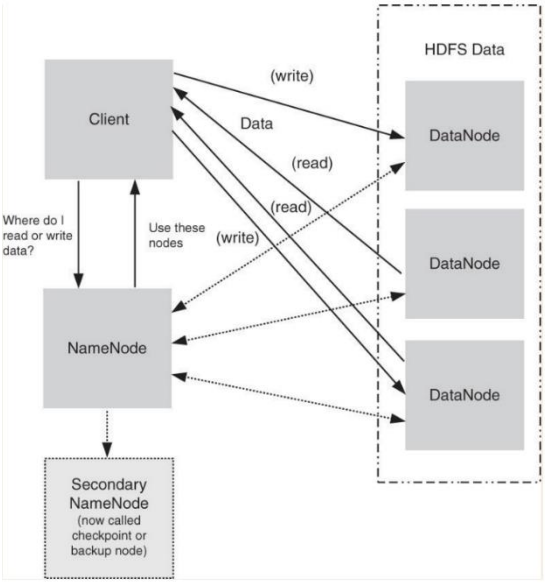
USN | | | | | | | | | | |

Internal Assessment Test 2 – December 2021

Sub:	BIG DATA AND ANALYTICS					Sub Code:	18CS72	Branch:	ISE
Date:	16/12/2021	Duration:	90 min's	Max Marks:	50	Sem / Sec:	VII / A, B & C		OBE

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1	**Explain in detail about HDFS Components with neat Diagrams.** **Scheme:** Components + Diagram + Explanation = 2+5+3 =10M **Solution:** HDFS COMPONENTS: i. HDFS Block Replication ii. HDFS Safe Mode iii. Rack Awareness iv. NameNode High Availability v. HDFS NameNode Federation vi. HDFS Checkpoints and Backups vii. HDFS Snapshots viii. HDFS NFS Gateway • The design of HDFS is based on two types of nodes: a NameNode and multiple DataNodes • NameNode manages all the metadata needed to store and retrieve the actual data from the DataNodes. • No data is actually stored on the NameNode. • The design is a master/slave architecture in which the master (NameNode) manages the file system namespace and regulates access to files by clients. • File system namespace operations such as opening, closing, and renaming files and directories are all managed by the NameNode • The NameNode also determines the mapping of blocks to DataNodes and handles DataNode failures • The NameNode manages block creation, deletion, and replication • The slaves (DataNodes) are responsible for serving read and write requests from the file system to the clients  Figure 3.1 Various system roles in an HDFS deployment	[10]	CO2	L2

	• The design of HDFS is based on two types of nodes: a NameNode and multiple DataNodes. **i. HDFS Block Replication** When HDFS writes a file, it is replicated across the cluster. The amount of replication is based on the value of dfs.replication in the hdfs-site.xml file. This default value can be overruled with the hdfs dfs-setrep command. For Hadoop clusters containing more than eight DataNodes, the replication value is usually set to 3. **ii. HDFS Safe Mode** When the NameNode starts, it enters a read-only safe mode where blocks cannot be replicated or deleted. **iii. Rack Awareness** • Rack awareness deals with data locality. • The main design goals of Hadoop MapReduce is to move the computation to the data. Assuming that most data center networks do not offer full bisection bandwidth. **iv.** NameNode High Availability • 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. **v.** 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. **vi.** HDFS Checkpoints and Backups • The NameNode stores the metadata of the HDFS file system in a file called fsimage. • File systems modifications are written to an edits log file, and at startup the NameNode merges the edits into a new fsimage. • The Secondary NameNode or CheckpointNode periodically fetches edits from the NameNode, merges them, and returns an updated fsimage to the NameNode. **vii.** HDFS Snapshots HDFS snapshots are similar to backups, but are created by administrators using the hdfs dfs snapshot command. **viii.** HDFS NFS Gateway The HDFS NFS Gateway supports NFSv3 and enables HDFS to be mounted as part of the client's local file system. Users can browse the HDFS file system through their local file systems that provide an NFSv3 client compatible operating system.			
2 (a)	**Write short note on MapReduce Framework.** **Scheme:** Mapreduce Framework Explanation = 5M **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.	[05]	CO2	L2

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 sire 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
- 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:

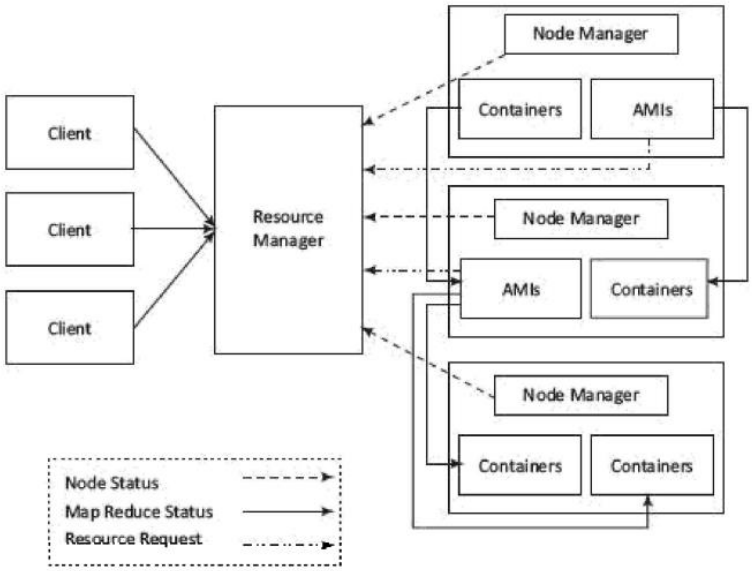
(i) estimate the need of resources for processing that request,
(ii) analyze the states of the slave nodes,
(iii) place the mapping tasks in queue,
(iv) monitor the progress of task, and on the failure, restart the task on slots of time available.

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.			
2 (b)	**Explain Hadoop YARN with diagram.** **Scheme:** YARN Diagram and Explanation = 2+3 = 5M **Solution:** 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. **Hadoop 2 Execution Model** 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	[05]	CO2	L2

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.



YARN-based execution model

3	**Explain in detail about NoSQL Data Architecture Patterns with the list of NoSQL databases.** **Scheme:** Five Patterns with explanation = 5*2= 10M **Solution:** NoSQL data stores broadly categorize into architectural patterns described in the following: • Key-Value Store • Document Store • Tabular Data • Object Data Store • Graph Database	[10]	CO3	L2
4	**Compare SQL and NoSQL with different features with detail explanation for all features.** **Scheme:** Features with explanation = 10M **Solution:**	[10]	CO3	L2

Table 3.7 Comparison of NoSQL with SQL/RDBMS

Features	NoSQL Data store	SQL/RDBMS
Model	Schema-less model	Relational
Schema	Dynamic schema	Predefined
Types of data architecture patterns	Key/value based, column-family based, document based, graph based, object based	Table based
Scalable	Horizontally scalable	Vertically scalable
Use of SQL	No	Yes
Dataset size preference	Prefers large datasets	Large dataset not preferred
Consistency	Variable	Strong
Vendor support	Open source	Strong
ACID properties	May not support, instead follows Brewer's CAP theorem or BASE properties	Strictly follows

5	**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 Ways of handling Big Data Problems	[10]	CO3	L2

6	**What are the features of MongoDB and explain the commands for the following operations in MongoDB.**
	Scheme: Features + Commands = 3+7 = 10M
	Solution:
	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 (Section 3.5.1.3), and (xi) fault tolerant. Document data store in JSON-like documents. The data store uses the dynamic schemas.
	• 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”
	},
	{
	“USN”:12389,
	”Name”:”Shahul”,
	”Section”:”C”
	}]
	)
	• **Display** all the documents from the collection.
	db.student.find()
	• **Remove** a document whose “USN”:12345.
	db.student.remove(“USN”:12345))
		[10]	CO3	L3

Faculty Signature CCI Signature HOD Signature