






INTRODUCTION TO NOSQL

- 
- 
- Unlike relational databases, the NoSQL databases do not have a strict schema.
 - The records can be in the form of key-value pairs or documents.
 - The NOSQL database was invented by Carlo Strozzi in 1998.
 - It provides a mechanism for retrieval and storage of data.

RDBMS	NoSQL
<ul style="list-style-type: none"> • Data is stored in a relational model, with rows and columns. • A row contains information about an item while columns contain specific information, such as 'Model', 'Date of Manufacture', 'Color'. • Follows fixed schema. Meaning, the columns are defined and locked before data entry. In addition, each row contains data for each column. • Supports vertical scaling. Scaling an RDBMS across multiple servers is a challenging and time-consuming process. • Atomicity, Consistency, Isolation & 	<ul style="list-style-type: none"> • Data is stored in a host of different databases, with different data storage models. • Follows dynamic schemas. Meaning, you can add columns anytime. • Supports horizontal scaling. You can scale across multiple servers. Multiple servers are cheap commodity hardware or cloud instances, which make scaling cost-effective compared to vertical scaling. • Not ACID Compliant.





Characteristics of NOSQL

- NOSQL databases do not use SQL.
 - The creators of NOSQL meant that it is a query language that doesn't follow the principles of RDBMS.
 - They are open-source projects.
 - NOSQL databases offer a range of options for consistency and distribution.
 - We can freely add fields to the database records without having to define any change to the structure because NoSQL operate without a schema.
- 




Advantages of NOSQL

- Can easily scale up and down: NoSQL database supports scaling rapidly and elastically.
 - Cluster scale: It allows distribution of database across 100+ nodes often in multiple data centres
 - Performance scale: It sustains over 100,000+ database reads and writes per second.
 - Data Scale: It supports housing of 1 billion+ documents in the database.
- 

- 
- Doesn't require pre-defined schema.
 - Cheap and easy to implement.
 - Data can be replicated to multiple nodes and can be partitioned
 - Sharding: Sharding is a method of storing data records across many server instances.
 - Replication: Multiple copies of data are stored across clusters and even across data centers. This promises high availability and fault tolerance.




Column Family Databases

- In column family databases the basic unit of data storage is a column, which has a name and a value.
 - A collection of columns make up a row which is identified by a row-key.
 - Columns are grouped together into columns families.
 - Unlike, relational databases, the column family databases do not need to have fixed schemas and a fixed number of columns in each row.
 - The number of columns in a column family database can vary across different rows.
- 



HBase

- HBase is a scalable, non-relational, distributed, column-family database that provides structured data storage for large tables.
 - HBase can store both structured and unstructured data.
 - The data storage in HBase can scale linearly and automatically by the addition of new nodes
- 

Table

RowKey-1	ColumnFamily-1		ColumnFamily-2
	Column-1	Column-2	Column-3
RowKey-2

RowKey-3	Column-2		..

RowKey-4	Column-1	Column-2	Column-4






PERSON TABLE					
row key	personal_data		demographic		...
PersonID	Name	Address	BirthDate	Gender	...
1	H. Houdini	Budapest, Hungary	1926-10-31	M	
2	D. Copper	New Jersey, USA	1956-09-16	M	
3	Merlin	Stonehenge, England	1136-12-03	F	
...	
500,000,000	F. Cadillac	Nevada, USA	1964-01-07	M	

Figure 6-1. Census data as an HBase schema

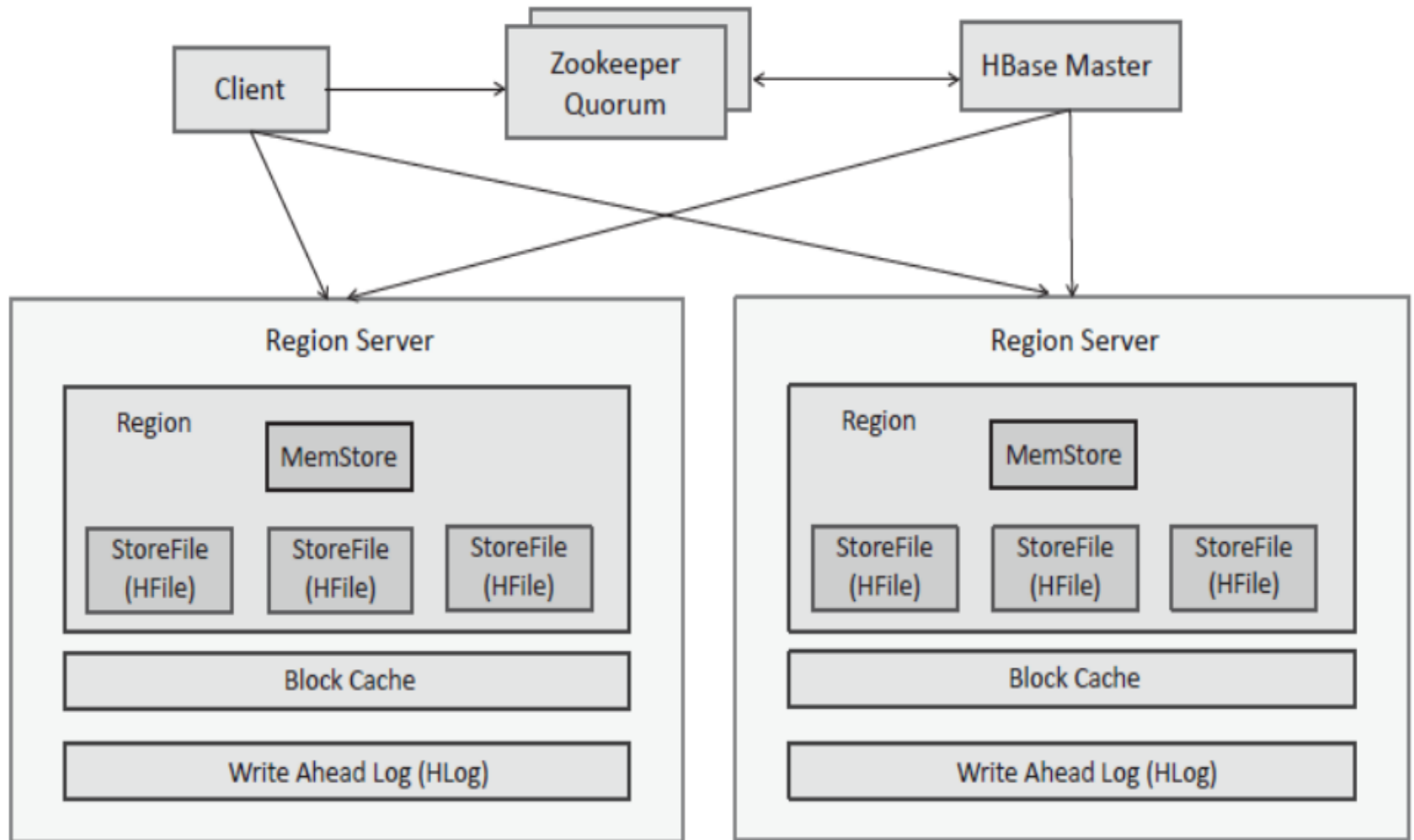


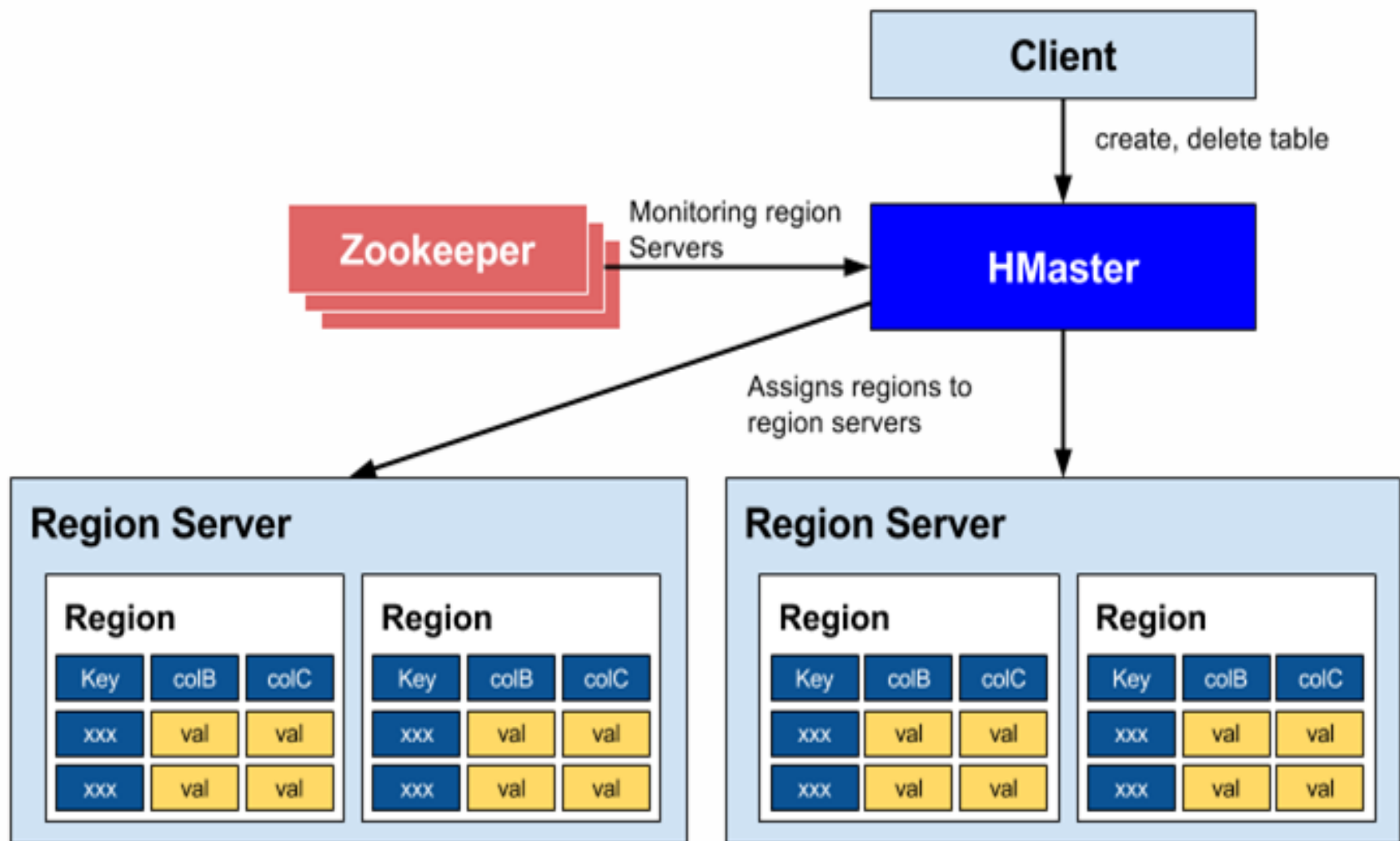
HBase is often described as



1. Sparse: HBase, in contrast, has sparse tables as each row doesn't need to have all the columns. Only the columns which are populated in a row are stored.
 2. Distributed: HBase is a distributed database. HBase tables are partitioned based on row keys into regions. Each region contains a range of row keys. A typical HBase deployment contains multiple Region Servers. Each Region Server contains several regions from different tables.
- 



- 
- 
3. Persistent: HBase works on top of HDFS and all data stored in HBase tables is persisted on HDFS.
 4. Multi-dimensional : HBase stores data as key-value pairs where the keys are multidimensional. A key includes: (Table, RowKey, ColumnFamily, Column, TimeStamp)
 5. Sorted Map: HBase rows are sorted by the row key in lexicographic order. Columns in a column family are sorted by the column key.



HBase Architecture






- 
- 
- HBase tables are partitioned by the row key into multiple regions (H regions)
 - Each region server has multiple regions.
 - HBase has a master-slave architecture with one of the nodes acting as the master node (HMaster) and other nodes are slave nodes.
 - The HMaster is responsible for maintaining the HBase meta-data and assignment of regions to region servers.

- 
- 
- HBase uses Zookeeper for distributed state coordination.
 - HBase has two special tables - ROOT and META, for identifying which region server is responsible for serving a read/write request for a specific row key.

- 
- 
- Each Region Server stores two types of files - a store file (HFile) and a write-ahead log (HLog).
 - HLog is stored on HDFS, it ensures that even in the event of loss of Memstore (which is an in-memory buffer), the writes are never lost.
 - Each Region Server has a Memstore and a Block Cache.
 - The Memstore stores the recent edits to the data in memory and the Block Cache caches the data blocks.




HBase supports the following operations:

- Get: Get operation is used to return values for a given row key.
 - Scan: Scan operation returns values for a range of row keys.
 - Put: Put operation is used to add a new entry.
 - Delete: Delete operation adds a special marker called Tombstone to an entry. Entries marked with Tombstones are removed during the compaction process
- 



Read Path:

- For read operations the client first contacts Zookeeper to get the location of the ROOT table.
 - The client then checks the ROOT table for correct META table containing the row key and obtains the Region Server name that is responsible for serving requests for that row-key.
 - The client then contacts the Region Server directly to complete the read operation.
- 




Write Path:

- All write requests are first logged into the (HLog) sequentially.
- Once data is logged, it is also written to the Memstore. The Memstore stores the most recent updates to enable fast lookups. Over time, the Memstore starts filling up as new updates are stored.
- When the Memstore is filled up, it is flushed to the disk creating a new store file (HFile).





Graph Databases:

- Graph stores are NoSQL databases designed for storing data that has graph structure with nodes and edges.
 - graph databases model data in the form of nodes and relationships.
 - Nodes have a set of attributes. A node can represent different types of entities.
 - The relationships between the entities are represented in the form of links between the nodes.
- 

	Key-Value DB	Document DB	Column Family DB	Graph DB
Data model	Key-value pairs uniquely identified by keys	Documents (having key-value pairs) uniquely identified by document IDs	Columns having names and values, grouped into column families	Graphs comprising of nodes and relationships
Querying	Query items by key, Database specific APIs	Query documents by document-ID, Database specific APIs	Query rows by key, Database specific APIs	Graph query language such as Cypher, Database specific APIs
Use	Applications involving frequent small reads and writes with simple data models	Applications involving data in the form of documents encoded in formats such as JSON or XML, documents can have varying number of attributes	Applications involving large volumes of data, high throughput reads and writes, high availability requirements	Applications involving data on entities and relationships between the entities, spatial data
Examples	DynamoDB, Cassandra	MongoDB, CouchDB	HBase, Google BigTable	Neo4j, AllegroGraph




INTRODUCTION TO HADOOP

- 
- 
- Hadoop is a framework that allows for distributed processing of large data sets across clusters of computers using simple programming models.
 - It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.



Features of Hadoop

- 1. Hadoop Brings Flexibility In Data Processing**
 - Hadoop manages data whether structured or unstructured, encoded or formatted, or any other type of data.
 - Hadoop brings the value to the table where unstructured data can be useful in decision making process.
- 





2. **Hadoop Is Easily Scalable**

- It is an open source platform and runs on industrystandard hardware.
- That makes Hadoop extremely scalable platform where new nodes can be easily added in the system as and data volume of processing needs grow without altering anything in the existing systems or programs.




3. Hadoop Is Fault Tolerant

- In Hadoop, the data is stored in HDFS where data automatically gets replicated at two other locations.
- So, even if one or two of the systems collapse, the file is still available on the third system at least.
- This brings a high level of fault tolerance

- 
- 
4. Hadoop Is Great At Faster Data Processing
- Hadoop can perform batch processes 10 times faster than on a single thread server or on the mainframe.



5. Hadoop Ecosystem Is Robust

- Hadoop Ecosystem comes with a suite of tools and technologies making it very much suitable to deliver to a variety of data processing needs.
 - Just to name a few, Hadoop ecosystem comes with projects such as MapReduce, Hive, HBase, Zookeeper, HCatalog, Apache Pig etc.
- 




6. Hadoop Is Very Cost Effective

- Hadoop generates cost benefits by bringing massively parallel computing to commodity servers, resulting in a substantial reduction in the cost per terabyte of storage, which in turn makes it reasonable to model all your data.






HDFS Architecture

- HDFS follows the master-slave architecture.
 - It has a NameNode and a number of DataNodes.
 - The NameNode is a master that manages the various DataNodes.
- 




NameNode

- The NameNode is the commodity hardware that contains the GNU/Linux operating system and the NameNode software.
 - It is a software that can be run on commodity hardware.
 - The NameNode manages the HDFS cluster metadata.
 - Operations on them, such as modification, opening or closing are performed by NameNode .
 - The system having the NameNode acts as the master server
- 

- 
- 
- it does the following tasks:
 - ☐ Manages the file system namespace.
 - ☐ Regulates client's access to files.
 - ☐ It also executes file system operations such as renaming, closing, and opening files and directories.




DataNode

- The DataNode is a commodity hardware having the GNU/Linux operating system and DataNode software.
 - For every node (Commodity hardware/System) in a cluster, there will be a DataNode.
 - These nodes manage the data storage of their system.
 - DataNodes perform read-write operations on the file systems, as per client request.
 - They also perform operations such as block creation, deletion, and replication according to the instructions of the NameNode.
- 




Block

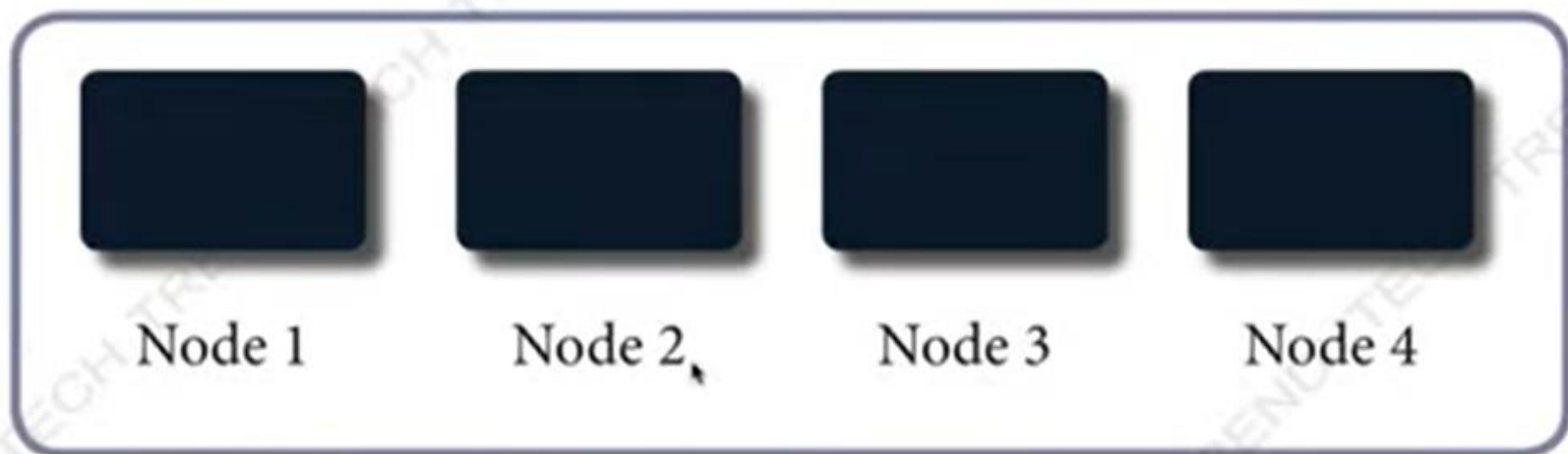
- Generally the user data is stored in the files of HDFS.
 - The file in a file system will be divided into one or more segments and/or stored in individual data nodes.
 - These file segments are called as blocks.
- 



Hadoop Heartbeat

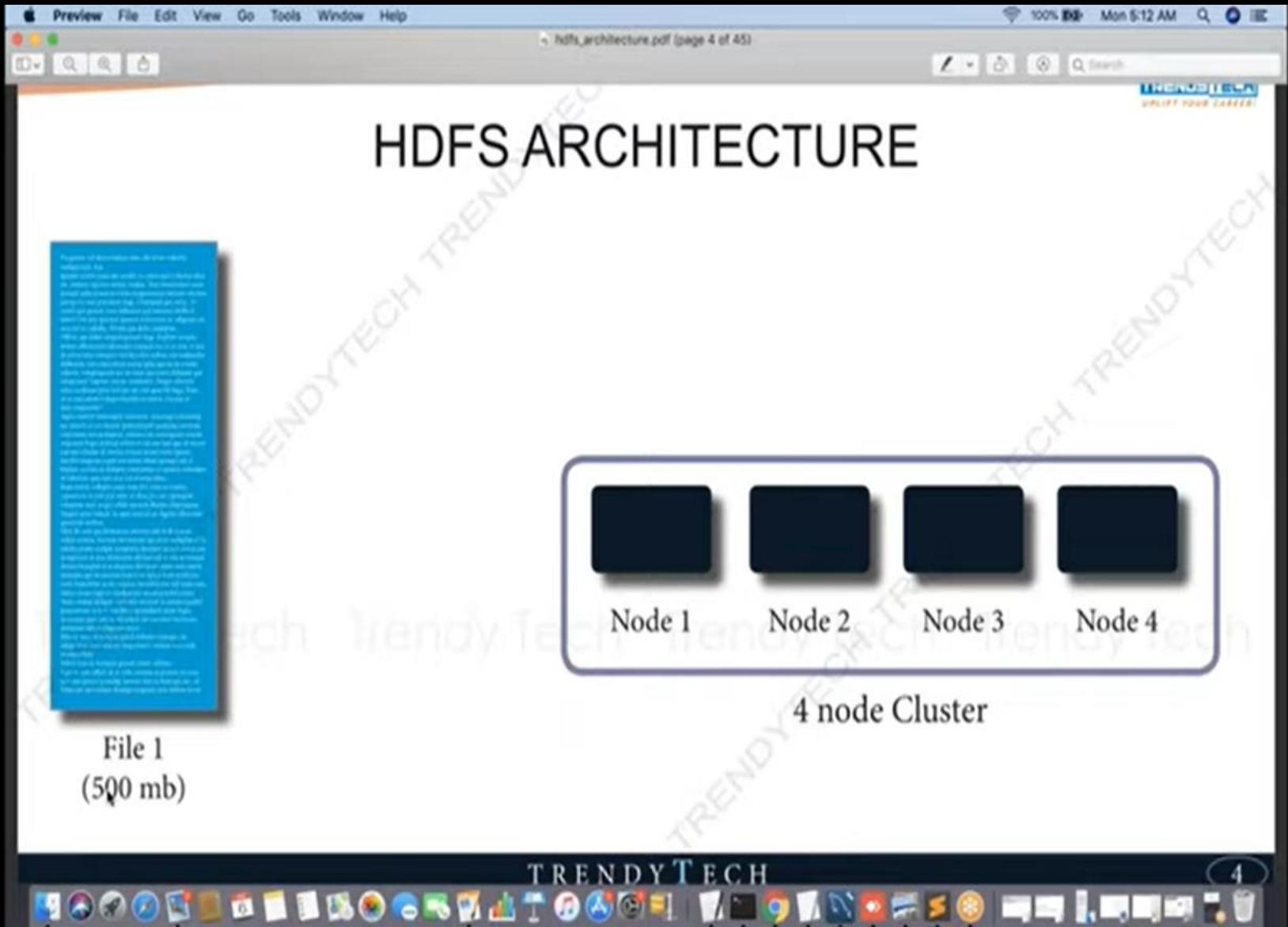
- During the loss of connectivity between name and data nodes, each data node sends periodic heartbeat messages to its NameNodes so the latter can detect loss of connectivity.
 - Data stored on a data node is no longer available to an HDFS client from that node, which is effectively removed from the system.
 - If the death of a node causes the replication factor of data blocks to drop below their minimum value, the NameNode initiates additional replication to normalized state.
- 

HDFS ARCHITECTURE

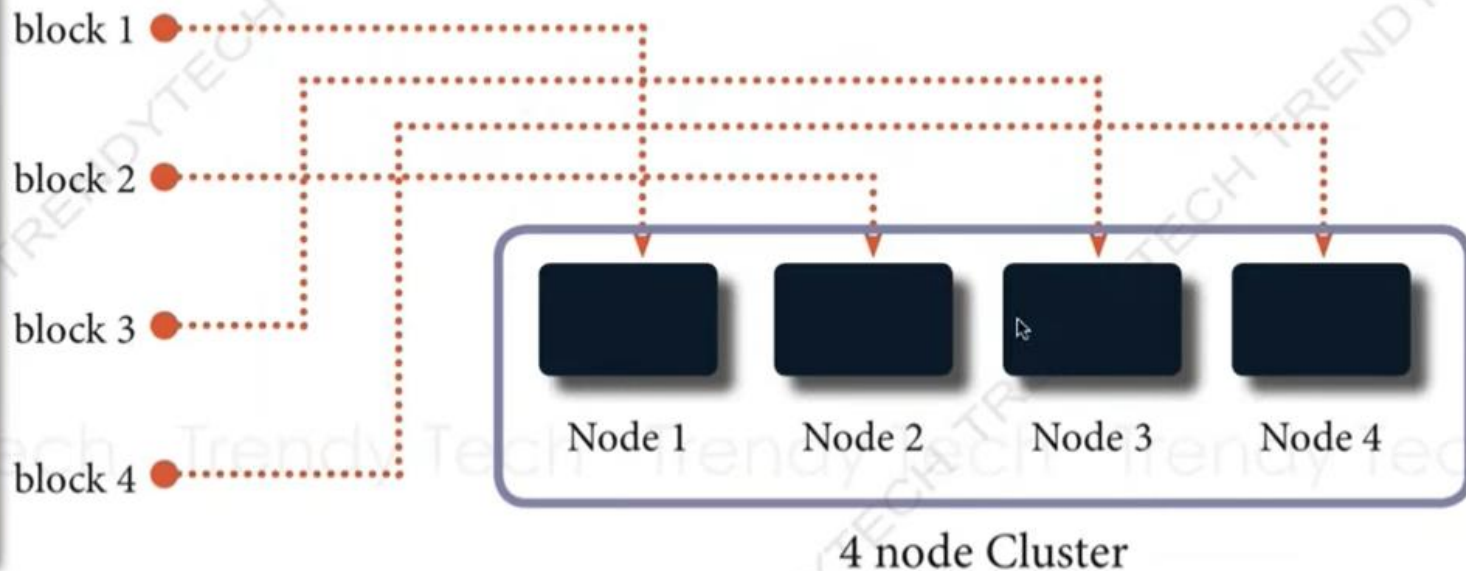
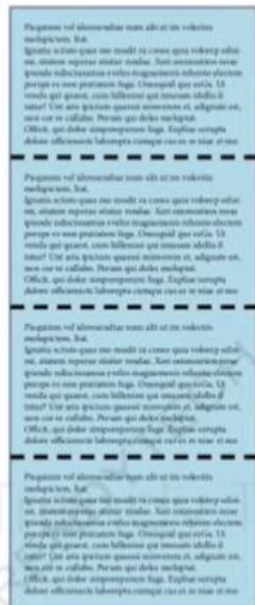


4 Node Cluster





HDFS ARCHITECTURE



File 1
(500 mb)

Preview File Edit View Go Tools Window Help

hdfs_architecture.pdf (page 8 of 45)

HDFS ARCHITECTURE

HDFS Block Size

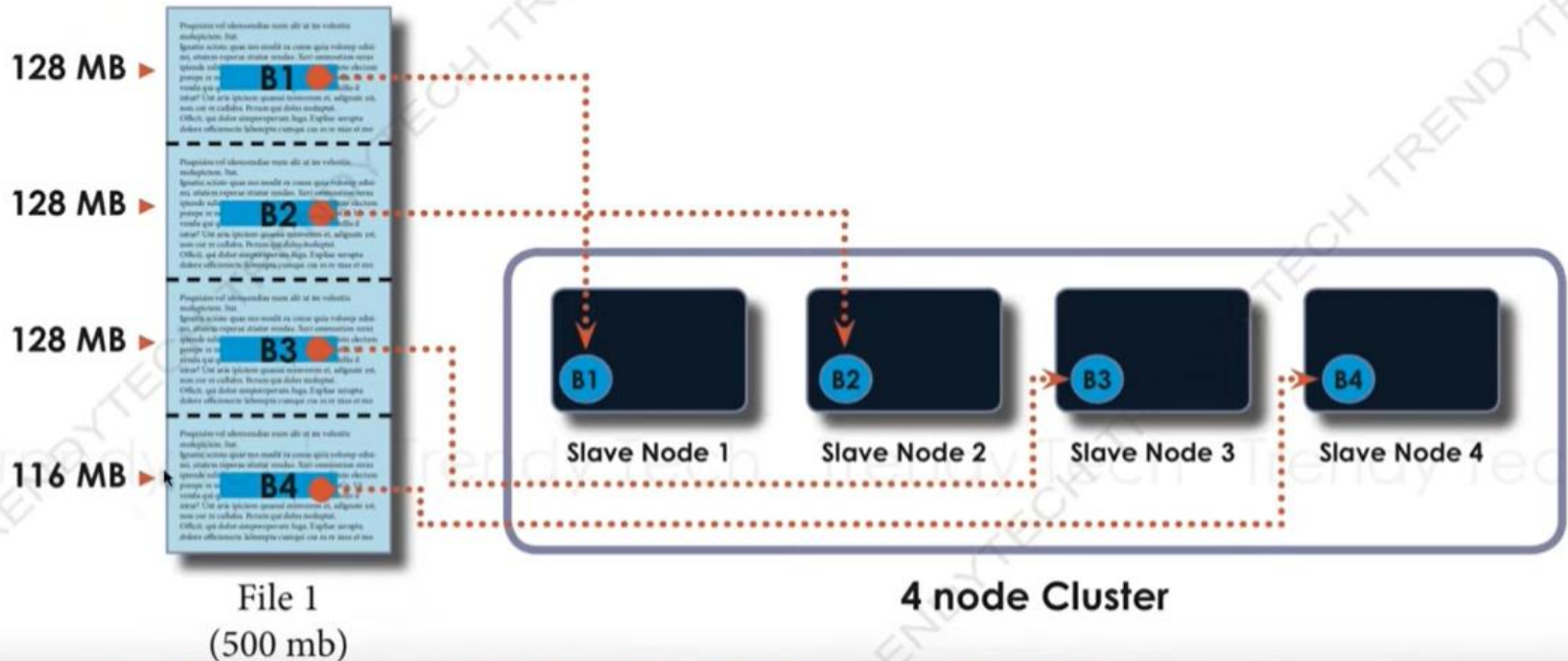
The diagram illustrates the evolution of HDFS block sizes. It is divided into two sections by a vertical dotted line. The left section, labeled 'older version', shows a dark blue oval representing Hadoop 1 with a 64 MB block size. The right section, labeled 'current version', shows a similar dark blue oval representing Hadoop 2 with a 128 MB block size. A green checkmark is placed above the Hadoop 2 oval, indicating it is the preferred or current standard. The text 'HDFS ARCHITECTURE' and 'HDFS Block Size' are centered at the top of the diagram area.

Version	Block Size
Hadoop 1 (older version)	64 MB
Hadoop 2 (current version)	128 MB

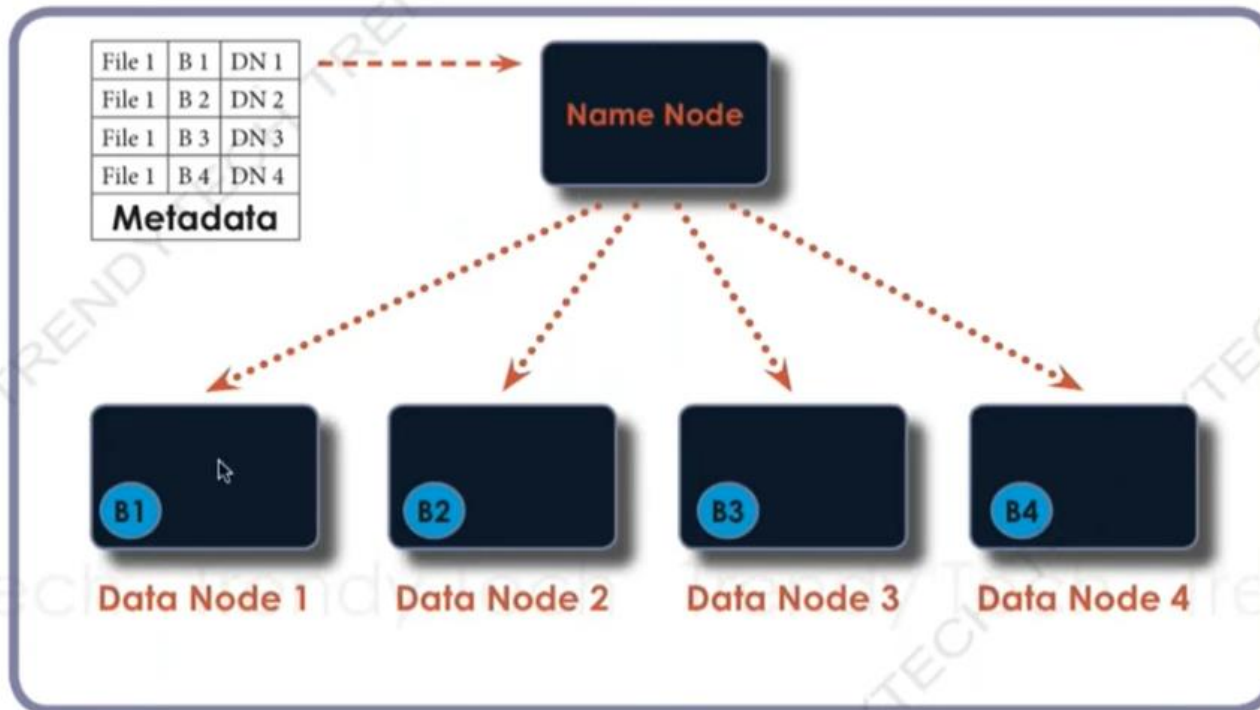
TRENDY TECH

8

HDFS ARCHITECTURE

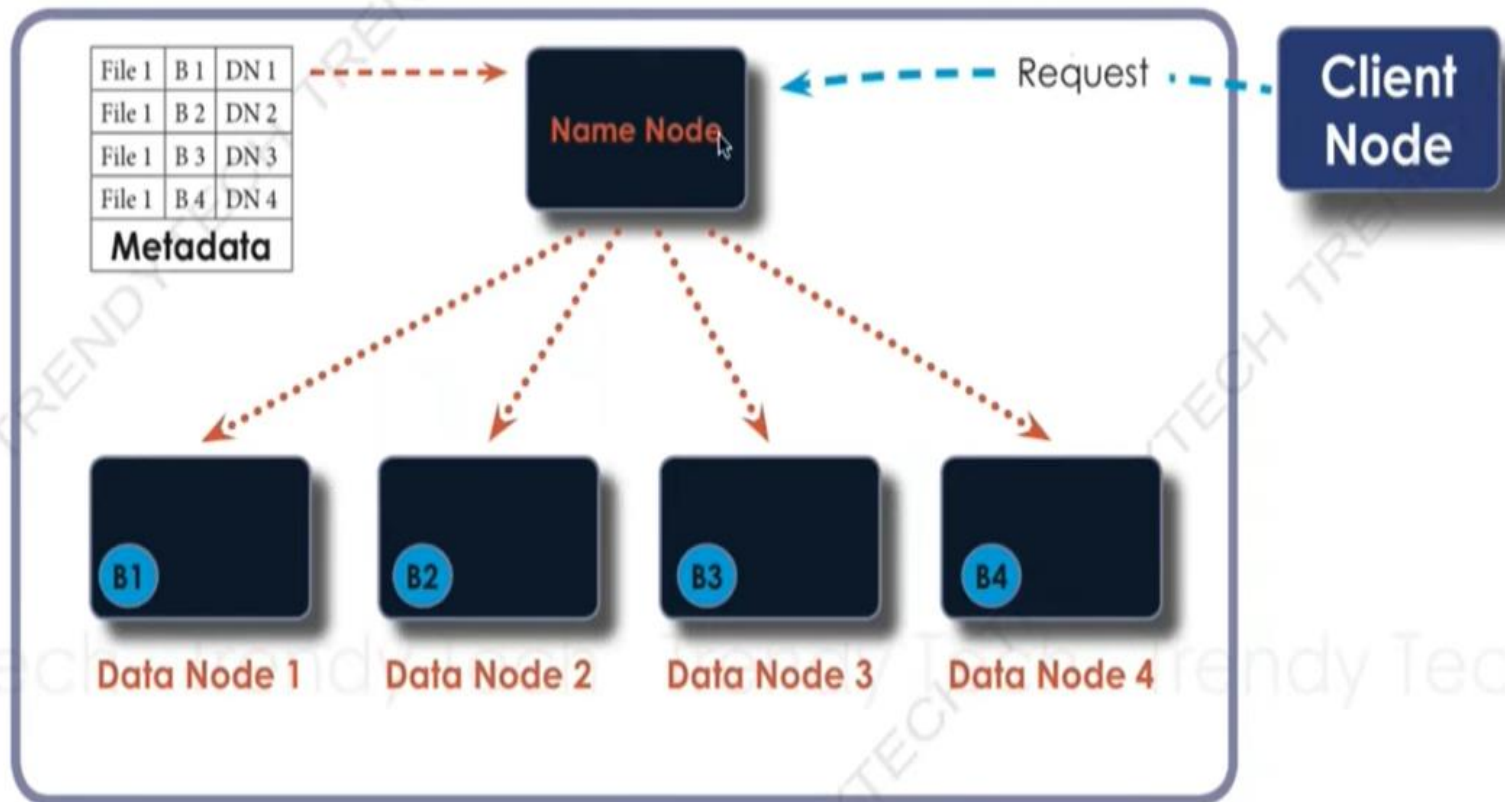


HDFS ARCHITECTURE



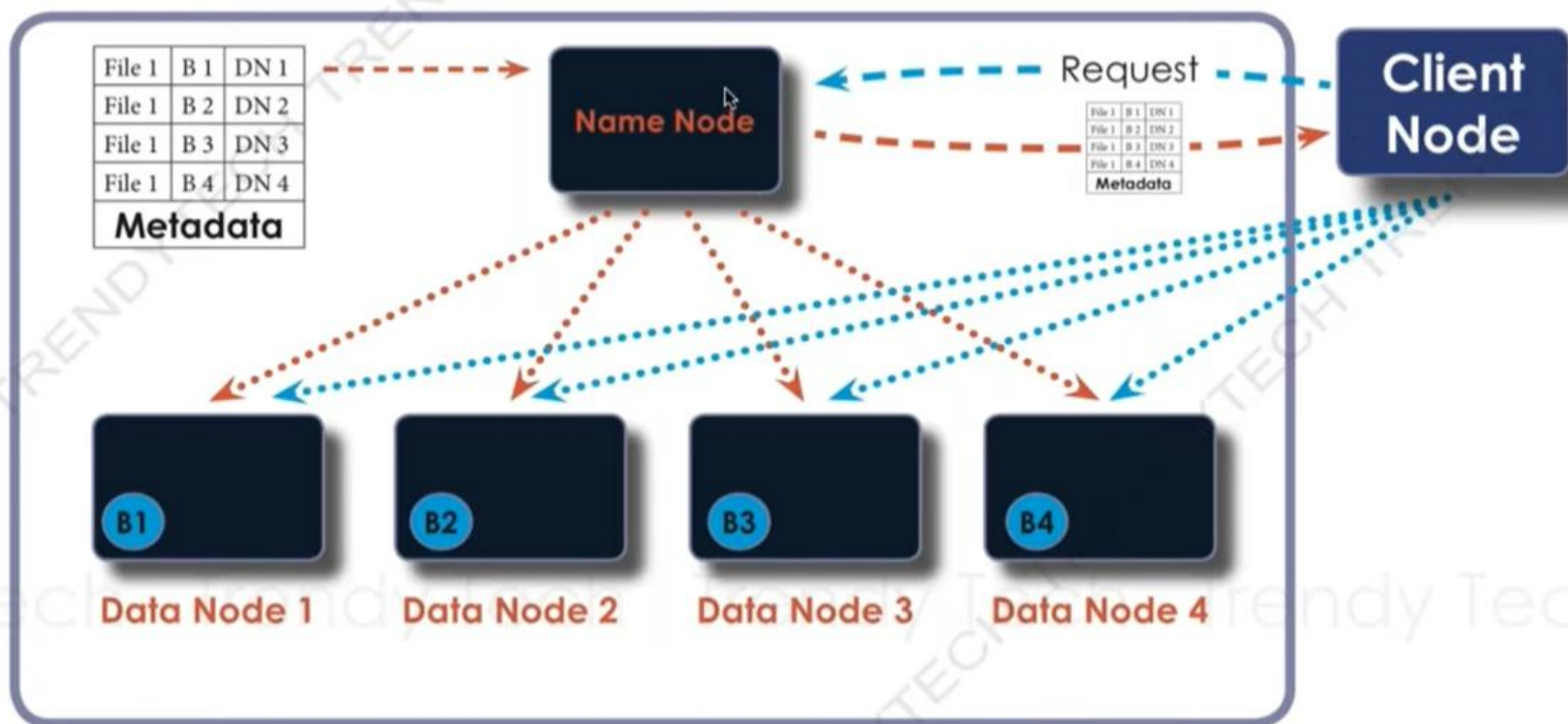
4 node Cluster

HDFS ARCHITECTURE



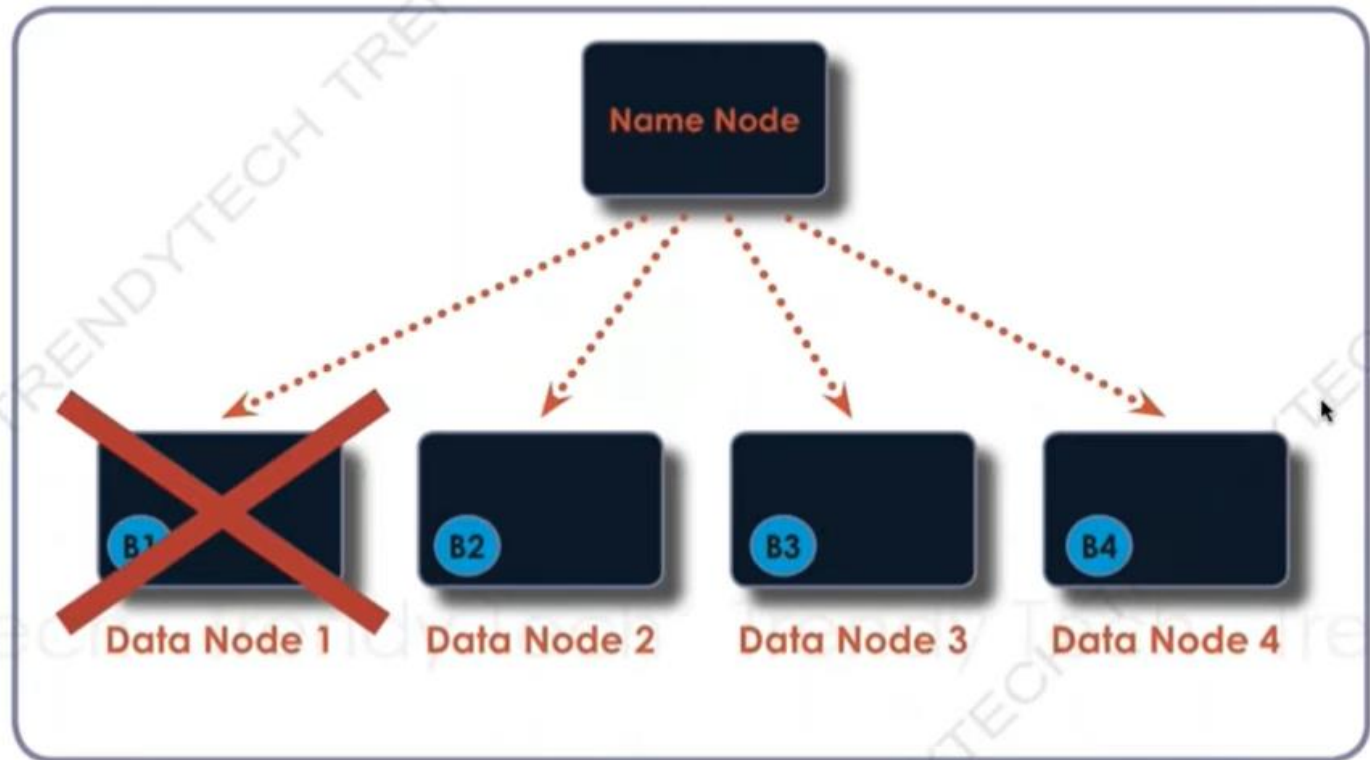
4 node Cluster

HDFS ARCHITECTURE



4 node Cluster

HDFS ARCHITECTURE



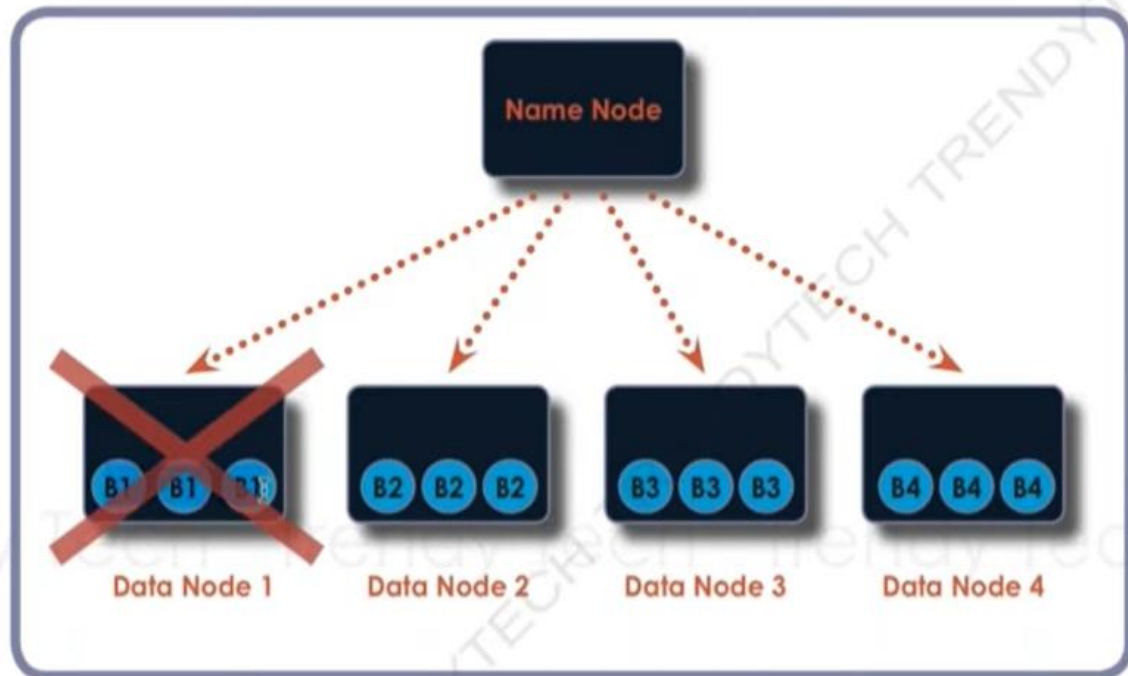
4 node Cluster

HDFS ARCHITECTURE

Replication Factor

Default Hadoop
Replication Factor:

3



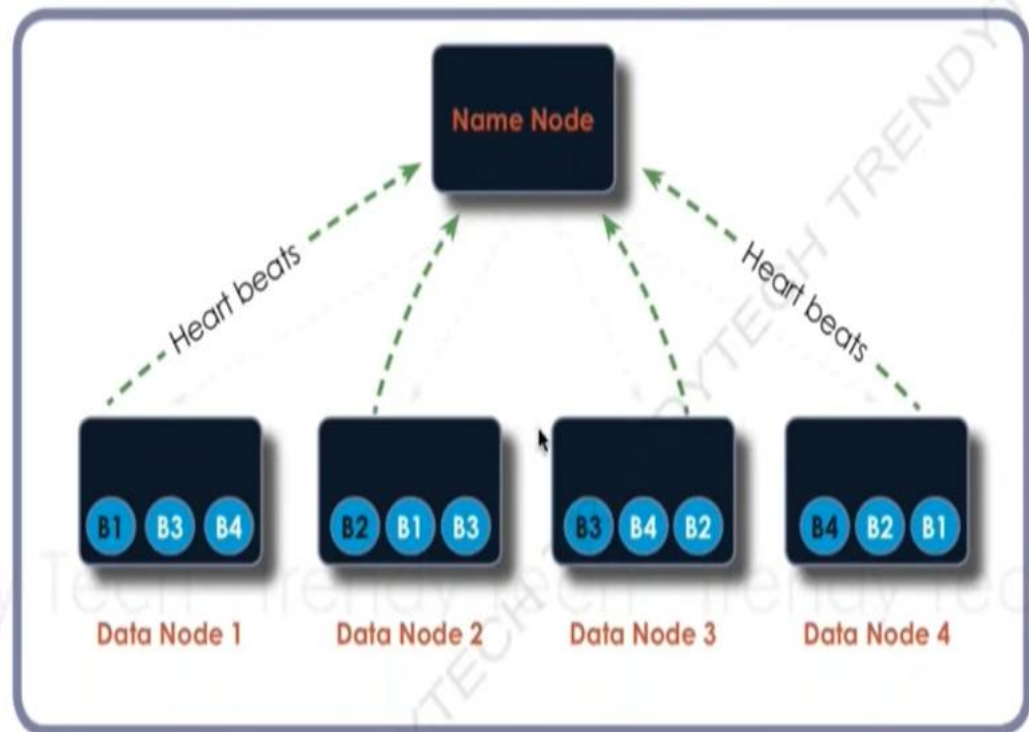
4 node Cluster

HDFS ARCHITECTURE

Heart Beat

Each Data Node sends heart beats to Name Node in every 3 seconds

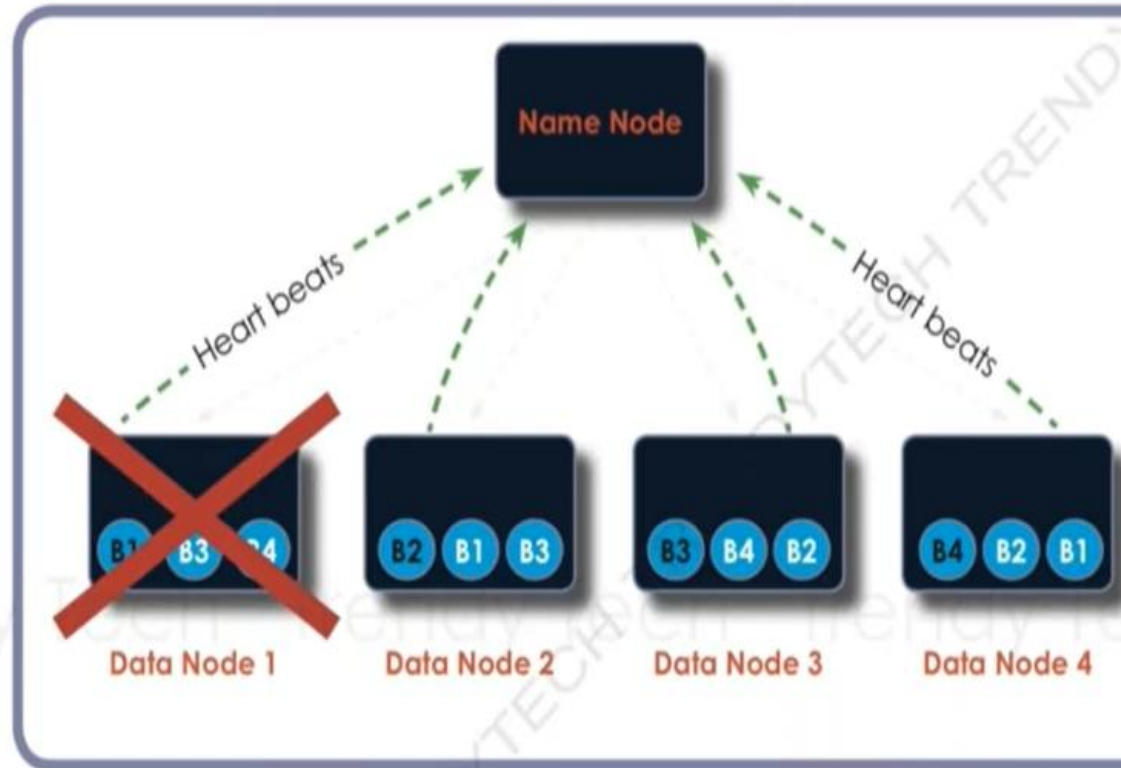
If a Name Node doesn't receive 10 consecutive heart beats, it assumes that the Data Node is dead or running very slow



4 node Cluster

Fault Tolerance

If a Data Node goes down
the replication factor came
down to < 3



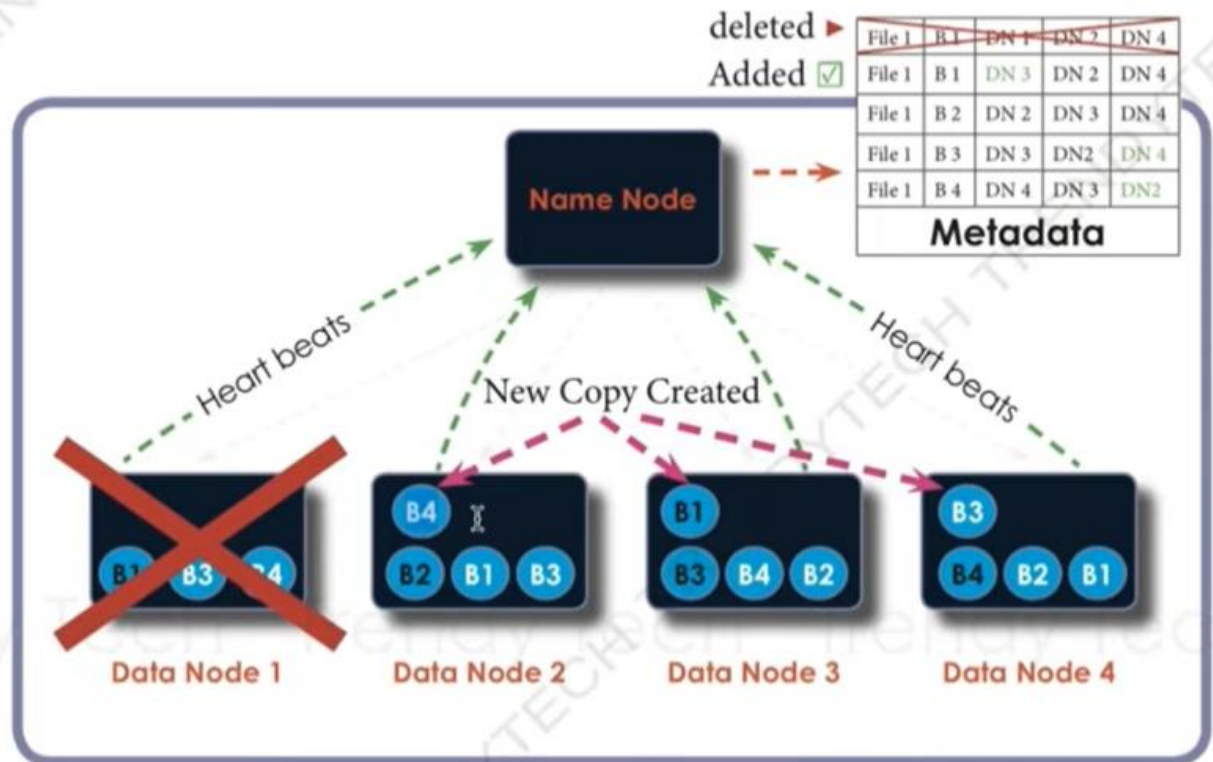
4 node Cluster

HDFS ARCHITECTURE

Fault Tolerance

If a Data Node goes down
the replication factor came
down to < 3


Name Node will create one
more copy to maintain the
replication factor



4 node Cluster



History of YARN

- The data processing component is separated from the resource management and scheduling components of MapReduce.
 - This helps Hadoop ecosystem to run interactive queries and streaming data applications simultaneously through MapReduce batch jobs.
- 

HADOOP 1

MapReduce
(Resource Management +
Data Processing)



HDFS



HADOOP 2


MapReduce +
Other Types of Jobs



YARN
(Resource Management)



HDFS

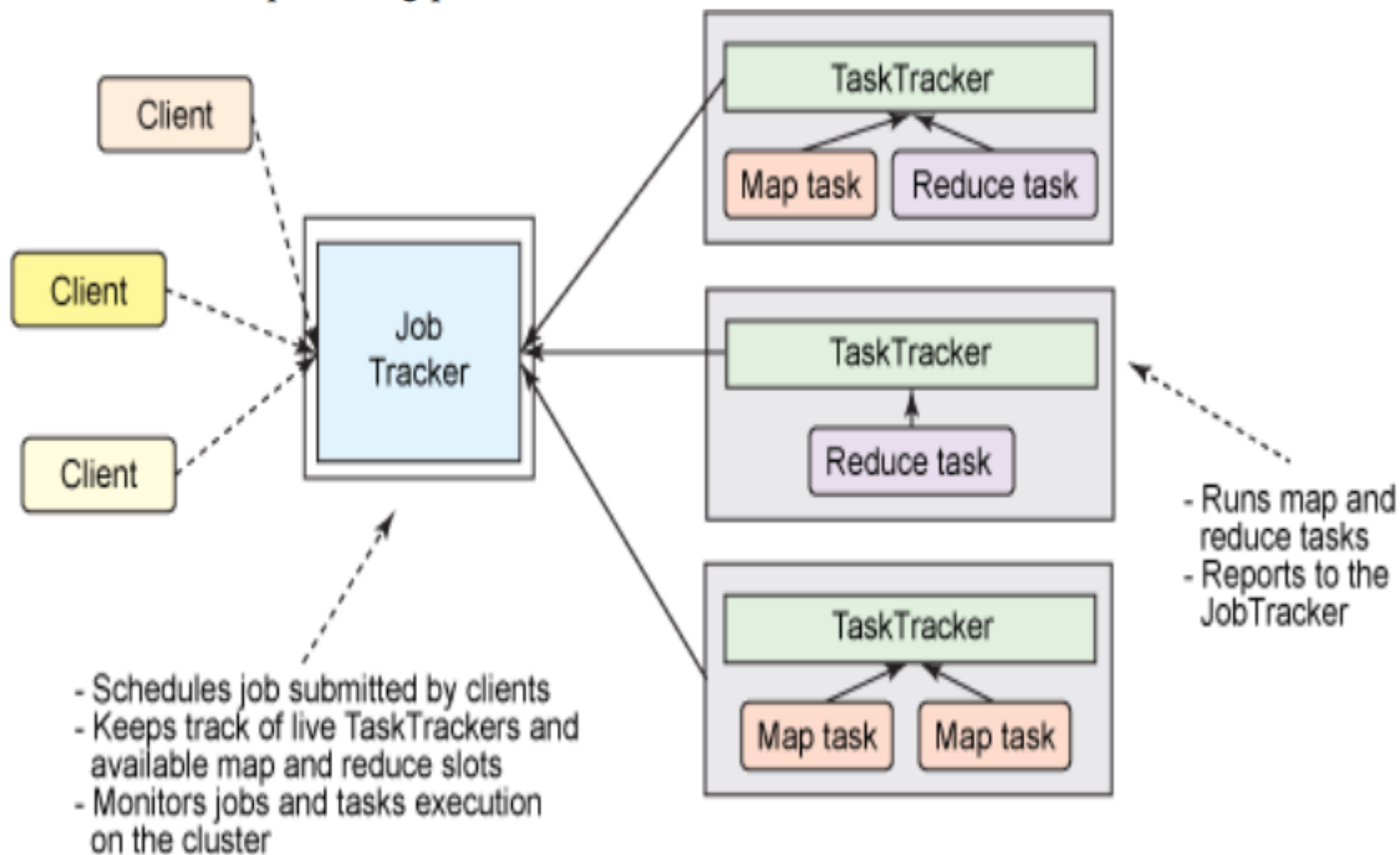
- 
- 
- In *Hadoop 1*, there is HDFS which is used for storage and top of it, Map Reduce which works as Resource Management as well as Data Processing. Due to this workload on Map Reduce, it will affect the performance.
 - In *Hadoop 2*, there is again HDFS which is again used for storage and on the top of HDFS, there is YARN which works as Resource Management. It basically allocates the resources and keeps all the things going on.

- 
- 
- applications of different types on multiple platforms simultaneously.
 - Hadoop 1 with core MapReduce processing provides a way in which MapReduce programs could be written in various languages such as Java, Python, Ruby on Rails etc.
 - The issue with Hadoop 1 is that programs are run on the basis of MapReduce processing cannot deal with all big data processing problems.

- 
- MapReduce is a programming model to process large amounts of data.
 - It divides large data sets into multiple small datasets distributed over a number of mapper and reducer nodes used for processing data.
 - The MapReduce consists of a JobTracker node that is also called the master and multiple TaskTracker nodes known as slaves.


- 
- The JobTracker performs the resource management tasks which include managing the slaves, tracking the availability and consumption of resources and managing the job life cycle.
- 



- 
- TaskTracker nodes, on the other hand, perform fundamental operations that include breaking down of tasks on the orders of JobTracker and providing periodic status information to the Jobtracker.
- 



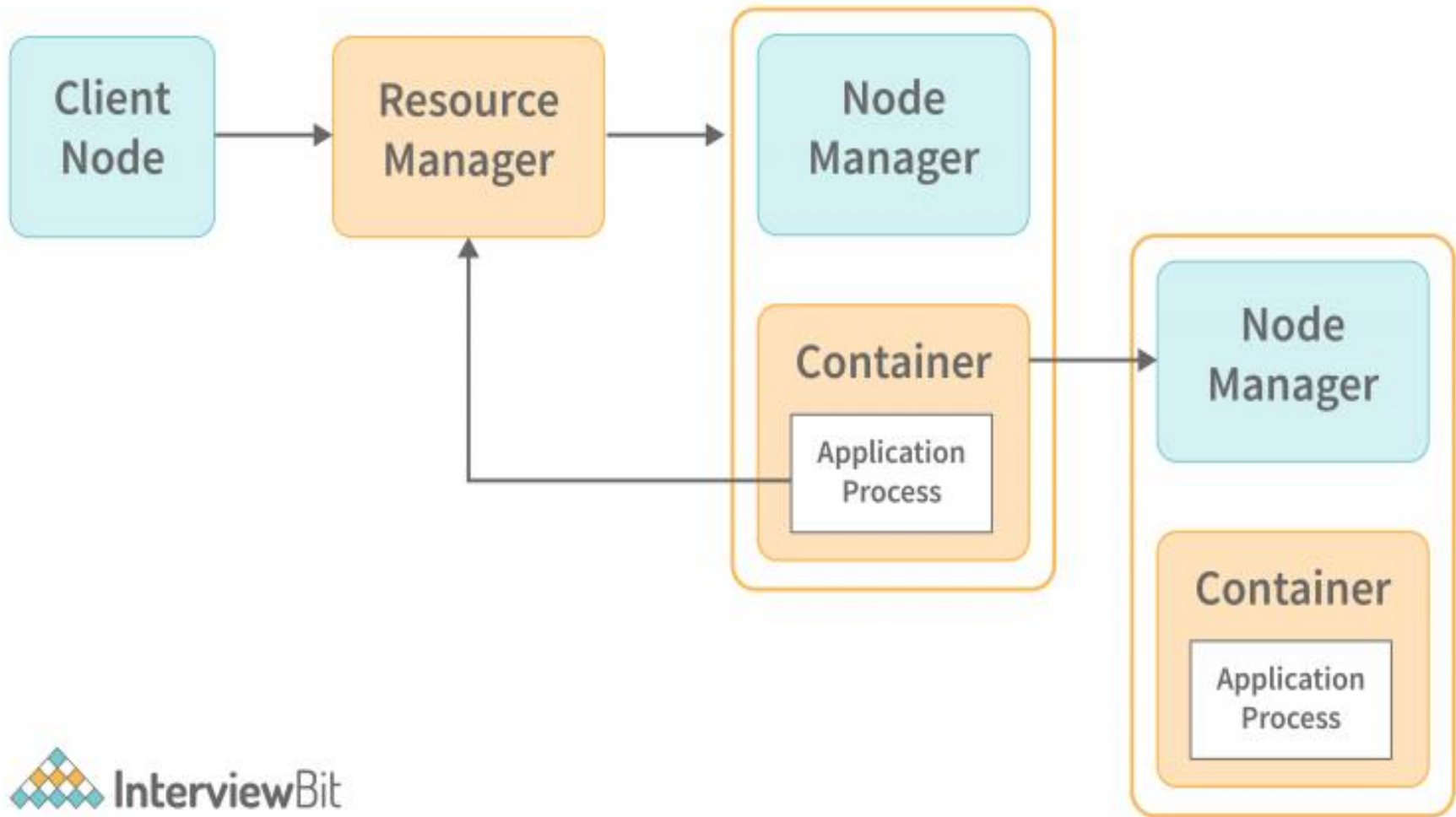


Advantages of YARN

- YARN provides efficient resource utilization.
 - The available resources are scheduled to perform map and reduce tasks per requirement instead of prior allocation of resources for map and reduce job execution
 - Multiple applications that share common resources can be run simultaneously.
 - The resource management and scheduling tasks of MapReduce are separated from data processing component by YARN.
- 


- 
- 
- Hence, a large variety of data processing tasks and wide range of applications are supported by YARN
 - YARN provides backward compatibility i.e. applications developed on Hadoop 1 can run on Hadoop 2 directly
 - Managing of resources and monitoring of jobs are assigned to two components called resource manager and application manager



YARN Architecture






Resource Manager

- The Resource Manager is described as the ultimate authority that arbitrates resources among all the applications in the system. The resource manager consists of two parts:
 1. **Applications Manager:** is responsible for accepting job submissions and starting a container for an entity called the ApplicationMaster. It also restarts the ApplicationMaster container if the container fails.
- 

- 
- 
2. **Scheduler:** The scheduler is responsible for allocating resources such as disk, CPU, and network running applications, subject to restrictions imposed by queues and capacity.




Containers

- A container is a set of physical resources on a single node.
 - It contains of memory, CPU and disks.
 - Depending on the resources in the node, a node can have multiple containers that are assigned to specific ApplicationManager.
 - It is supervised by a NodeManager and supervised by a ResourceManager
- 




Node Manager

- It is a per-machine slave, which is responsible for launching the application containers and reporting the status of resource usage to the ResourceManager.
 - NodeManager manages each node with a YARN cluster.
 - It provides per-node services within the cluster.
 - These services range from managing a container to monitoring resources and tracking the health of its node.
- 




YARN Schedulers

- There are two types of schedulers commonly available with YARN for negotiating resources with the ApplicationManager via ResourceManager.
- 



a. CapacityScheduler

- Its purpose is to allow multitenancy and share resources between multiple organizations and applications on the same cluster.
 - The job of CapacityScheduler is to make sure that any user, application should not consume a lot of resources available in a cluster.
- 



b. FairScheduler

- FairScheduling is a method of assigning resources to applications via ApplicationManager such that all applications get an equal share of resources during their course of running.
- 