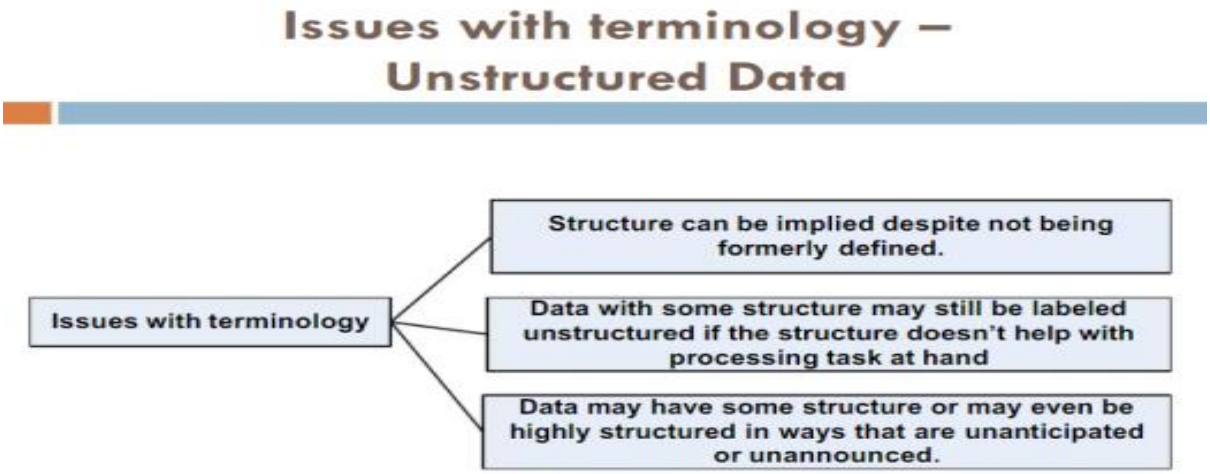
PE-I, BDA[CSE4059] _ENDSEM_Make-Up_Q&Answer_Scheme_May2023

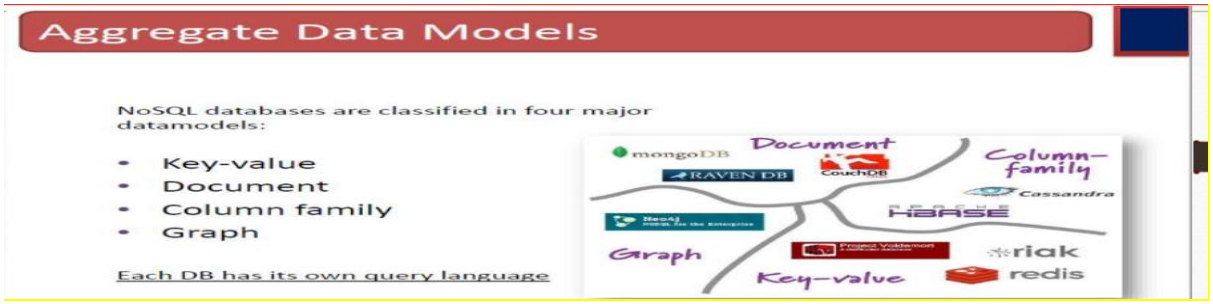
Q1. Discuss issue related with terminology of unstructured data. (3)

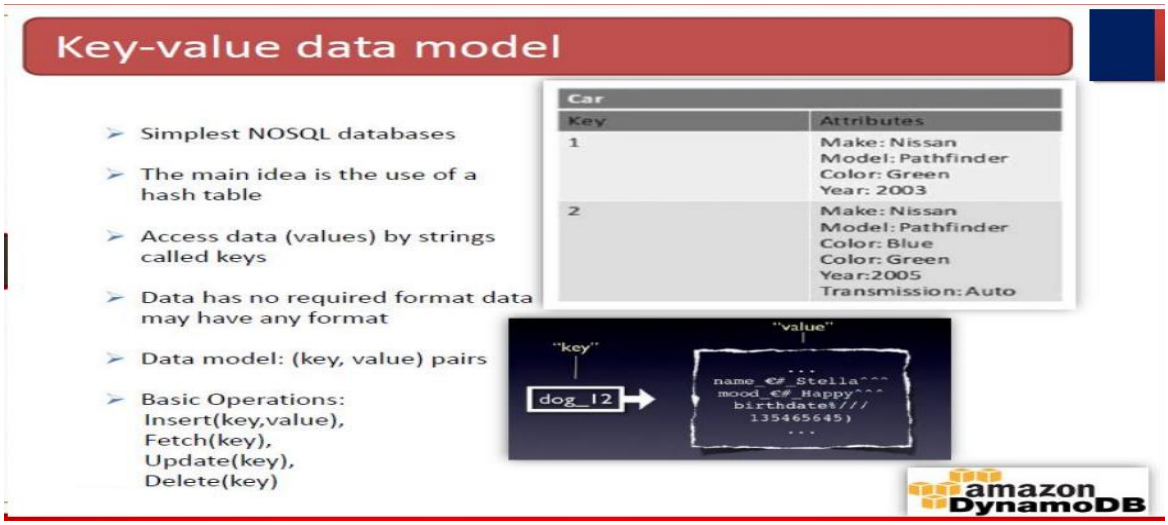
Ans:[1X3=M] .



Q2. Discuss on various classified data models of NoSQL database. (4)

Ans:[1X4=4]

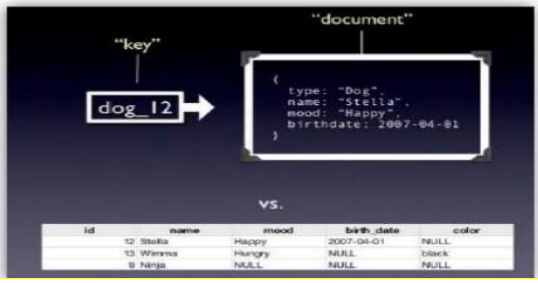


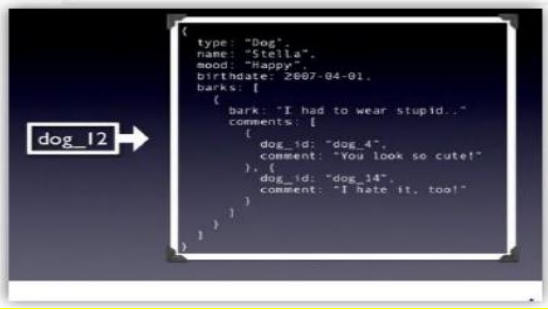


Document based data model

- Pair each key with complex data structure known as data structure.
- Indexes are done via B-Trees.
- Documents can contain many different key-value pairs, or key-array pairs, or even nested documents.

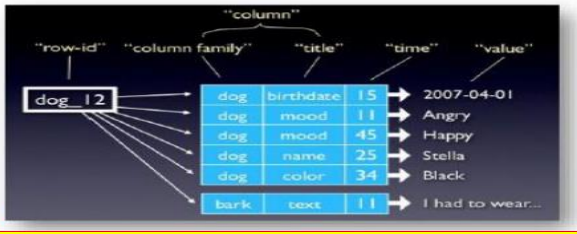
```
{
  person: {
    first_name: "Peter",
    last_name: "Peterson",
    addresses: [
      {street: "123 Peter St"},
      {street: "504 Not Peter St"}
    ],
  }
}
```





Column family data model

- The column is lowest/smallest instance of data.
- It is a tuple that contains a name, a value and a timestamp



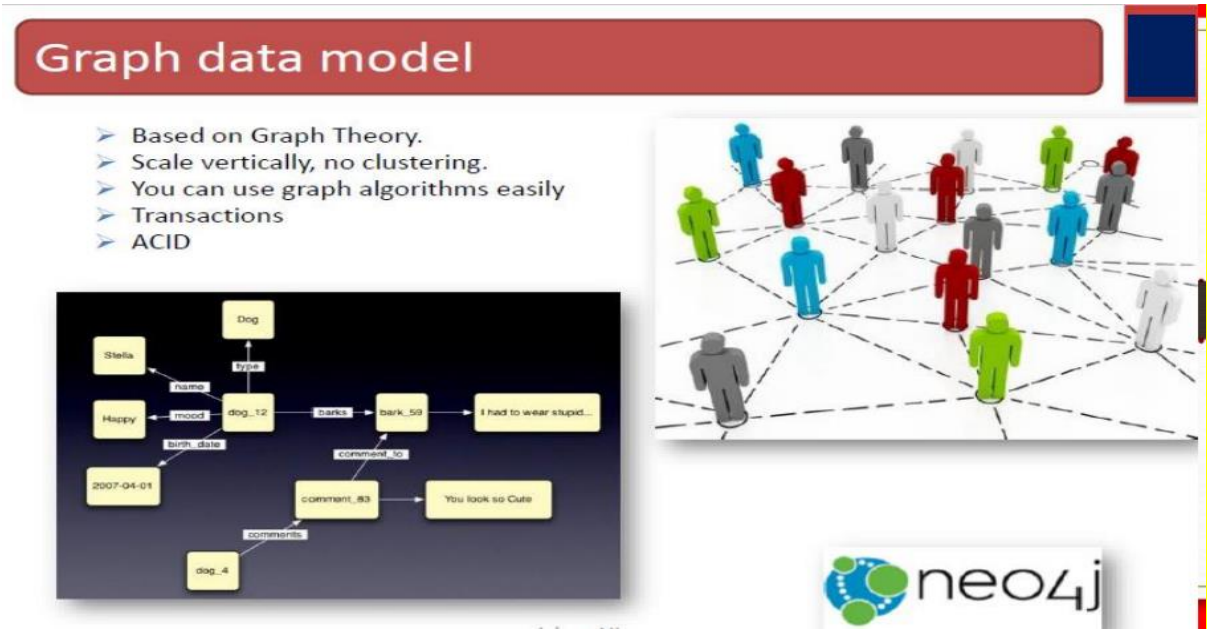
ColumnFamily: Authors

Key	Value
"Eric Long"	Columns: Name "email" → Value "eric (at) long.com"; "country" → "United Kingdom"; "registeredSince" → "01/01/2002"
"John Steward"	Columns: Name "email" → Value "john.steward (at) somedomain.com"; "country" → "Australia"; "registeredSince" → "01/01/2009"
"Ronald Mathies"	Columns: Name "email" → Value "ronald (at) sodeso.nl"; "country" → "Netherlands, The"; "registeredSince" → "01/01/2010"

Column family data model

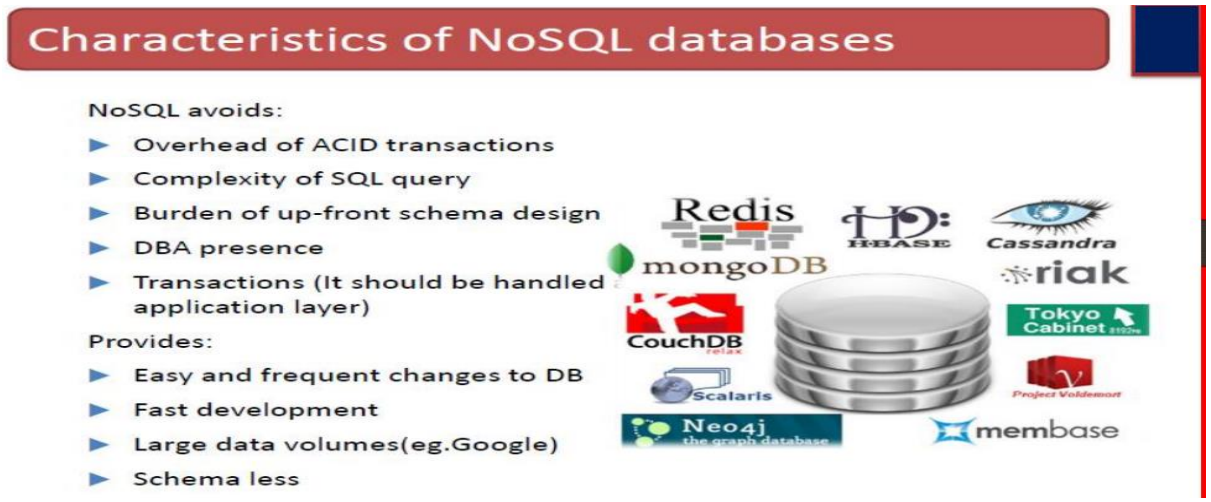
Some statistics about Facebook Search (using Cassandra)

- MySQL > 50 GB Data
  - Writes Average : ~300 ms
  - Reads Average : ~350 ms
- Rewritten with Cassandra > 50 GB Data
  - Writes Average : 0.12 ms
  - Reads Average : 15 ms



Q3. Differentiate NoSQL database characteristics with traditional database. (3)

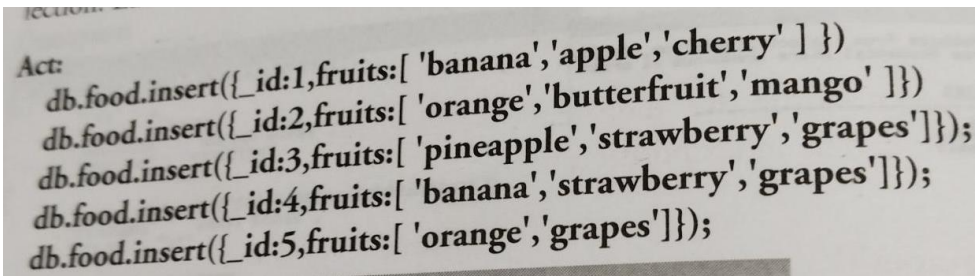
Ans:[Avoids =1.5,Provides=1.5]



Q4. Illustrate MongoDb query for the following: (5)

(i)To create a collection by the name "food" and then insert documents into the "food" collection (5 id's). Each document should have a "fruits" array.

Ans: [2+1+1+1=5]



```
Act:
db.food.insert({_id:1,fruits:[ 'banana','apple','cherry' ] })
db.food.insert({_id:2,fruits:[ 'orange','butterfruit','mango' ]})
db.food.insert({_id:3,fruits:[ 'pineapple','strawberry','grapes']});
db.food.insert({_id:4,fruits:[ 'banana','strawberry','grapes']});
db.food.insert({_id:5,fruits:[ 'orange','grapes']});
```

(ii) To find those documents from the "food" collection where grapes is present in the2nd index position of the" fruits" array.

Ans: db. food. Find({'fruits.2':'grapes'})

(iii) To find the document with (_id:1) from the "food" collection and display two elements from the array" fruits", starting with the element at 1st index position.

Ans: db. food. find({_id:1},{"fruits":{$slice:[1,2]}})

(iv) To find all documents from the "food" collection which have elements "orange" and "grapes" in the array "fruits".

Ans. db. food. find ({fruits: {$all: ["orange", "grapes"]}}). pretty ();

Q5. Discuss on features of Cassandra. (3)

Ans .[0.5 X 6=3]

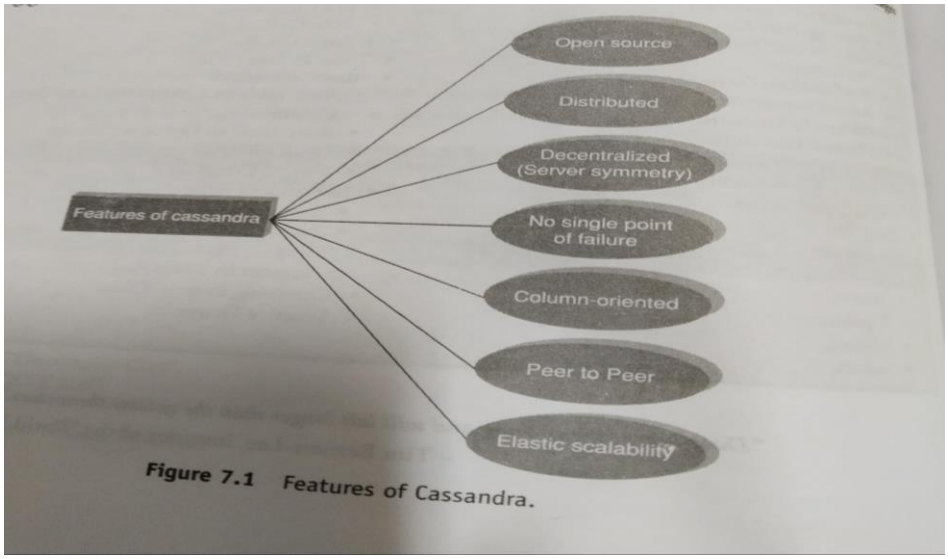
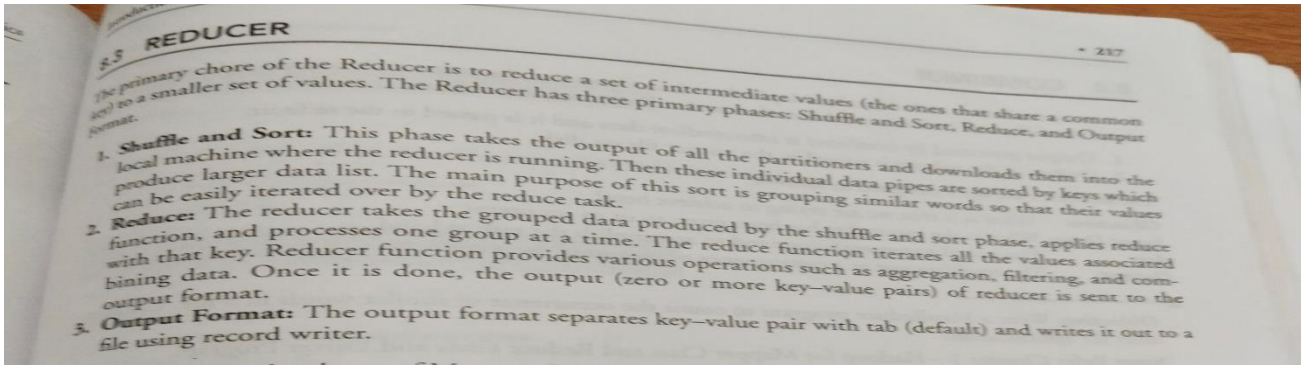


Figure 7.1 Features of Cassandra.

Q6. Discuss functional component of reducer class. (2)

Ans. [3points=2m]



8.5 REDUCER

The primary chore of the Reducer is to reduce a set of intermediate values (the ones that share a common key) to a smaller set of values. The Reducer has three primary phases: Shuffle and Sort, Reduce, and Output format.

1. **Shuffle and Sort:** This phase takes the output of all the partitioners and downloads them into the local machine where the reducer is running. Then these individual data pipes are sorted by keys which produce larger data list. The main purpose of this sort is grouping similar words so that their values can be easily iterated over by the reduce task.
2. **Reduce:** The reducer takes the grouped data produced by the shuffle and sort phase, applies reduce function, and processes one group at a time. The reduce function iterates all the values associated with that key. Reducer function provides various operations such as aggregation, filtering, and combining data. Once it is done, the output (zero or more key–value pairs) of reducer is sent to the output format.
3. **Output Format:** The output format separates key–value pair with tab (default) and writes it out to a file using record writer.

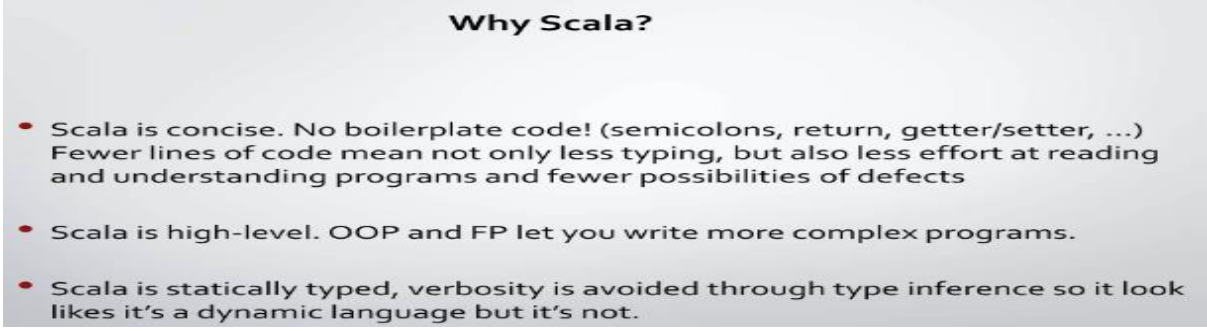
Q7. Discuss on Recommendation systems. (2)

Ans: [1+1]

Recommender systems are machine learning systems that help users discover new products and services.
Every time you shop online, a recommendation system is guiding you towards the most likely product you might purchase. A recommendation system is a subset of machine learning that uses data to help users find products and content.
Websites and streaming services use recommender systems to generate "for you" or "you might also like" pages and content. Recommender systems are an essential feature in our digital world, as users are often overwhelmed by choice and need help finding what they're looking for. This leads to happier customers and, of course, more sales. Recommender systems are like salesmen who know, based on your history and preferences, what you like.

Q8. Discuss on requirements of scala. (3)

Ans .[1X3=3]



Q9. Discuss on solutions for following problems related to distributed hardware performance woes. (5)

P1: You want to determine if a job runs slowly due to hardware problems, and you want to see if nodes are blacklisted or graylisted.
Solution: Use the JobTracker UI to check for nodes that are graylisted or blacklisted.

P2: You want to determine if a job runs slowly due to CPU overutilization.
Solution: Use the Linux tool vmstat to observe the CPU context switches.

P3: You want to determine if a job runs slowly due to swapping.
Solution: Use the Linux tool vmstat to observe if memory is being swapped in and out of disk.

P4: You want to understand if a drive runs in degraded or read-only mode.
Solution: The Linux tool iostat can be used to look at drive request queues and IO wait times. Other Linux tools such as dmesg can help determine if a drive has gone into a readonly mode.

P5: You want to determine if a job runs slowly due to network issues.
Solution: Examining the output of the Linux tools ethtool and sar can help diagnose network mis-configurations.

Q10.Discuss on Hadoop Limitations. (2)

Ans.[0.5 X4 =2]

Common areas identified as weaknesses across HDFS and MapReduce include availability and security. All of their master processes are single points of failure, although you should note that there's active work on High Availability versions in the community. Security is another area that has its wrinkles, and again another area that's receiving focus.

HIGH AVAILABILITY

Until the Hadoop 2.x release, HDFS and MapReduce employed single-master models, resulting in single points of failure.11 The Hadoop 2.x version will eventually bring both Name Node and Job Tracker High Availability (HA) support. The 2.x Name Node HA design requires shared storage for NameNode metadata, which may require expensive HA storage. It supports a single standby NameNode, preferably on a separate rack.

SECURITY

Hadoop does offer a security model, but by default it's disabled. With the security model disabled, the only security feature that exists in Hadoop is HDFS file and directory-level ownership and permissions. But it's easy for malicious users to subvert and assume other users' identities. By default, all other Hadoop services are wide open, allowing any user to perform any kind of operation, such as killing another user's

MapReduce jobs.

Hadoop can be configured to run with Kerberos, a network authentication protocol, which requires Hadoop daemons to authenticate clients, both user and other Hadoop components. Kerberos can be integrated with an organization's existing. Active Directory, and therefore offers a single sign-on experience for users. Finally, and most important for the government sector, there's no storage or wire-level encryption in Hadoop. Overall, configuring Hadoop to be secure has a high pain point due
to its complexity.

Q11.Discuss on high level of Hadoop architecture. (4)

Ans.[block Diagram=1.5+ Theory =2.5]

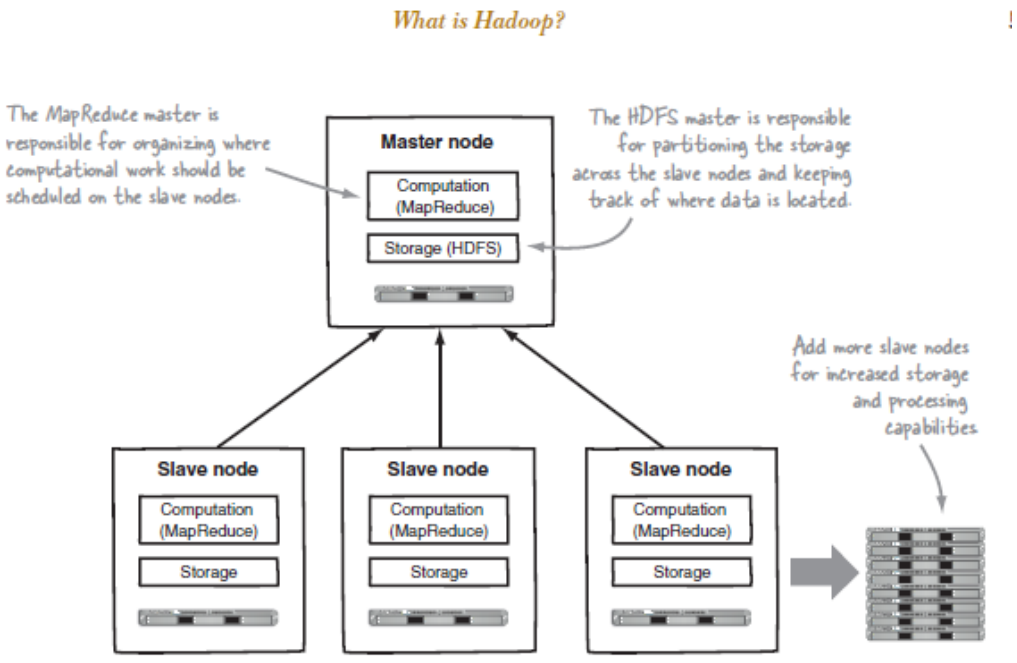
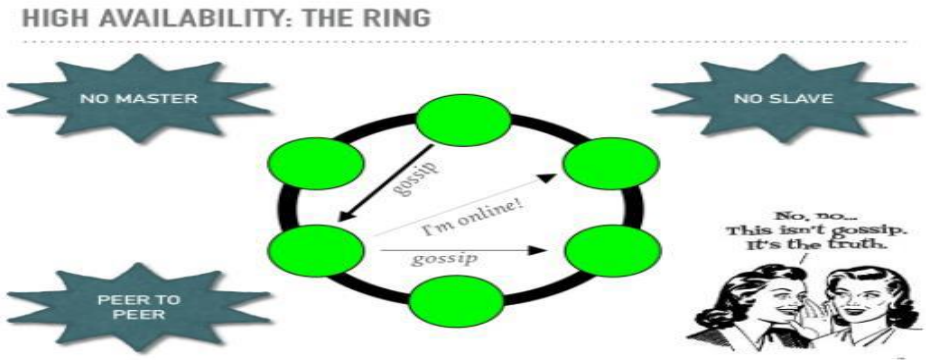


Figure 1.2 High-level Hadoop architecture

Q12. Discuss Gossip Protocol with an example. (4)

Ans: [2+2]

Network Communication protocols inspired for real life rumor spreading.
Periodic, Pairwise, inter node communication.
Low frequency communication ensures low cost. Random selection of peers.
Example Node A wish to search for pattern in data
Round 1 Node A searches locally and then gossips with node B.
Round 2 Node A,B gossips with C and D.
Round 3 Nodes A,B,C and D gossips with 4 other nodes Round by round doubling makes protocol veryrobust.



Variety of Gossip Protocols exists
Dissemination protocol
Event Dissemination: multicasts events via gossip. high latency might cause network strain.
Background data dissemination: continuous gossip about information regarding participating nodes

Anti-Entropy protocol: Used to repair replicated data by comparing and reconciling differences. This type of protocol is used in Cassandra to repair

Q13. Discuss on Spark SQL Architecture. (3)

Ans:[Block diagram 1+ theory=2]

This architecture contains three layers namely, Language API, Schema RDD, and Data Sources.

- **Language API** − Spark is compatible with different languages and Spark SQL. It is also, supported by these languages- API (python, Scala, java, HiveQL).
- **Schema RDD** − Spark Core is designed with special data structure called RDD. Generally, Spark SQL works on schemas, tables, and records. Therefore, we can use the Schema RDD as temporary table. We can call this Schema RDD as Data Frame.
- **Data Sources** − Usually the Data source for spark-core is a text file, Avro file, etc. However, the Data Sources for Spark SQL is different. Those are Parquet file, JSON document, HIVE tables, and Cassandra database
- A Data Frame is a distributed collection of data, which is organized into named columns. Conceptually, it is equivalent to relational tables with good optimization techniques.
- A Data Frame can be constructed from an array of different sources such as Hive tables, Structured Data files, external databases, or existing RDDs. This API was designed for modern Big Data and data science applications taking inspiration from

Data Frame in **R Programming** and Pandas in **Python**

Core Hadoop components

To understand Hadoop's architecture, we'll start by looking at the basics of HDFS.

HDFS

HDFS is the storage component of Hadoop. It's a distributed filesystem that's modeled after the Google File System (GFS) paper.4 HDFS is optimized for high throughput and works best when reading and writing large files (gigabytes and larger). To support this throughput HDFS leverages unusually large (for a filesystem) block sizes and data locality optimizations to reduce network input/output (I/O).

Scalability and availability are also key traits of HDFS, achieved in part due to data replication

and fault tolerance. HDFS replicates files for a configured number of times, is tolerant of both software and hardware failure, and automatically re-replicates data blocks on nodes that have failed.

Figure 1.3 shows a logical representation of the components in HDFS: the Name-Node and the Data Node. It also shows an application that's using the Hadoop filesystem library to access HDFS.

Now that you have a bit of HDFS knowledge, it's time to look at MapReduce, Hadoop's computation engine.

MAPREDUCE

MapReduce is a batch-based, distributed computing framework modeled after Google's paper on MapReduce.5 It allows you to parallelize work over a large amount of

Q14. Discuss the advantages and drawback of anomaly detection. (2)

Ans : [Advantages=1 + drawback=1]

Clustering Based Anomaly Detection

- **Advantages:**
  - No need to be supervised
  - Easily adaptable to on-line anomaly detection from temporal data
- **Drawbacks**
  - Computationally expensive – Time complexity is $O(cn)$, c - # of clusters
    - Using indexing structures (k-d tree, R* tree) may alleviate this problem
  - If normal points do not create any clusters the techniques may fail
  - In high dimensional spaces, data is sparse and distances between any two data records may become quite similar.
    - Clustering algorithms may not give any meaningful clusters

Q15. Illustrate a program to demonstrates how to load a LIBSVM data file, parse it as an RDD of LabeledPoint and then perform regression using a decision tree with variance as an impurity measure and a maximum tree depth of 5. The Mean Squared Error (MSE) is computed at the end to evaluate goodness of fit. (5)
Ans. [Program Steps =1 X 5=5]

```
import org.apache.spark.mllib.tree.DecisionTree
import org.apache.spark.mllib.tree.model.DecisionTreeModel
import org.apache.spark.mllib.util.MLUtils

// Load and parse the data file.
val data = MLUtils.loadLibSVMFile(sc, "data/mllib/sample_libsvm_data.txt")
// Split the data into training and test sets (30% held out for testing)
val splits = data.randomSplit(Array(0.7, 0.3))
val (trainingData, testData) = (splits(0), splits(1))

// Train a DecisionTree model.
//  Empty categoricalFeaturesInfo indicates all features are continuous.
val categoricalFeaturesInfo = Map[Int, Int]()
val impurity = "variance"
val maxDepth = 5
val maxBins = 32

val model = DecisionTree.trainRegressor(trainingData, categoricalFeaturesInfo, impurity,
  maxDepth, maxBins)

// Evaluate model on test instances and compute test error
val labelsAndPredictions = testData.map { point =>
  val prediction = model.predict(point.features)
  (point.label, prediction)
}
val testMSE = labelsAndPredictions.map{ case (v, p) => math.pow(v - p, 2) }.mean()
println(s"Test Mean Squared Error = $testMSE")
println(s"Learned regression tree model:\n ${model.toDebugString}")

// Save and load model
model.save(sc, "target/tmp/myDecisionTreeRegressionModel")
val sameModel = DecisionTreeModel.load(sc, "target/tmp/myDecisionTreeRegressionModel")
```