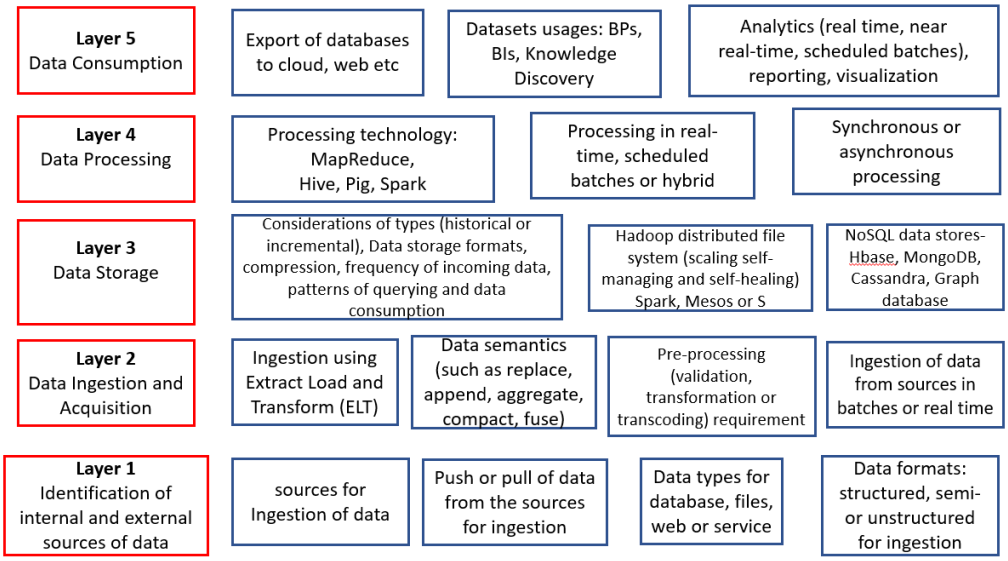
USN | | | | | | | | | | |

Internal Assessment Test 4 – February 2022

Sub:	BIG DATA AND ANALYTICS					Sub Code:	18CS72	Branch:	ISE
Date:	01/02/2022	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 designing data architecture with neat diagram.** **Scheme:** Five-layer explanation + Diagram 6+4 = 10M **Solution:**  Data processing architecture consists of five layers: (i) identification of data sources, (ii) acquisition, ingestion, extraction, pre-processing, transformation of data, (iii) data storage at files, servers, cluster or cloud, (iv) data-processing, and (v) data consumption	[10]	CO1	L2
2	**List out Applications of Big data and explain each application in detail.** **Scheme:** List of applications with explanation = 10M **Solution:** Many applications such as social network and social media, cloud applications, public and commercial web sites, scientific experiments, simulators and e-government services generate Big Data. Big Data analytics find applications in many areas. Some of the popular ones are marketing, sales, health care, medicines, advertising etc. Following subsections describe these use cases, applications and case studies. • Big Data in Marketing and Sales • Big Data Analytics in Detection of Marketing Frauds • Big Data Risks	[10]	CO1	L2

	• Big Data Credit Risk Management • Big Data and Algorithmic Trading • Big Data and Healthcare • Big Data in Medicine • Big Data in Advertising			
3	**Explain in detail about NoSQL data store with suitable diagram.** **Scheme:** NoSQL Data store explanation + diagram 6*4 = 10M **Solution: NoSQL** 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. 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. NoSQL data stores are considered as semi-structured data. Big Data Store uses NoSQL. NoSQL data store characteristics are as follows: 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. **Features in NoSQL Transactions** NoSQL transactions have following features: (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. Big Data NoSQL solutions use standalone-server, master-slave and peer-topeer distribution models. **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.	[10]	CO3	L2

Apache's HBase	HDFS compatible, open-source and non-relational data store written in Java; A column-family based NoSQL data store, data store providing BigTable-like capabilities (Sections 2.6 and 3.3.3.2); scalability, strong consistency, versioning, configuring and maintaining data store characteristics
Apache's MongoDB	HDFS compatible; master-slave distribution model (Section 3.5.1.3); document-oriented data store with JSON-like documents and dynamic schemas; open-source, NoSQL, scalable and non-relational database; used by Websites Craigslist, eBay, Foursquare at the backend
Apache's Cassandra	HDFS compatible DBs; decentralized distribution peer-to-peer model (Section 3.5.1.4); open source; NoSQL; scalable, non-relational, column-family based, fault-tolerant and tuneable consistency (Section 3.7) used by Facebook and Instagram
Apache's CouchDB	A project of Apache which is also widely used database for the web. CouchDB consists of Document Store. It uses the JSON data exchange format to store its documents, JavaScript for indexing, combining and transforming documents, and HTTP APIs
Oracle NoSQL	Step towards NoSQL data store; distributed key-value data store; provides transactional semantics for data manipulation, horizontal scalability, simple administration and monitoring
Riak	An open-source key-value store; high availability (using replication concept), fault tolerance, operational simplicity, scalability and written in Erlang

CAP Theorem Among C, A and P, two are at least present for the application/service/process.

- *Consistency* means all copies have the same value like in traditional DBs.
- *Availability* means at least one copy is available in case a partition becomes inactive or fails.
- *Partition* means parts which are active but may not cooperate (share) as in distributed DBs.

The CAP theorem implies that for a network partition system, the choice of consistency and availability are mutually exclusive. CA means consistency and availability; AP means availability and partition tolerance and CP means consistency and partition tolerance. Figure 3.1 shows the CAP theorem usage in Big Data.

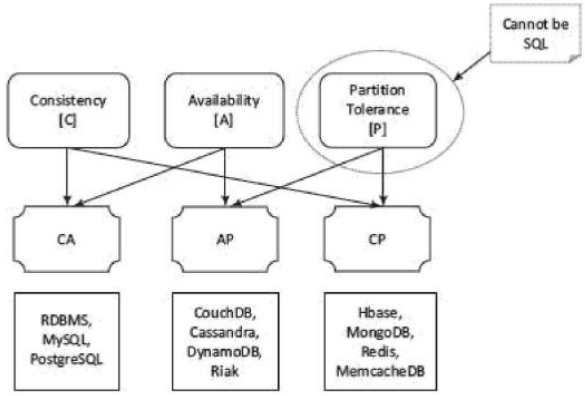


Figure 3.1 CAP theorem in Big Data solutions

Schema-less Models

Schema of a database system refers to designing of a structure for datasets and data structures for storing into the database. NoSQL data not necessarily have a fixed table schema. Data written at one node replicates to multiple nodes. Therefore, these are identical, fault-tolerant and partitioned into shards. Distributed databases can store and process a set of information on more than one computing nodes.

NoSQL data model offers relaxation in one or more of the ACID properties (Atomicity, consistence, isolation and durability) of the database. Distribution follows CAP theorem. CAP theorem states that out of the three properties, two must at least be present for the application/service/process. Data stores use non-mathematical relations but store this information as an aggregate called metadata.

Metadata refers to data describing and specifying an object or objects. Metadata is a record with all the information about a particular dataset and the inter-linkages. Metadata helps in selecting an object, specifications of the data and, usages that design where and when. Metadata specifies access permissions, attributes of the objects and enables additions of an attribute layer to the objects. Files, tables, documents and images are also the objects.

Increasing Flexibility for Data Manipulation

NoSQL data store characteristics are schema-less. The additional data may not be structured and follow fixed schema. The data store consists of additional data, such as documents, blogs, Facebook pages and tweets. NoSQL data store possess characteristic of increasing flexibility for data manipulation. BASE is a flexible model for NoSQL data stores. Provisions of BASE increase flexibility.

BASE Properties BA stands for basic availability, S stands for soft state and E stands for eventual consistency.

1. *Basic availability* ensures by distribution of shards (many partitions of huge data store) across many data nodes with a high degree of replication. Then, a segment failure does not necessarily mean a complete data store unavailability.
2. *Soft state* ensures processing even in the presence of inconsistencies but achieving consistency eventually. A program suitably takes into account the inconsistency found during processing. NoSQL database design does not consider the need of consistency all along the processing time.
3. *Eventual consistency* means consistency requirement in NoSQL databases meeting at some point of time in future. Data converges eventually to a consistent state with no time-frame specification for achieving that. ACID rules require consistency all along the processing on completion of each transaction. BASE does not have that requirement and has the flexibility.

BASE model is not necessarily appropriate in all cases but it is flexible and is an alternative to SQL-like adherence to ACID properties. Example 3.11 (Section 3.3.5) explains the concept of BASE in transactions using graph databases.

Schema is not a necessity in NoSQL DB, implying information storage flexibility. Data can store and retrieve without having knowledge of how a database stores and functions internally.

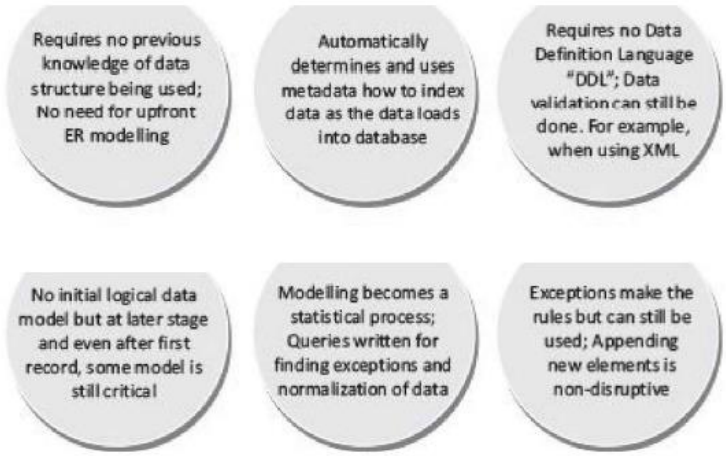


Figure 3.2 Characteristics of Schema-less model

4	**Write short note on Cassandra Databases and explain CQL commands and its functionalities with example.** **Scheme:** Cassandra database + CQL commands 6+4 = 10M **Solution:**	[10]	CO3	L2

Cassandra was developed by Facebook and released by Apache. Cassandra was named after Trojan mythological prophet Cassandra, who had classical allusions to a curse on oracle. Later on, IBM also released the enhancement of Cassandra, as open-source version. The open-source version includes an IBM Data Engine which processes No SQL data store. The engine has improved throughput when workload of read-operations is intensive.

Cassandra is basically a column family database that stores and handles massive data of any format including structured, semi-structured and unstructured data.

Apache Cassandra DBMS contains a set of programs. They *create* and *manage* databases. Cassandra provides functions (commands) for querying the data and accessing the required information. Functions do the viewing, querying and changing (update, insert or append or delete), visualizing and perform transactions on the DB.

Apache Cassandra has the distributed design of Dynamo. Cassandra is written in Java. Big organizations, such as Facebook, IBM, Twitter, Cisco, Rackspace, eBay, Twitter and Netflix have adopted Cassandra.

Characteristics of Cassandra are (i) open source, (ii) scalable (iii) non-relational (v) NoSQL (iv) Distributed (vi) column based, (vii) decentralized, (viii) fault tolerant and (ix) tuneable consistency.

Features of Cassandra are as follows:

1. Maximizes the number of writes - writes are not very costly (time consuming)
2. Maximizes data duplication
3. Does not support Joins, group by, OR clause and aggregations
4. Uses Classes consisting of ordered keys and semi-structured data storage systems
5. Is fast and easily scalable with write operations spread across the cluster. The cluster does not have a master-node, so any read and write can be handled by any node in the cluster.
6. Is a distributed DBMS designed for handling a high volume of structured data across multiple cloud servers
7. Has peer-to-peer distribution in the system across its nodes, and the data is distributed among all the nodes in a cluster (Section 3.5.1.4).

Data Replication Cassandra stores data on multiple nodes (data replication) and thus has no single point of failure, and ensures availability, a requirement in CAP theorem. Data replication uses a replication strategy. Replication factor determines the total number of replicas placed on different nodes. Cassandra returns the most recent value of the data to the client. If it has detected that some of the nodes responded with a stale value, Cassandra performs a read repair in the background to update the stale values.

Components at Cassandra Table 3.13 gives the components at Cassandra and their description.

Components	Description
Node	Place where data stores for processing
Data Center	Collection of many related nodes
Cluster	Collection of many data centers
Commit log	Used for crash recovery; each write operation written to commit log
Mem-table	Memory resident data structure, after data written in commit log, data write in mem-table temporarily
SSTable	When mem-table reaches a certain threshold, data flush into an SSTable disk file
Bloom filter	Fast and memory-efficient, probabilistic-data structure to find whether an element is present in a set, Bloom filters are accessed after every query.

Scalability Cassandra provides linear scalability which increases the throughput and decreases the response time on increase in the number of nodes at cluster.

Transaction Support Supports ACID properties (Atomicity, Consistency, Isolation, and Durability).

Replication Option Specifies any of the two replica placement strategy names. The strategy names are Simple Strategy or Network Topology Strategy. The replica placement strategies are:

1. Simple Strategy: Specifies simply a replication factor for the cluster.
2. Network Topology Strategy: Allows setting the replication factor for each data center independently.

Data Types Table 3.14 gives the data types built into Cassandra, their usage and descriptions

Table 3.14 Data types built into Cassandra, their usage and description

CQL Type	Description
ascii	US-ASCII character string
bigint	64-bit signed long integer
blob	Arbitrary bytes (no validation), BLOB expressed in hexadecimal
boolean	True or false
counter	Distributed counter value (64-bit long)
decimal	Variable-precision decimal integer, float
double	64-bit IEEE-754 *double precession* floating point integer, float
float	32-bit IEEE-754 *single precession* floating point integer, float
inet	IP address string in IPv4 or IPv6 format, used by the python-cql driver and CQL native protocols
int	32-bit signed integer

	list	A collection of one or more ordered elements			
	map	A JSON-style array of literals: {literal: literal, literal: literal ...}			
	set	A collection of one or more elements			
	text	UTF-8 encoded string			
	timestamp	Date plus time, encoded as 8 bytes since epoch integers, strings			
	varchar	UTF-8 encoded string			
	varint	Arbitrary-precision integer			

5	**Explain MapReduce Processing steps and write the java program for Map phase and Reduce phase of Word Count Problem.** **Scheme:** Processing steps + program 6+4 = 10M **Solution:**	[10]	CO4	L3

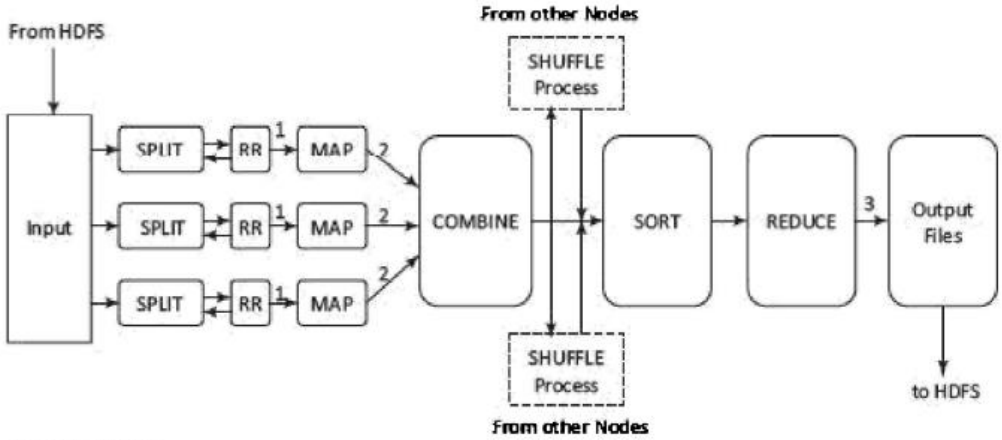


Figure 4.6 MapReduce execution steps

MapReduce programming model refers to a programming paradigm for processing Big Data sets with a parallel and distributed environment using map and reduce tasks.

```
public class WordCount {
  public static class TokenizerMapper
    extends Mapper<Object, Text, Text, IntWritable>
    {
      private final static IntWritable one = new IntWritable(1);
      private Text word = new Text();

      public void map(Object key, Text value, Context context)
        throws IO Exception, Interrupted Exception
        {
          String Tokenizer itr = new String Tokenizer(value.
            toString());
```

```
        while (itr.hasMoreTokens())
        {
            word.set(itr.nextToken());
            context.write(word, one);
        }
    }
}
public static class IntSumReducer
 extends Reducer <Text, IntWritable, Text, IntWritable>
{
  private IntWritable result = new IntWritable();
  public void reduce(Text key, Iterable<IntWritable>
    values, context context) throw IOException,
     InterruptedException
  {
     int sum=0;
     for (IntWritable val: Values)
     {
        sum+= val.get();
     }
     result.sem(sum);
     context.write(key, result);
  }
}
```

6	**How MapReduce can be used for Matrix Multiplication and explain its seven steps**. **Scheme:** Matrix Multiplication steps = 10M **Solution:** Consider matrices named A (i rows and j columns) and B (j rows and k columns) to produce the matrix C;(i rows and k columns). Consider the elements of matrices A, B and C as follows: $A = \begin{matrix} a_{11}\ a_{12} \ldots a_{1j} \\ a_{21}\ a_{22} \ldots a_{2j} \\ \ldots\ldots \\ \ldots\ldots \\ a_{i1}\ a_{i2} \ldots a_{ij} \end{matrix}$ $B = \begin{matrix} b_{11}\ b_{12} \ldots b_{1k} \\ b_{21}\ b_{22} \ldots b_{2k} \\ \ldots\ldots \\ \ldots\ldots \\ b_{j1}\ b_{i2} \ldots b_{jk} \end{matrix}$ $C = \begin{matrix} c_{11}\ c_{12} \ldots c_{1k} \\ c_{21}\ c_{22} \ldots c_{2k} \\ \ldots\ldots \\ \ldots\ldots \\ c_{i1}\ c_{i2} \ldots c_{ik} \end{matrix}$ ... (4.3)	[10]	CO4	L2

A.B = C; Each element evaluates as follow:

$$C_{ik} = \text{Sum } (a_{ij} \times b_{jk})_{j=1 \text{ to } j}.\ v_a = a_{ij} \text{ and } v_b = b_{jk} \quad \ldots (4.4)$$

First row of C

C first column element = $(a_{11}b_{11} + a_{12}b_{21} + \ldots + a_{1j}.b_{j1})$. Second column element = $(a_{11}b_{12} + a_{12}b_{22} + \ldots + a_{1j}.b_{j2})$.
The k^{th} column element = $(a_{11}b_{1k} + a_{12}b_{2k} + \ldots + a_{1j}\ b_{jk})$.

Second row of C

C first column element = $(a_{21}b_{11} + a_{22}b_{21} + \ldots + a_{2j}.b_{j1})$. Second column element = $(a_{21}b_{12} + a_{22}b_{22} + \ldots + a_{2j}.b_{j2})$.
The k^{th} column element = $(a_{21}b_{1k} + a_{22}b_{2k} + \ldots + a_{2j}.\ b_{jk})$.

The i^{th} row of C

C first column element = $(a_{i1}b_{11} + a_{i2}b_{21} + \ldots + a_{ij}.b_{j1})$. Second column element = $(a_{i1}b_{12} + ai_2b_{22} + \ldots + a_{ij}.b_{j2})$. The k^{th} column element = $(a_{i1}b_{1k} + a_{i2}b_{2k} + \ldots + a_{ij}.b_{jk})$.

Consider two solutions of matrix multiplication.

Table 4.2 Seven steps for multiplication of A and B for cascading of two MapReduce Steps

Step	Step description	Matrix A	Matrix B	Matrix C = A.B
1	Name	A	B	C
2	Specify attributes of (Key, Value pairs of each element [row number, column number, value]	(I, J, v_a)	(J, K, v_b)	(I, K, v_c)
3	Specify relations	R_A = A (I, J, v_a)	R_B = A (J, K, v_b)	R_C = A (I, K, v_c)
4	Consider tuples of A, B and C	(i, j, a_{ij})	(j, k, b_{jk}).	(i, k, c_{ik})
5	Find natural Join of R_A and R_B = Matrix elements (a_{ij}, b_{jk}) [j is common in both]	-	-	tuples (i, j, k, v_a, v_b)
6	Get tuples for finding Product C			Four-component tuple (i, j, k, $v_a \times v_b$)
7	Grouping and aggregation of tuples with attributes I and K			<I, K> SUM ($v_a \times v_b$).

Matrix Multiplication with One MapReduce Step MapReduce tasks for Steps 5 to 7 in a single step.

(e) Map Function: For each element a_{ij} of A, the *Mapper* emits all the key-value pairs
[(i, k), (A, j, a_{ij})] for k = 1, 2, . . ., up to the number of columns of B. Similarly, emits all the key-value pairs [(i, k), (B, j, b_{jk})] for i = 1, 2, . . ., up to the number of rows of A. for each element b_{jk} of B.

(f) Reduce Function: Consider the tuples of A = (A, i, a_{ij}) for each key j. Consider tuples of
B = (B, k, b_{jk}) for each key j. Emits the key-value pairs with key equal to (i, k) and value = sum of ($a_{ij} \times b_{jk}$) for all values j.

Memory required in one step MapReduce is large as compared to two steps in cascade. This is due to the need to store intermediate values of v_c and then sum them in the same *Reducer* step.

Faculty Signature CCI Signature HOD Signature