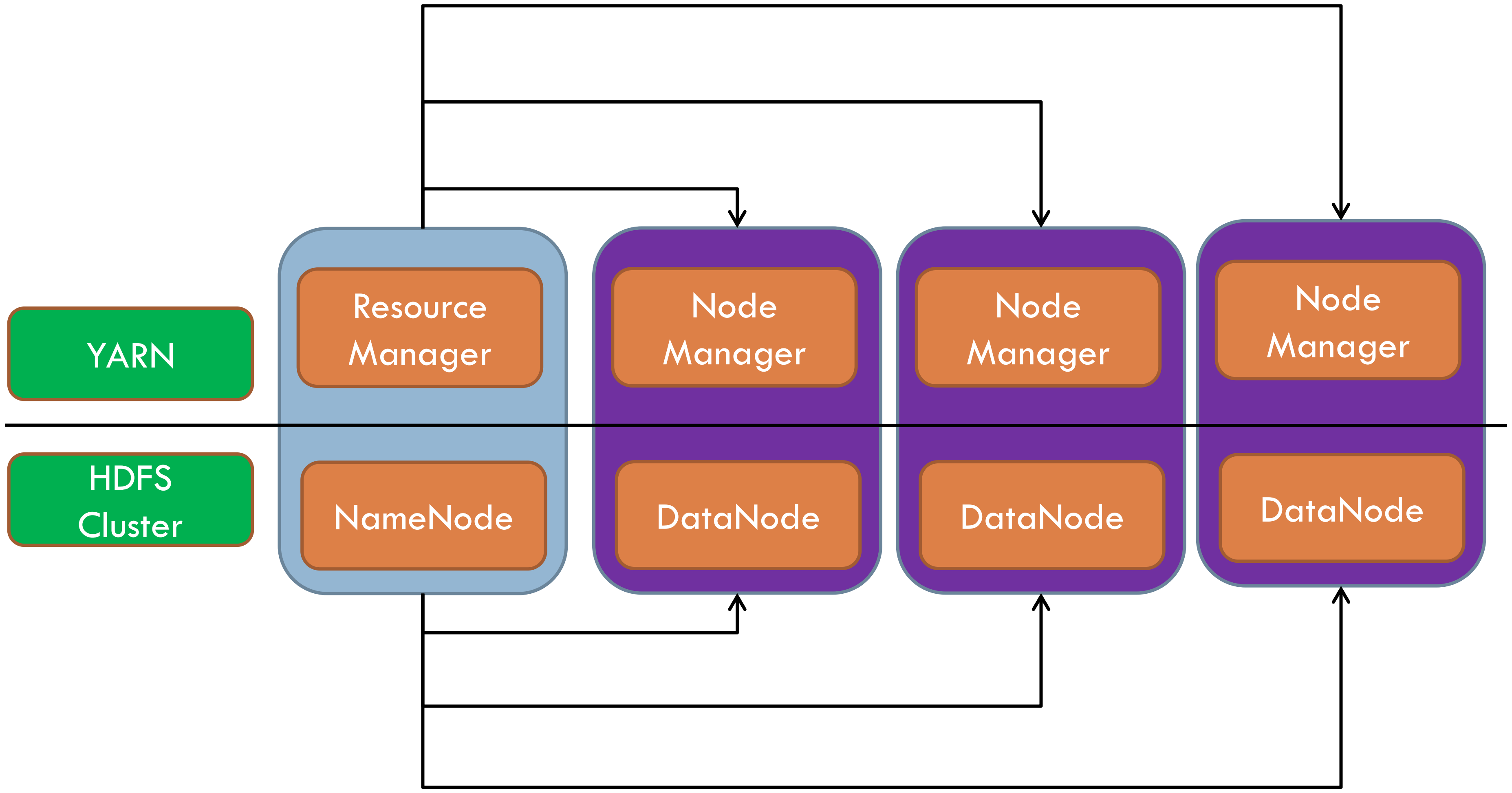
Characteristics	Hadoop 2.x	Hadoop 3.x
Minimum supported version of java	Java 7	Java 8
Fault tolerance	Handled by replication (which is wastage of space).	Handled by erasure coding
Data Balancing	Uses HDFS balancer	Uses Intra-data node balancer, which is invoked via the HDFS disk balancer CLI.
Storage Scheme	Uses 3X replication scheme. E.g. If there is 6 block so there will be 18 blocks occupied the space because of the replication scheme.	Support for erasure encoding in HDFS. E.g. If there is 6 block so there will be 9 blocks occupied the space 6 block and 3 for parity.
Scalability	Scale up to 10,000 nodes per cluster.	Scale more than 10,000 nodes per cluster.

High Level Hadoop 2.0 Architecture

Hadoop is distributed Master-Slave architecture.



High Level Hadoop 2.0 Architecture cont'd

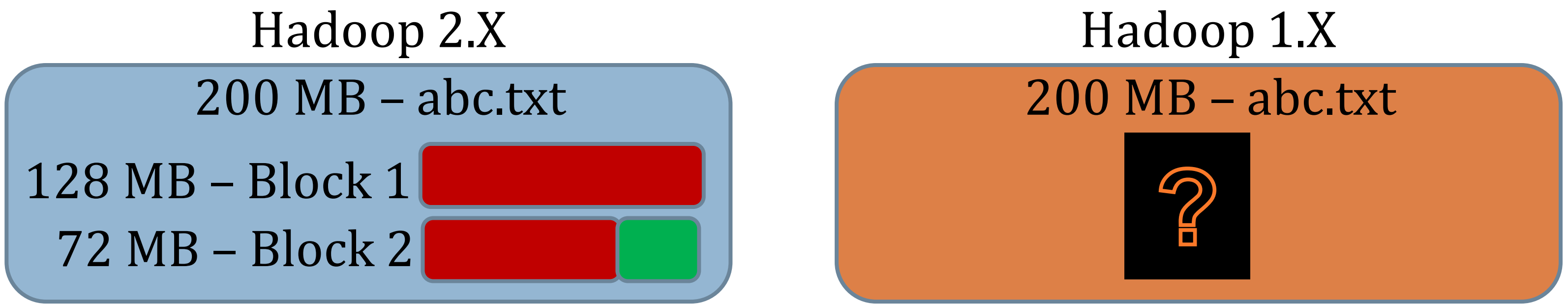


HDFS Physical Architecture

Key components of HDFS are as follows:

1. NameNode
2. DataNodes
3. Secondary NameNode
4. Standby NameNode

Blocks: Generally the user data is stored in the files of HDFS. HDFS breaks a large file into smaller pieces called **blocks**. **In other words, the minimum amount of data that HDFS can read or write is called a block.** By default the block size is 128 MB in Hadoop 2.x and 64 MB in Hadoop 1.x. But it can be increased as per the need to change in HDFS configuration.

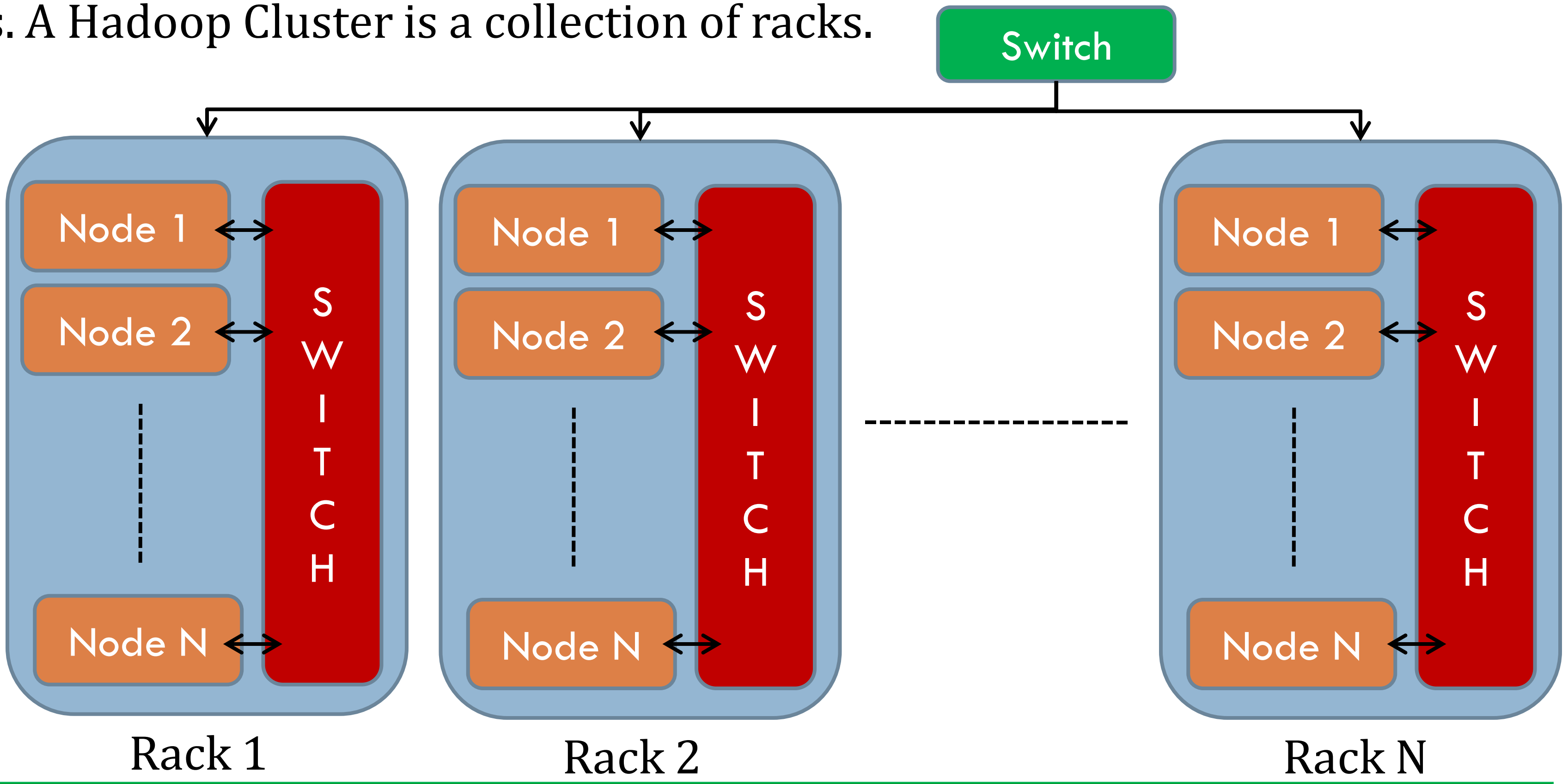


Why block size is large?

1. Reduce the cost of seek time and
2. Proper usage of storage space

Rack

A rack is a collection of 30 or 40 nodes that are physically stored close together and are all connected to the same network switch. Network bandwidth between any two nodes in rack is greater than bandwidth between two nodes on different racks. A Hadoop Cluster is a collection of racks.



Rack Awareness

All machines in rack are connected using the same network switch and if that network goes down then all machines in that rack will be out of service. Thus the rack is down. **Rack Awareness was introduced by Apache Hadoop to overcome this issue.**

In Rack Awareness, **NameNode chooses the DataNode which is closer to the same rack or nearby rack.** NameNode maintains Rack ids of each DataNode to achieve rack information. **Thus, this concept chooses DataNodes based on the rack information.**

NameNode in Hadoop makes ensures that all the replicas should not stored on the same rack or single rack. Default replication factor is 3..

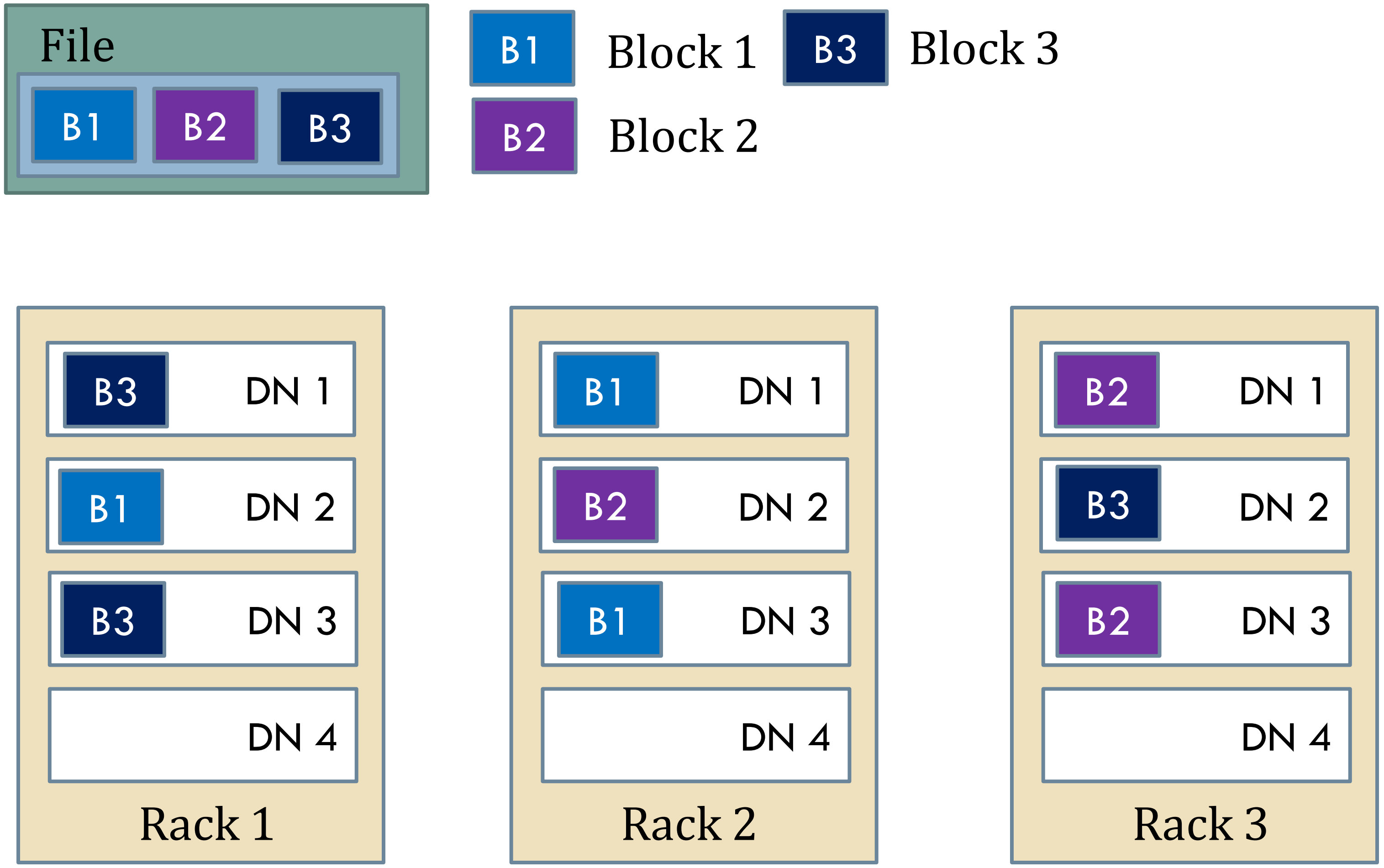
Rack Awareness

Therefore according to Rack Awareness Algorithm:

- When a Hadoop framework creates new block, it places first replica on the local node, and place a second one in a different rack, and the third one is on different node on same remote node.
- When re-replicating a block, if the number of existing replicas is one, place the second on a different rack.
- When number of existing replicas are two, if the two replicas are in the same rack, place the third one on a different rack.
- Replication of data blocks in multiple racks in HDFS via rack awareness is done using a policy called **Replica Replacement Policy.**

The policy states that **"No more than one replica is placed on one node. And no more than 2 replicas are placed on the same rack."**

Rack Awareness & Replication



NameNode Metadata

1. Metadata stored about the file consists of **file name, file path, number of blocks, block Ids, replication level.**
2. **This metadata information is stored on the local disk. Namenode uses two files for storing this metadata information.**
   - ❑ FsImage ❑ EditLog
3. NameNode in HDFS also keeps in it's **memory, location** of the DataNodes that store the blocks for any given file. Using that information Namenode can reconstruct the whole file by getting the location of all the blocks of a given file.

Example

(File Name, numReplicas, rack-ids, machine-ids, block-ids, …)
/user/in4072/data/part-0, 3, r:3, M3, {1, 3}, …
/user/in4072/data/part-1, 3, r:2, M1, {2, 4, 5}, …
/user/in4072/data/part-2, 3, r:1, M2, {6, 9, 8}, …

Standby NameNode

With Hadoop 2.0, built into the platform, HDFS now has **automated failover** with a **hot standby**, with full stack resiliency.

1. **Automated Failover:** Hadoop pro-actively detects NameNode host and process failures and will automatically switch to the standby NameNode to maintain availability for the HDFS service. There is no need for human intervention in the process – System Administrators can sleep in peace!

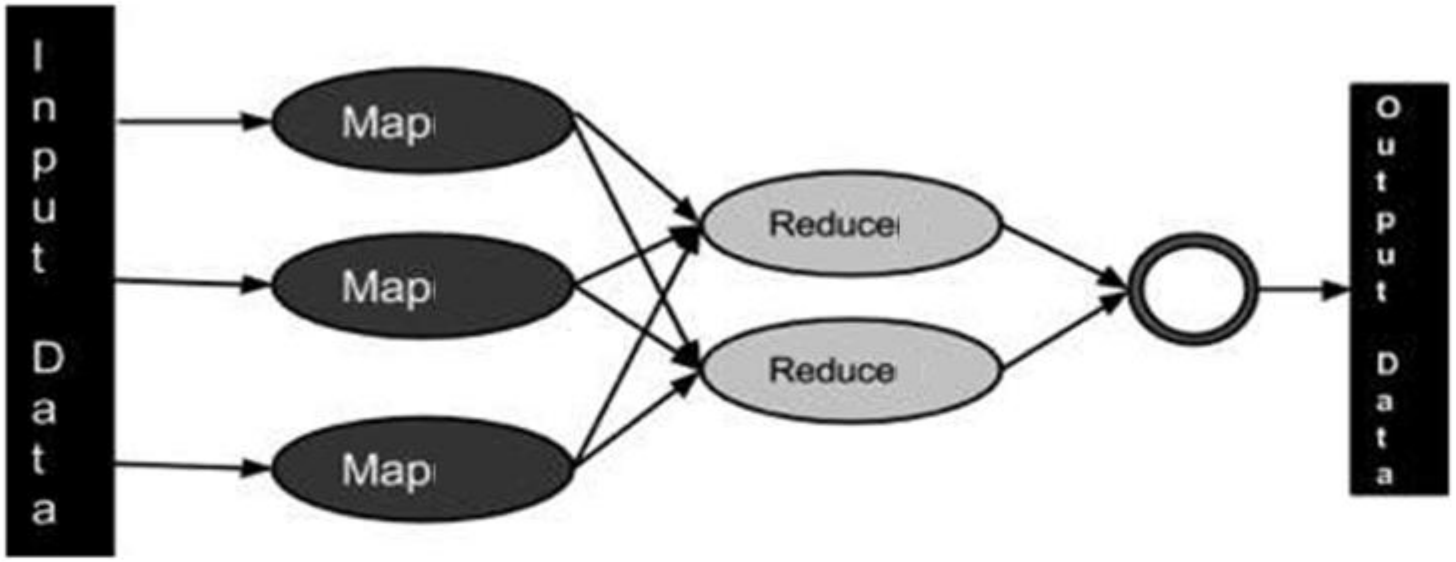
2. **Hot Standby**: Both Active and Standby NameNodes have **up to date HDFS metadata**, ensuring seamless failover even for large clusters – **which means no downtime for your HDP cluster**!

3. **Full Stack Resiliency:** The **entire Hadoop stack** (MapReduce, Hive, Pig, HBase, Oozie etc.) has been certified to handle a NameNode failure scenario without losing data or the job progress. This is vital to ensure long running jobs that are critical to complete on schedule will not be adversely affected during a NameNode failure scenario.

MapReduce

1. MapReduce is a processing technique and a program model for **distributed computing** based on java. It is built on divide and conquer algorithm.
2. In MapReduce Programming, the **input dataset in split into independent chunks.**
3. **It contains two important tasks, namely Map and Reduce.**
4. Map takes a set of data and converts it into another set of data, where **individual elements are broken down into tuples** (key/value pairs). The processing primitive is called **mapper**. The processing is done in parallel manner. The output produced by the map tasks serves as intermediate data and is stored on the local disk of that server.
5. **Reduce** task **takes the output from a map** as an input and combines those data **tuples into a smaller set of tuples**. The processing primitive is called **reducer**. The input and output are stored in a file system.
6. Reduce task is always performed after the map job.

MapReduce cont’d



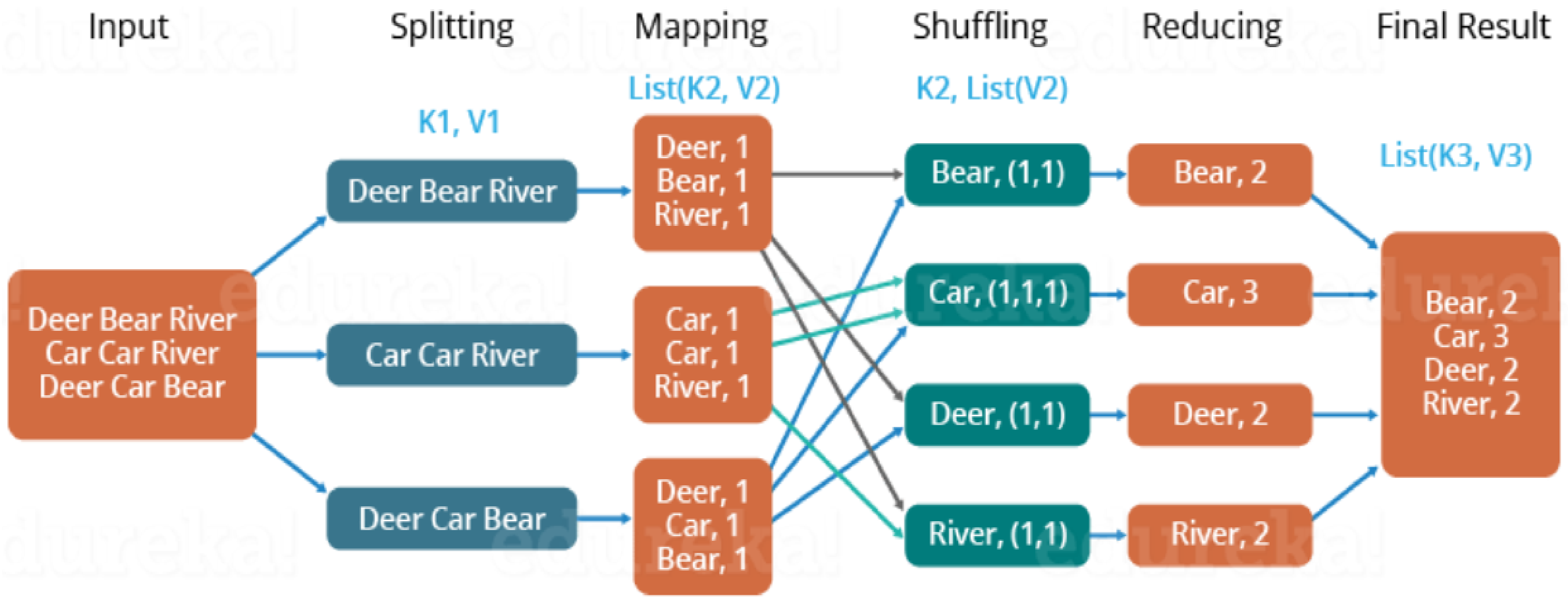
MapReduce cont'd

- The main advantages is that we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster with a configuration change.
- MapReduce program executes in three stages: **map stage, shuffle & sorting stage, and reduce stage.**
- **Map Stage:** The map or mapper's job is to process the input data. Generally the **input data is in the form of file or directory** and is stored in the Hadoop file system (HDFS). The input file is passed to the **mapper function line by line.** The mapper processes the data and creates several small chunks of data.
- **Shuffle & Sorting Stage: Shuffle phase** in Hadoop transfers the map output from Mapper to a Reducer in MapReduce. **Sort phase** in MapReduce covers the merging and sorting of map outputs.
- **Reducer Stage:** The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

How MapReduce Work?

At the crux of MapReduce are two functions: Map and Reduce. They are sequenced one after the other.

- The Map function takes input from the disk as <key,value> pairs, processes them, and produces another set of intermediate <key,value> pairs as output.
- The Reduce function also takes inputs as <key,value> pairs, and produces <key,value> pairs as output.



Working of MapReduce

The types of keys and values differ based on the use case. All inputs and outputs are stored in the HDFS. While the map is a mandatory step to filter and sort the initial data, the reduce function is optional.

<k1, v1> -> Map() -> list(<k2, v2>)
<k2, list(v2)> -> Reduce() -> list(<k3, v3>)

Mappers and Reducers are the Hadoop servers that run the Map and Reduce functions respectively. It doesn't matter if these are the same or different servers.

- **Map**: The input data is first split into smaller blocks. Each block is then assigned to a mapper for processing. For example, if a file has 100 records to be processed, 100 mappers can run together to process one record each. Or maybe 50 mappers can run together to process two records each. The Hadoop framework decides how many mappers to use, based on the size of the data to be processed and the memory block available on each mapper server.

Working of MapReduce cont'd

- **Reduce**: After all the mappers complete processing, the framework shuffles and sorts the results before passing them on to the reducers. A reducer cannot start while a mapper is still in progress. All the map output values that have the same key are assigned to a single reducer, which then aggregates the values for that key.

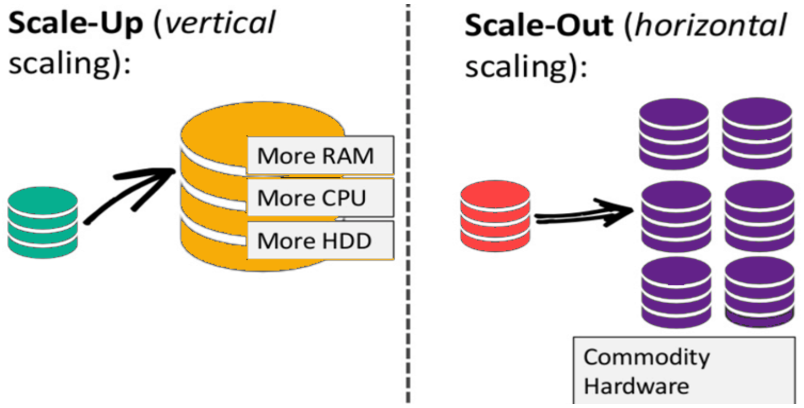
Class Exercise 1

Draw the MapReduce process to count the number of words for the input:

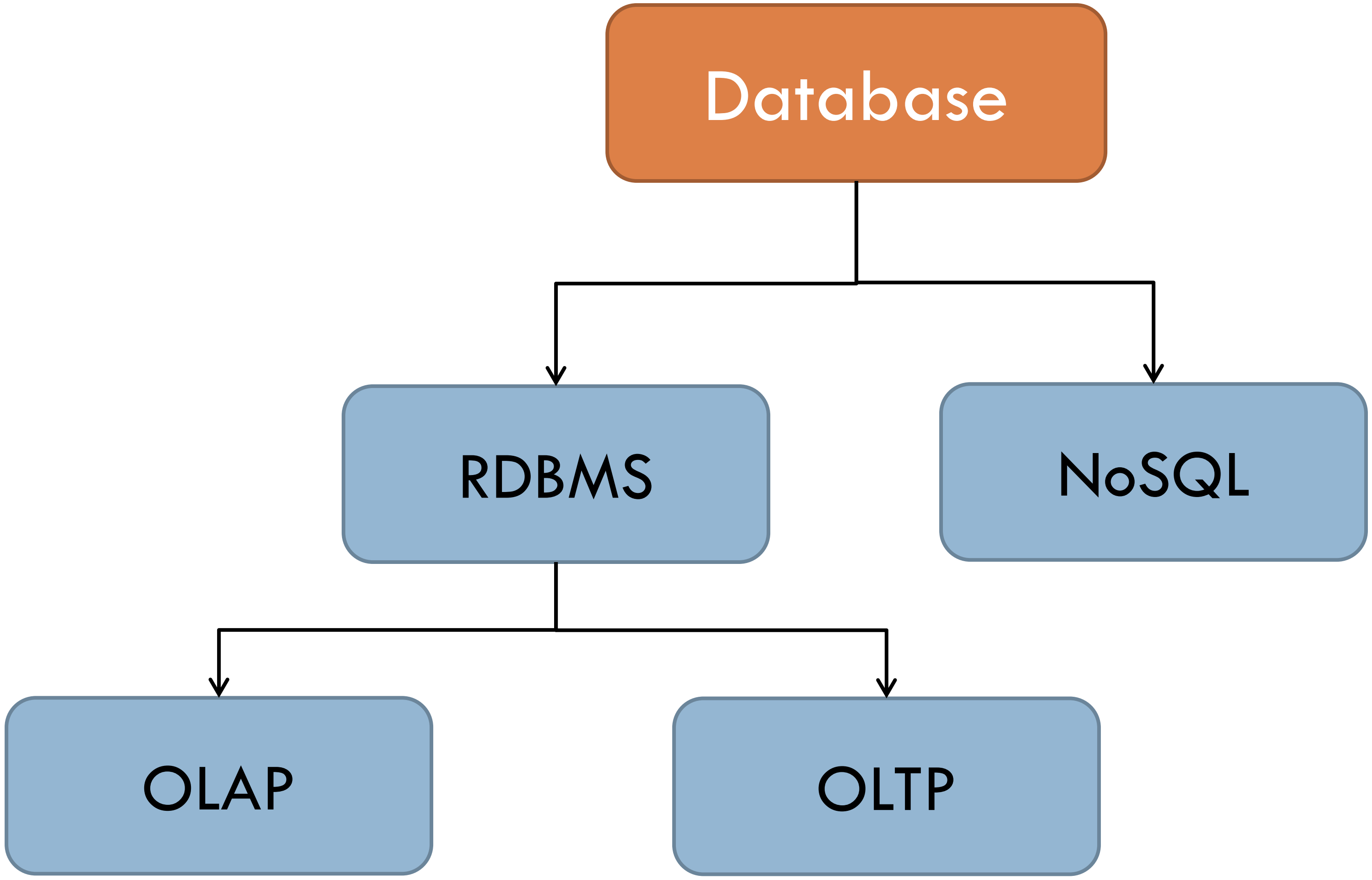
Dog Cat Rat
Car Car Rat
Dog car Rat
Rat Rat Rat

NoSQL

- NoSQL database stands for "Not Only SQL" or "Not SQL."
- It is a non-relational database, that does not require a fixed schema, and avoids joins.
- It is used for distributed data stores and specifically targeted for big data, for example Google or Facebook which collects terabytes of data every day for their users.
- Traditional RDBMS uses SQL syntax to store and retrieve data for further insights. Instead, a NoSQL database system encompasses a wide range of database technologies that can store structured, semi-structured, and unstructured data.
- **It adhere to Brewer's CAP theorem.**
- The tables are stored as ASCII files and each field is separated by tabs
- The data scale horizontally.



NoSQL cont…



RDBMS vs. NoSQL

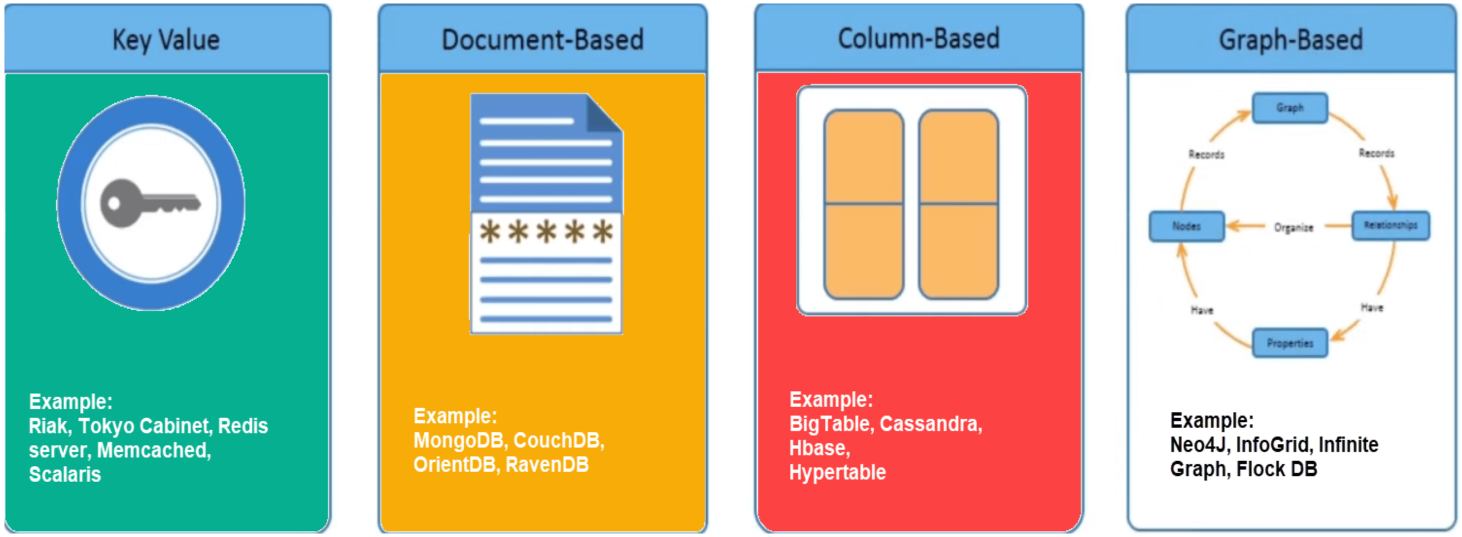
RDBMS	NoSQL
Relational database	Non-relational, distributed database
Relational model	Model-less approach
Pre-defined schema	Dynamic schema for unstructured data
Table based databases	Document-based or graph-based or wide column store or key-value pairs databases
Vertically scalable (by increasing system resources)	Horizontally scalable (by creating a cluster of commodity machines)
Uses SQL	Uses UnQL (Unstructured Query Language)
Not preferred for large datasets	Largely preferred for large datasets
Not a best fit for hierarchical data	Best fit for hierarchical storage as it follows the key-value pair of storing data similar to JSON
Emphasis on ACID properties	Follows Brewer's CAP theorem

RDBMS vs. NoSQL cont'd

RDBMS	NoSQL
Excellent support from vendors	Relies heavily on community support
Supports complex querying and data keeping needs	Does not have good support for complex querying
Can be configured for strong consistency	Few support strong consistency (e.g., MongoDB), few others can be configured for eventual consistency (e.g., Cassandra)
Examples: Oracle, DB2, MySQL, MS SQL, PostgreSQL, etc.	Examples: MongoDB, HBase, Cassandra, Redis, Neo4j, CouchDB, Couchbase, Riak, etc.

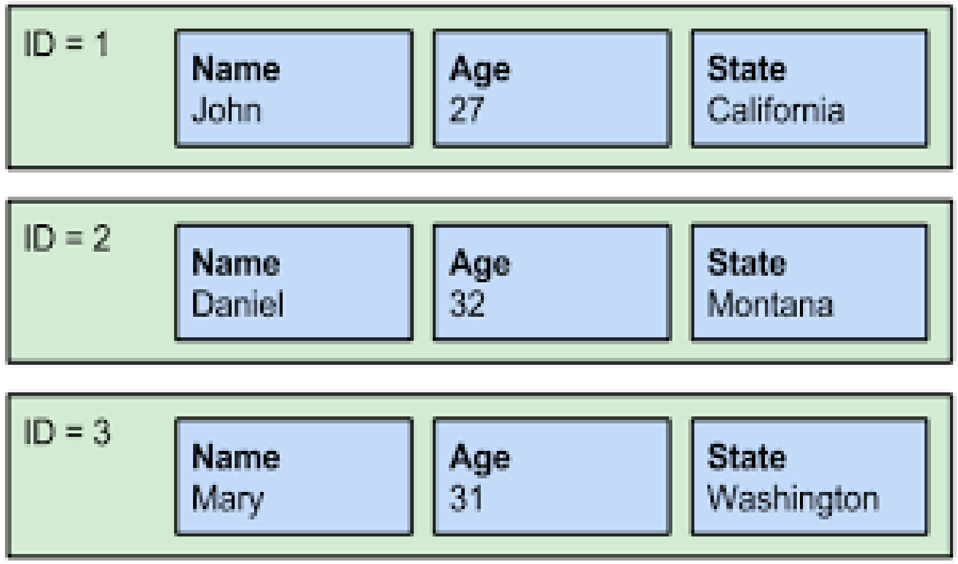
NoSQL Data Architectural Patterns

There are mainly four categories of NoSQL data stores. Each of these categories has its unique attributes and limitations.


Key Value
Example:
Riak, Tokyo Cabinet, Redis server, Memcached, Scalaris
Document-Based
*****
Example:
MongoDB, CouchDB, OrientDB, RavenDB
Column-Based
Example:
BigTable, Cassandra, Hbase, Hypertable
Graph-Based
Graph
Records
Records
Nodes
Organize
Relationships
Have
Have
Properties
Example:
Neo4J, InfoGrid, Infinite Graph, Flock DB

Key Value

Data is stored in key/value pairs. It is designed in such a way to handle lots of data and heavy load. Key-value pair storage databases store data as a hash table where each key is unique, and the value can be a JSON, BLOB, string, etc. It is one of the most basic types of NoSQL databases. This kind of NoSQL database is used as a collection, dictionaries, associative arrays, etc. Key value stores help the developer to store schema-less data. They work best for shopping cart contents. Redis, Dynamo, Riak are some examples of key-value store.



SQL

ID	Name	Age	State
1	John	27	California

NoSQL – Key Value

Key (i.e. ID)	Values
1	Name: John Age:27 State: California

Document-Based

Document-Oriented NoSQL DB stores and retrieves data as a key value pair but the value part is stored as a document. The document is stored in JSON or XML formats. The document type is mostly used for CMS (Content Management Systems), blogging platforms, real-time analytics & e-commerce applications. It should not use for complex transactions which require multiple operations or queries against varying aggregate structures.

SQL

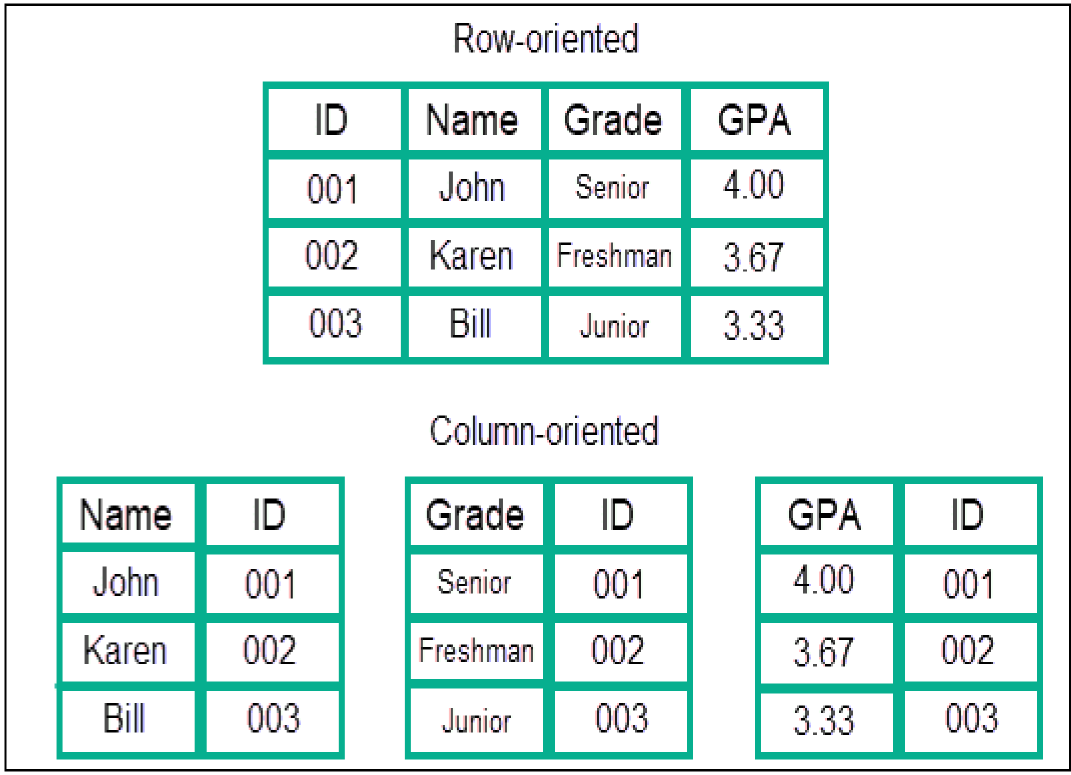
ID	Name	Age	State
1	John	27	California

NoSQL – Document-Based

Key (ID)	Value (JSON)
1	{ “Name”: John “Age”:27 “State”: California }

Column-Based Database

Column-oriented databases work on column family and based on BigTable paper by Google. Every column is treated separately. Values of single column databases are stored contiguously. They deliver high performance on aggregation queries like SUM, COUNT, AVG, MIN etc. as the data is readily available in a column. Such NoSQL databases are widely used to manage data warehouses, business intelligence, CRM, Library card catalogs etc.



Row-oriented

ID	Name	Grade	GPA
001	John	Senior	4.00
002	Karen	Freshman	3.67
003	Bill	Junior	3.33

Column-oriented

Name	ID
John	001
Karen	002
Bill	003

Grade	ID
Senior	001
Freshman	002
Junior	003

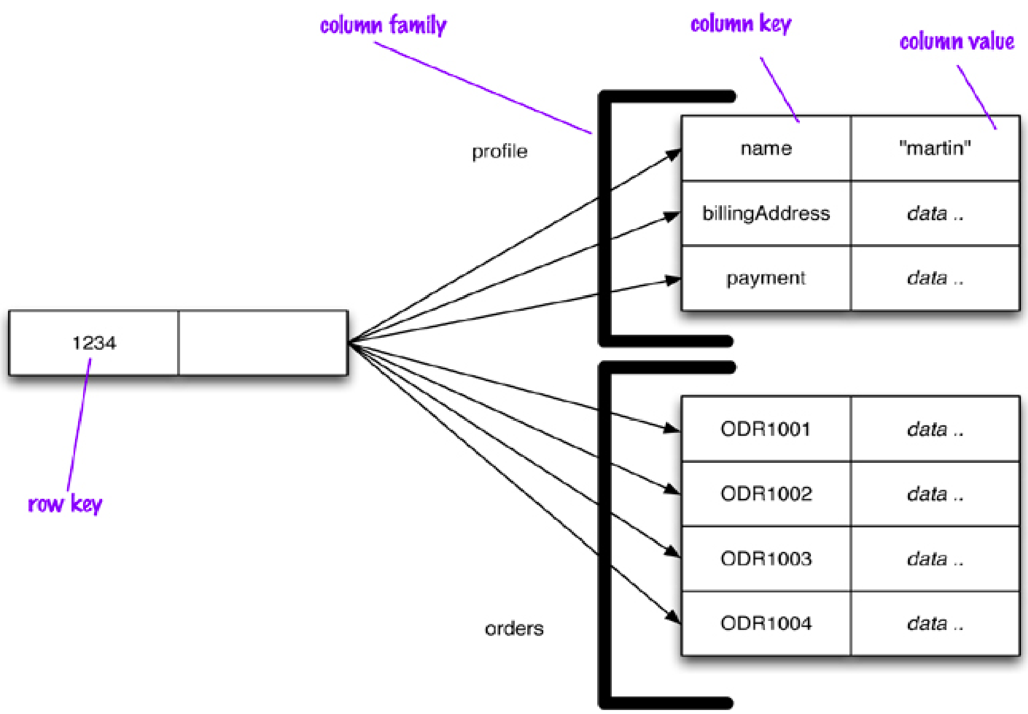
GPA	ID
4.00	001
3.67	002
3.33	003

Column-Oriented vs. Row-Oriented Database cont'd

Row Oriented	Column Oriented
Data is stored and retrieved **one row at a time and hence could read** unnecessary data if some of the data in a row are required.	In this type of data stores, **data are stored and retrieve in columns and hence it can only able to read only the relevant data** if required.
Records in Row Oriented Data stores are easy to read and write.	In this type of data stores, read and write operations are slower as compared to row-oriented.
Row-oriented data stores are best suited for online transaction system.	Column-oriented stores are best suited for online analytical processing.
These are not efficient in performing operations applicable to the entire datasets and hence aggregation in row-oriented is an expensive job or operations.	These are efficient in performing operations applicable to the entire dataset and hence enables aggregation over many rows and columns.

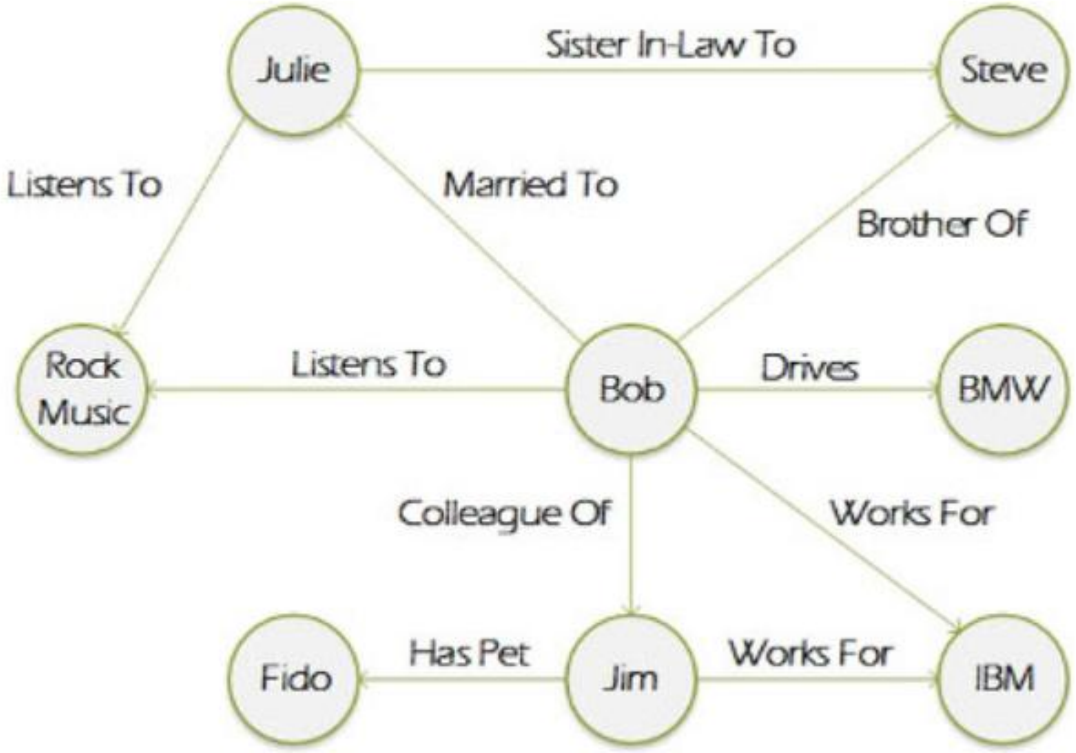
Column-Based Database

Column-oriented databases work on column family and based on BigTable paper by Google. Every column is treated separately. Values of single column databases are stored contiguously. They deliver high performance on aggregation queries like SUM, COUNT, AVG, MIN etc. as the data is readily available in a column. Such NoSQL databases are widely used to manage data warehouses, business intelligence, CRM, Library card catalogs etc.



Graph-Based

A graph type database stores entities as well the relations amongst those entities. The entity is stored as a node with the relationship as edges. An edge gives a relationship between nodes. Every node and edge has a unique identifier. Graph base database mostly used for social networks, logistics, spatial data.



Advantages of NoSQL

- Can be used as Primary or Analytic Data Source
- Big Data Capability
- No Single Point of Failure
- Easy Replication
- No Need for Separate Caching Layer
- Provides fast performance and horizontal scalability.
- Can handle structured, semi-structured, and unstructured data with equal effect
- NoSQL databases don't need a dedicated high-performance server
- Support Key Developer Languages and Platforms
- Simple to implement than using RDBMS
- It can serve as the primary data source for online applications.
- Handles big data which manages data velocity, variety, volume, and complexity
- Excels at distributed database and multi-data center operations
- Eliminates the need for a specific caching layer to store data
- Offers a flexible schema design which can easily be altered without downtime or service disruption

Disadvantages of NoSQL

- No standardization rules
- Limited query capabilities
- RDBMS databases and tools are comparatively mature
- It does not offer any traditional database capabilities, like consistency when multiple transactions are performed simultaneously.
- When the volume of data increases it is difficult to maintain unique values as keys become difficult
- Doesn't work as well with relational data
- The learning curve is stiff for new developers
- Open source options so not so popular for enterprises.

THANK YOU!