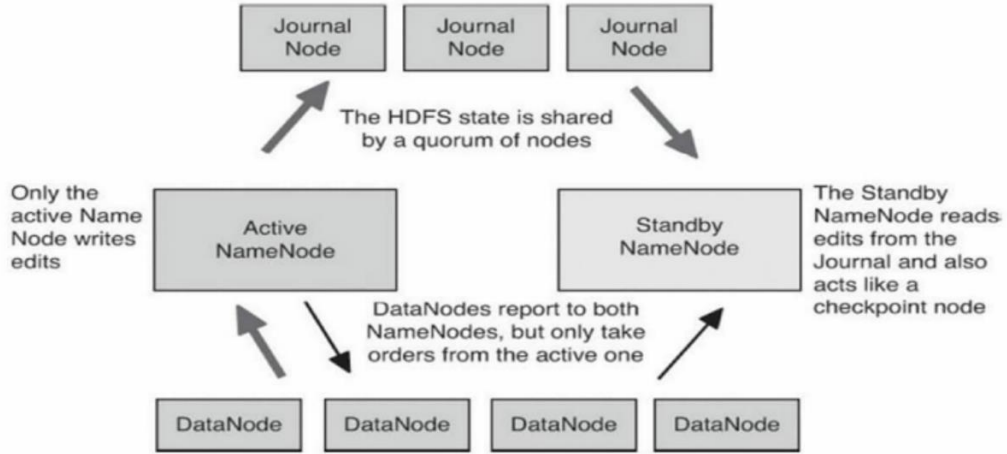


Internal Assessment Test 2 – December 2022

Sub:	BIG DATA AND ANALYTICS	Sub Code:	18CS72	Branch:	ISE
Date:	02/12/2022	Duration:	90 min's	Max Marks:	50
				Sem/Sec:	VII / A, B & C
<u>Answer any FIVE FULL Questions</u>					MARKS
1	<p>What are NameNode High Availability and NameNode Federation? Scheme: NameNode High Availability and NameNode Federation = 5+5=10 NameNode High Availability</p> <ul style="list-style-type: none"> • The NameNode was a single point of failure that could bring down the entire Hadoop cluster. • NameNode hardware often employed redundant power supplies and storage to guard against such problems, but it was still susceptible to other failures. • The solution was to implement NameNode High Availability (HA) as a means to provide true failover service. • As shown in Figure 3.3, an HA Hadoop cluster has two (or more) separate NameNode machines. • Each machine is configured with exactly the same software. • One of the NameNode machines is in the Active state, and the other is in the Standby state. • Like a single NameNode cluster, the Active NameNode is responsible for all client HDFS operations in the cluster. • The Standby NameNode maintains enough state to provide a fast failover (if required). <div style="text-align: center;">  <p>The diagram illustrates the HDFS High Availability design. At the top, three 'Journal Node' boxes are connected by arrows to a central text: 'The HDFS state is shared by a quorum of nodes'. Below this, two 'NameNode' boxes are shown: 'Active NameNode' on the left and 'Standby NameNode' on the right. An arrow points from the Active NameNode to the Journal Nodes, with the text 'Only the active Name Node writes edits'. Another arrow points from the Standby NameNode to the Journal Nodes, with the text 'The Standby NameNode reads edits from the Journal and also acts like a checkpoint node'. At the bottom, four 'DataNode' boxes are shown. Arrows point from the DataNodes to both the Active and Standby NameNodes, with the text 'DataNodes report to both NameNodes, but only take orders from the active one'.</p> </div> <p style="text-align: center;">Figure 3.3 HDFS High Availability design</p>	[10]	CO2	L2	

HDFS NameNode Federation

Another important feature of HDFS is NameNode Federation. Older versions of HDFS provided a single namespace for the entire cluster managed by a single NameNode. Thus, the resources of a single NameNode determined the size of the namespace. Federation addresses this limitation by adding support for multiple NameNodes/namespaces to the HDFS file system. The key benefits are as follows: Namespace scalability. HDFS cluster storage scales horizontally without placing a burden on the NameNode. Better performance. Adding more NameNodes to the cluster scales the file system read/write operations throughput by separating the total namespace. System isolation. Multiple NameNodes enable different categories of applications to be distinguished, and users can be isolated to different namespaces.

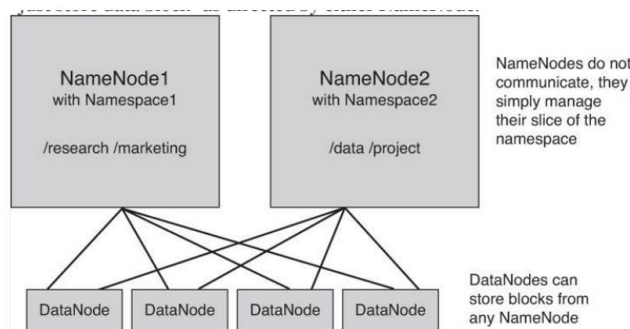


Figure 3.4 HDFS NameNode Federation example

Figure 3.4 illustrates how HDFS NameNode Federation is accomplished. NameNode1 manages the /research and /marketing namespaces, and NameNode2 manages the /data and /project namespaces. The NameNodes do not communicate with each other and the DataNodes “just store data block” as directed by either NameNode.

2

Explain Hadoop MapReduce and YARN with neat diagram.

Scheme: MapReduce and YARN = 5+5=10

Solution:

MapReduce functions as integral part of the Hadoop physical organization.

MapReduce is a programming model for distributed computing.

Mapper means software for doing the assigned task after organizing the data blocks imported using the keys. A key specifies in a command line of Mapper. The command maps the key to the data, which an application uses.

Reducer means software for reducing the mapped data by using the aggregation, query or user-specified function. The reducer provides a concise cohesive response for the application.

Aggregation function means the function that groups the values of multiple rows together to result a single value of more significant meaning or measurement. For example, function such as count, sum, maximum, minimum, deviation and standard deviation.

Querying function means a function that finds the desired values. For example, function for finding a best student of a class who has shown the best performance in examination.

MapReduce allows writing applications to process reliably the huge amounts of data, in parallel, on clusters of servers. The cluster size does not limit as such to process in parallel. The parallel programs MapReduce are useful for performing large scale data analysis using multiple machines in the cluster.

Features of MapReduce framework are as follows:

- Provides automatic parallelization and distribution of computation based on several processors.
- Processes data stored on distributed clusters of DataNodes and racks
- Allows processing large amount of data in parallel

[10]

CO2

L2

- Provides scalability for usages of large number of servers
- Provides MapReduce batch-oriented programming model in Hadoop version 1
- Provides additional processing modes in Hadoop 2 YARN-based system and enables required parallel processing. For example, for queries, graph databases, streaming data, messages, real-time OLAP and ad hoc analytics with Big Data 3V characteristics.

1. Hadoop MapReduce Framework

MapReduce provides two important functions. The distribution of job based on client application task or users query to various nodes within a cluster is one function. The second function is organizing and reducing the results from each node into a cohesive response to the application or answer to the query. The processing tasks are submitted to the Hadoop, The Hadoop framework in turns manages the task of issuing jobs, job completion, and copying data around the cluster between the DataNodes with the help of JobTracker. Daemon refers to highly dedicated program that runs in the background in a system. The user does not control or interact with that. An example is MapReduce in Hadoop system (Collins English language dictionary gives one of Daemon meaning as 'a person who concentrates very hard or is very skilled at an activity and puts in lot of energy into it). MapReduce runs as per assigned Job by JobTracker, which keeps track of the job submitted for execution and runs Task Tracker for tracking the tasks. MapReduce programming enables job scheduling and task execution as follows:

A client node submits a request of an application to the JobTracker. A JobTracker is a Hadoop daemon, (background program). The following are the steps on the request to MapReduce:

- estimate the need of resources for processing that request,
- analyze the states of the slave nodes,
- place the mapping tasks in queue,
- monitor the progress of task, and on the failure, restart the task on slots of time available.

2. MapReduce Programming Model

MapReduce program can be written in any language including JAVA, C++ PIPES or Python. Map function of MapReduce program do mapping to compute the data and convert the data into other data sets (distributed in HDFS). After the Mapper computations finish, the Reducer function collects the result of map and generates the final output result. MapReduce program can be applied to any type of data, i.e., structured or unstructured stored in HDFS. The input data is in the form of file or directory and is stored in the HDFS. The MapReduce program performs two jobs on this input data, the Map job and the Reduce job. They are also termed as two phases - Map phase and Reduce phase. The map job takes a set of data and converts it into another set of data. The individual elements are broken down into tuples (key/value pairs) in the resultant set of data. The reduce job takes the e output from a map as input and combines the data tuples into a smaller set of tuples. Map and reduce jobs run in isolation from one another. As the sequence of the name MapReduce implies, the reduce job is always performed after the map job. The MapReduce v2 uses YARN based resource scheduling which simplifies the software development. Here, the jobs can be split across almost any number of servers.

YARN is a resource management platform. It manages computer resources. The platform is responsible for providing the computational resources, such as CPUs, memory, network I/O which are needed when an application executes. An application task has a number of sub-tasks. YARN manages the schedules for running of the sub-tasks. Each sub-task uses the resources in allotted time intervals. Hadoop YARN for management and scheduling of resources for parallel running of application tasks.

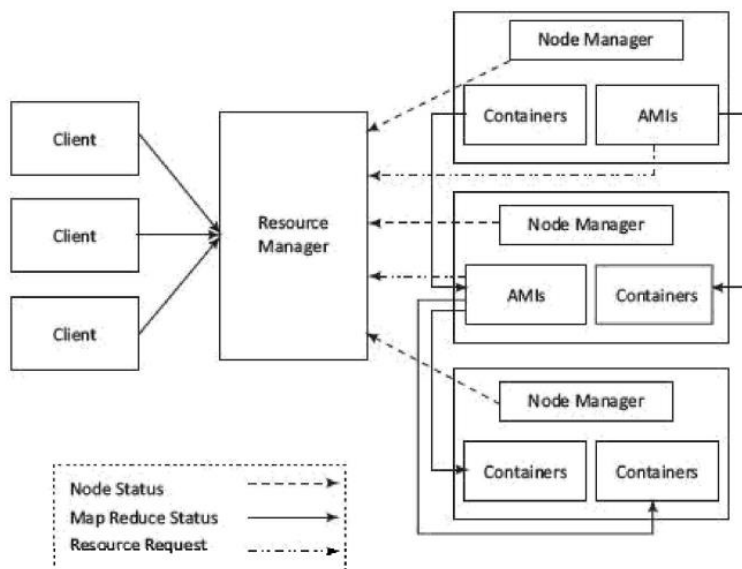
YARN separates the resource management and processing components. YARN stands for Yet Another Resource Negotiator. An application consists of a number of tasks. Each task can consist of a number of sub tasks (threads), which run in parallel at the nodes in the cluster. YARN enables running of multi-threaded applications. YARN manages and allocates the resources for the application sub-tasks and submits the resources for them at the Hadoop system.

YARN-based execution model. The figure shows the YARN components – Client, Resource Manager (RM), Node Manager (NM), Application Master (AM) and Containers. YARN components namely, Client, Resource Manager (RM), Node Manager (RM), Application Master (AM) and Containers. List of actions of YARN resource allocation and scheduling functions is as follows: A MasterNode has two components: (i) Job History Server and (ii) Resource Manager (RM).

A Client Node submits the request of an application to the RM. The RM is the master. One RM exists per cluster. The RM keeps information of all the slave NMs. Information is about the location (Rack Awareness) and the number of resources (data blocks and servers) they have. The RM also renders the Resource Scheduler service that decides how to assign the resources. It, therefore, performs resource management as well as scheduling. Multiple NMs are at a cluster. An NM creates an AM instance (AMI) and starts up. The AMI initializes itself and registers with the RM. Multiple AMIs can be created in an AM.

The AMI performs role of an Application Manager (AppIM), that estimates the resources requirement for running an application program or sub-task. The ApplMs send their requests for the necessary resources to the RM. Each NM includes several containers for uses by the subtasks of the application. NM is a slave of the infrastructure. It signals whenever it initializes. All active NMs send the controlling signal periodically to the RM signaling their presence.

Each NM assigns a container(s) for each AMI. The container(s) assigned at an instance may be at same NM or another NM. ApplM uses just a fraction of the resources available. The ApplM at an instance uses the assigned container(s) for running the application sub-task. RM allots the resources to AM, and thus to ApplMs for using assigned containers on the same or other NM for running the application subtasks in parallel.



Explain Shared-Nothing Architecture for Big Data Tasks with neat diagrams.

Scheme: 4 Models + 4 ways = 8+2 = 10M

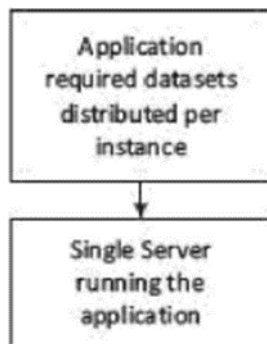
Solution:

Choosing the Distribution Model

1. Single Server Model
2. Sharding very large Databases
3. Master-Slave Distribution Model
4. Peer-to-Peer Distribution Model

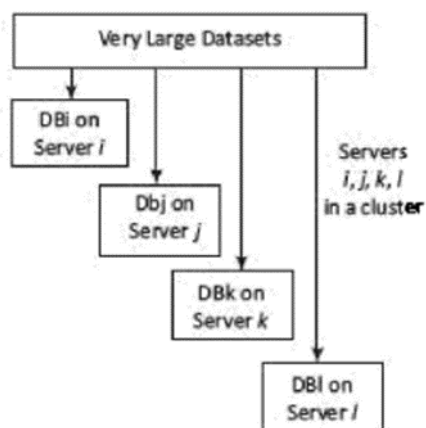
1. Single Server Model

Simplest distribution option for NoSQL data store and access is Single Server Distribution (SSD) of an application. A graph database processes the relationships between nodes at a server. The SSD model suits well for graph DBs. Aggregates of datasets may be key-value, column-family or BigTable data stores which require sequential processing. These data stores also use the SSD model.



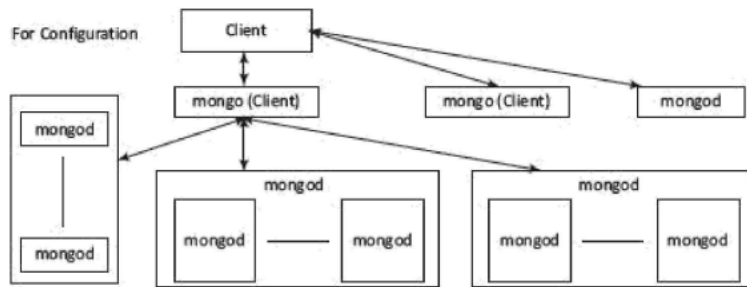
2. Sharding very large Databases

Figure 3.9(b) shows sharding of very large datasets into four divisions, each running the application on four i, j, k and l different servers at the cluster. DB_i, DB_j, DB_k and DB_l are four shards.



3. Master-Slave Distribution Model

A node serves as a master or primary node and the other nodes are slave nodes. Master directs the slaves. Slave nodes data replicate on multiple slave servers in

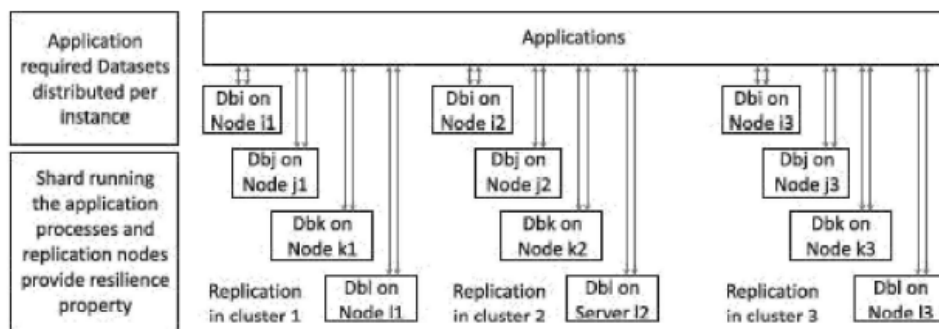


Master Slave Distribution (MSD) model. When a process updates the master, it updates the slaves also. A process uses the slaves for read operations. Processing performance improves when process runs large datasets distributed onto the slave nodes. Figure 3.10 shows an example of MongoDB. MongoDB database server is *mongod* and the client is *mongo*.

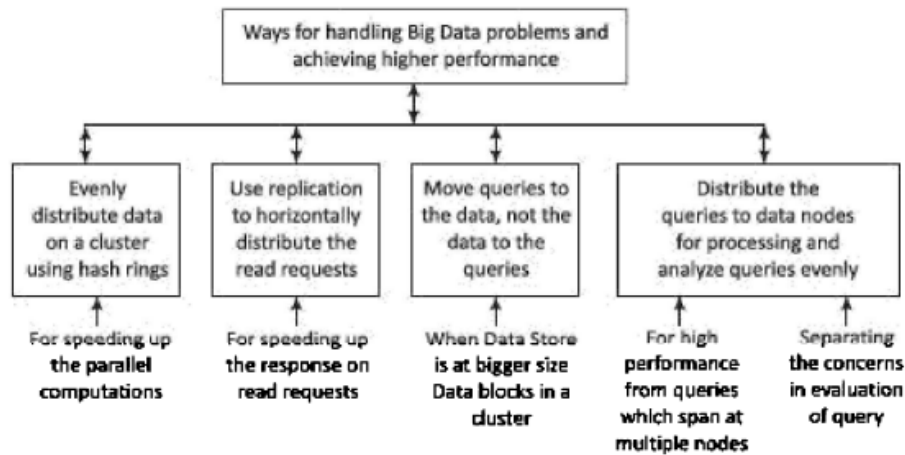
4. Peer-to-Peer Distribution Model

Peer-to-Peer distribution (PPD) model and replication show the following characteristics: (1) All replication nodes accept read request and send the responses. (2) All replicas function equally. (3) Node failures do not cause loss of write capability, as other replicated node responds.

Cassandra adopts the PPD model. The data distributes among all the nodes in a cluster.



Ways of handling Big Data Problems



4 Write short note on NoSQL Big Data Management and explain NoSQL Data store.

[10]

CO3

L2

Scheme: NoSQL Big Data Management = 10M

SQL is a programming language based on relational algebra. It is a declarative language and it defines the data schema. SQL creates databases and RDBMSs. RDBMS uses tabular data store with relational algebra, precisely defined operators with relations as the operands. Relations are a set of tuples. Tuples are named attributes. A tuple identifies uniquely by keys called candidate keys. Transactions on SQL databases exhibit ACID properties. ACID stands for atomicity, consistency, isolation and durability.

ACID Properties in SQL Transactions

- *Atomicity* of transaction means all operations in the transaction must complete, and if interrupted, then must be undone (rolled back).
- *Consistency in transactions* means that a transaction must maintain the integrity constraint, and follow the consistency principle.
- *Isolation* of transactions means two transactions of the database must be isolated from each other and done separately.

Durability means a transaction must persist once completed

Triggers, Views and Schedules in SQL Databases

Trigger is a special stored procedure. Trigger executes when a specific action(s) occurs within a database, such as change in table data or actions such as UPDATE, INSERT and DELETE. For example, a Trigger store procedure inserts new columns in the columnar family data store.

View refers to a logical construct, used in query statements. A View saves a division of complex query instructions and that reduces the query complexity. Viewing of a division is similar to a view of a table.

Schedule refers to a chronological sequence of instructions which execute concurrently. When a transaction is in the schedule then all instructions of the transaction are included in the schedule.

Join in SQL Databases

SQL databases facilitate combining rows from two or more tables, based on the related columns in them. *Combining* action uses *Join* function during a database transaction. *Join* refers to a clause which combines. Combining the products (AND operations) follows next the

	<p>selection process. A Join operation does pairing of two tuples obtained from different relational expressions.</p> <p>NoSQL</p> <p>A new category of data stores is NoSQL (means Not Only SQL) data stores. NoSQL is an altogether new approach of thinking about databases, such as schema flexibility, simple relationships, dynamic schemas, auto sharding, replication, integrated caching, horizontal scalability of shards, distributable tuples, semi-structures data and flexibility in approach.</p> <p>Issues with NoSQL data stores are lack of standardization in approaches, processing difficulties for complex queries, dependence on eventually consistent results in place of consistency in all states.</p> <p>NoSQL data stores are considered as semi-structured data. Big Data Store uses NoSQL. NoSQL data store characteristics are as follows:</p> <ol style="list-style-type: none"> 1. NoSQL is a class of non-relational data storage system with flexible data model. Examples of NoSQL data-architecture patterns of datasets are key-value pairs, name/value pairs, Column family Big-data store, Tabular data store, Cassandra (used in Facebook/Apache), HBase, hash table [Dynamo (Amazon S3)], unordered keys using JSON (CouchDB), JSON (PNUTS), JSON (MongoDB), Graph Store, Object Store, ordered keys and semi-structured data storage systems. 2. NoSQL not necessarily has a fixed schema, such as table; do not use the concept of Joins (in distributed data storage systems); Data written at one node can be replicated to multiple nodes. Data store is thus fault-tolerant. The store can be partitioned into unshared shards. <p>Features in NoSQL Transactions NoSQL transactions have following features:</p> <ol style="list-style-type: none"> (i) Relax one or more of the ACID properties. (ii) Characterize by two out of three properties (consistency, availability and partitions) of CAP theorem, two are at least present for the application/service/process. (iii) Can be characterized by BASE properties. <p>Big Data NoSQL solutions use standalone-server, master-slave and peer-to-peer distribution models.</p> <p>Big Data NoSQL Solutions NoSQL DBs are needed for Big Data solutions. They play an important role in handling Big Data challenges. Table 3.1 gives the examples of widely used NoSQL data stores.</p>			
5	<p>Compare MongoDB and Cassandra database. Scheme: Comparison 10M</p>	[10]	CO3	L2

	Criteria	MongoDB	Cassandra			
	Data model	Document	Wide Columnar			
	Implementation language	C++	Java			
	Developer	MongoDB, Inc	Originally developed by Facebook then Apache TLP			
	Query language	Mongo shell	CQL			
	Cloud database service	MongoDB Atlas, ScaleGrid for MongoDB	Cassandra datastax			
	Data access API	Proprietary protocol using JSON	Thrift			
	Replication	Master-Slave	Peer to Peer			
	Data storage	Multiple data storage both in memory and on disk	Data storage on disk only			
6	<p>What are the features of MongoDB and apply the commands for the following operations in MongoDB.</p> <ul style="list-style-type: none"> • Create a student database • Check the existence of the student database • Create a collection with three fields such as USN, Name and Section in the student database. • Add an array with three documents to a collection created. • Display all the documents from the collection. • Remove a document whose “USN”:12345. <p>Scheme: Features + Commands = 4+6 = 10M</p> <p>Solution:</p> <p>MongoDB is (i) non-relational, (ii) NoSQL, (iii) distributed, (iv) open source, (v) document based, (vi) cross-platform, (vii) Scalable, (viii) flexible data model, (ix) Indexed, (x) multi-master, and (xi) fault tolerant. Document data store in JSON-like documents. The data store uses the dynamic schemas.</p> <ul style="list-style-type: none"> • Create a student database Use student • Check the existence of the student database db • Create a collection with three fields such as USN, Name and Section in the student database. db.student.insert ({ “USN”:12345, ”Name”:”Raju”, ”Section”:”A” }) • Add an array with three documents to a collection created. db.student.insert ([{ “USN”:12345, ”Name”:”Raju”, ”Section”:”A” }, { “USN”:12356, ”Name”:”Ravi”, ”Section”:”B” },]) 			[10]	CO3	L3

	<pre>{ "USN":12389, "Name":"Shahul", "Section":"C" }])</pre> <ul style="list-style-type: none">• Display all the documents from the collection. db.student.find()• Remove a document whose "USN":12345. db.student.remove("USN":12345))			
--	--	--	--	--

Faculty Signature

CCI Signature

HOD Signature