Unit 3 BIB

1) Discuss the design of Hadoop distributed file system and concept in detail.

Hadoop File System was developed using distributed file system design. It is run on commodity hardware. Unlike other distributed systems, HDFS is highly fault-tolerant and designed using low-cost hardware.

HDFS holds very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines. These files are stored in redundant fashion to rescue the system from possible data losses in case of failure. HDFS also makes applications available to parallel processing.

Features of HDFS

 It is suitable for the distributed storage and processing.

 Hadoop provides a command interface to interact with HDFS.

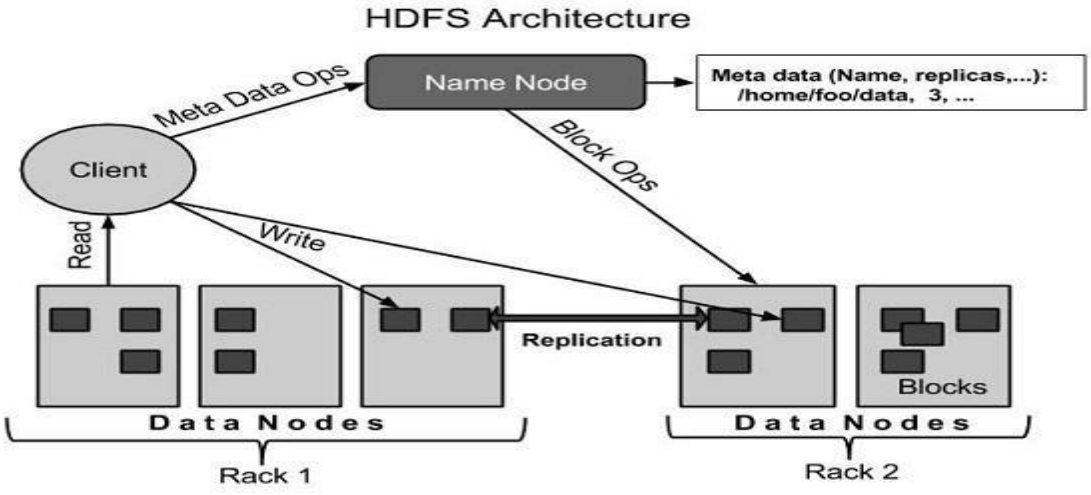
 The built-in servers of namenode and datanode help users to easily check the status of cluster.

 Streaming access to file system data.

 HDFS provides file permissions and authentication.

HDFS Architecture

Given below is the architecture of a Hadoop File System.



HDFS follows the master-slave architecture and it has the following elements.

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 system having the namenode acts as the master server and 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.

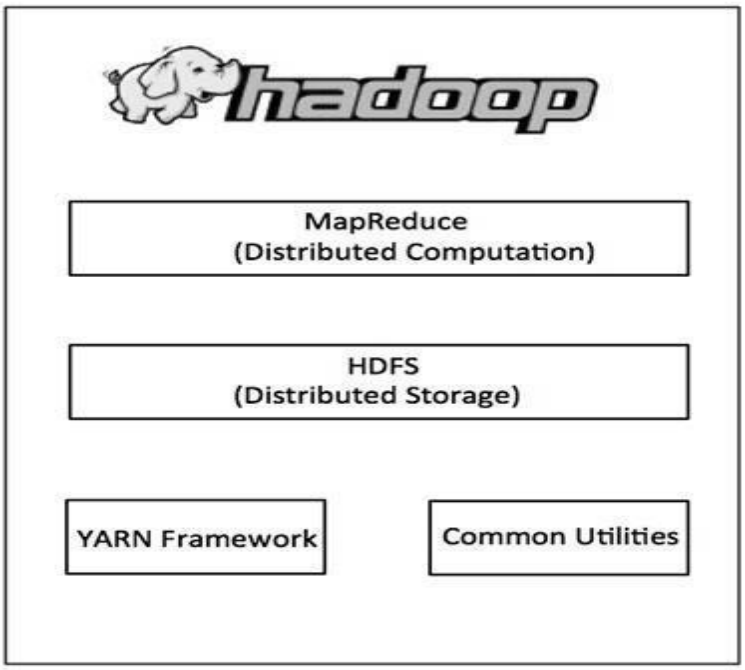
Block
Generally the user data is stored in the files of HDFS. The file in a file system will be divided into one or more segments and/or stored in individual data nodes. These file segments are called as blocks. In other words, the minimum amount of data that HDFS can read or write is called a *Block.* The default block size is 64MB, but it can be increased as per the need to change in HDFS configuration.

2) Discuss the architecture of Hadoop and its modules in Details.

Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. The Hadoop framework application works in an environment that provides distributed *storage* and *computation* across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

Hadoop Architecture
At its core, Hadoop has two major layers namely:
(a) Processing/Computation layer (MapReduce), and

(b) Storage layer (Hadoop Distributed File System).



MapReduce

MapReduce is a parallel programming model for writing distributed applications devised at Google for efficient processing of large amounts of data (multi-terabyte data-sets), on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. The MapReduce program runs on Hadoop which is an Apache open-source framework.

The term MapReduce actually refers to the following two different tasks that Hadoop programs perform:

- **The Map Task:** This is the first task, which takes input data and converts it into a set of data, where individual elements are broken down into tuples (key/value pairs).
- **The Reduce Task:** This task takes the output from a map task as input and combines those data tuples into a smaller set of tuples. The reduce task is always performed after the map task.

Typically both the input and the output are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

The MapReduce framework consists of a single master **JobTracker** and one slave **TaskTracker** per cluster-node. The master is responsible for resource management, tracking resource consumption/availability and scheduling the jobs component tasks on the

slaves, monitoring them and re-executing the failed tasks. The slaves TaskTracker execute the tasks as directed by the master and provide task-status information to the master periodically.

The JobTracker is a single point of failure for the Hadoop MapReduce service which means if JobTracker goes down, all running jobs are halted.

Hadoop Distributed File System

The Hadoop Distributed File System (HDFS) is based on the Google File System (GFS) and provides a distributed file system that is designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. It is highly fault-tolerant and is designed to be deployed on low-cost hardware. It provides high throughput access to application data and is suitable for applications having large datasets.

Apart from the above-mentioned two core components, Hadoop framework also includes the following two modules:

 Hadoop Common: These are Java libraries and utilities required by other Hadoop modules.

Apache Foundation has pre-defined set of utilities and libraries that can be used by other modules within the Hadoop ecosystem. For example, if HBase and Hive want to access HDFS they need to make of Java archives (JAR files) that are stored in Hadoop Common.

 Hadoop YARN: This is a framework for job scheduling and cluster resource management.

YARN forms an integral part of Hadoop 2.0.YARN is great enabler for dynamic resource utilization on Hadoop framework as users can run various Hadoop applications without having to bother about increasing workloads.

Yet Another Resource Negotiator (YARN).

YARN is a cluster management technology. It is one of the key features in second-generation Hadoop.

It is the next-generation MapReduce, which assigns CPU, memory and storage to applications running on a Hadoop cluster. It enables application frameworks other than MapReduce to run on Hadoop, opening up a wealth of possibilities.

Part of the core Hadoop project, YARN is the architectural center of Hadoop that allows multiple data processing engines such as interactive SQL, real-time streaming, data science and batch processing to handle data stored in a single platform.

3) How does Hadoop works?

It is quite expensive to build bigger servers with heavy configurations that handle large scale processing, but as an alternative, you can tie together many commodity computers with single-CPU, as a single functional distributed system and practically, the clustered machines can read the dataset in parallel and provide a much higher throughput. Moreover, it is cheaper than one high-end server. So this is the first motivational factor behind using Hadoop that it runs across clustered and low-cost machines.
Hadoop runs code across a cluster of computers. This process includes the following core tasks that Hadoop performs:

 Data is initially divided into directories and files. Files are divided into uniform sized blocks of 128M and 64M (preferably 128M).

 These files are then distributed across various cluster nodes for further processing.

 HDFS, being on top of the local file system, supervises the processing.

 Blocks are replicated for handling hardware failure.

 Checking that the code was executed successfully.

4) What are the advantages of Hadoop?

 Hadoop framework allows the user to quickly write and test distributed systems. It is efficient, and it automatic distributes the data and work across the machines and in turn, utilizes the underlying parallelism of the CPU cores.

 Hadoop does not rely on hardware to provide fault-tolerance and high availability (FTHA), rather Hadoop library itself has been designed to detect and handle failures at the application layer.

 Servers can be added or removed from the cluster dynamically and Hadoop continues to operate without interruption.

 Another big advantage of Hadoop is that apart from being open source, it is compatible on all the platforms since it is Java based.

5) Explain Features of Hadoop.

1. Hadoop Brings Flexibility In Data Processing:

One of the biggest challenges organizations have had in that past was the challenge of handling unstructured data. Let's face it, only 20% of data in any organization is structured while the rest is all unstructured whose value has been largely ignored due to lack of technology to analyze it.

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

This is a huge feature of Hadoop. It is an open source platform and runs on industry-standard 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.

The level of replication is configurable and this makes Hadoop incredibly reliable data storage system. This means, even if a node gets lost or goes out of service, the system automatically reallocates work to another location of the data and continues processing as if nothing had happened!

4. Hadoop Is Great At Faster Data Processing

While traditional ETL and batch processes can take hours, days, or even weeks to load large amounts of data, the need to analyze that data in real-time is becoming critical day after day.

Hadoop is extremely good at high-volume batch processing because of its ability to do parallel 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 has a very robust ecosystem that is well suited to meet the analytical needs of developers and small to large organizations. Hadoop Ecosystem comes with a suite of tools and technologies making i a 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. and many new tools and technologies are being added to the ecosystem as the market grows.

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.

Apache Hadoop was developed to help Internet-based companies deal with prodigious volumes of data. According to some analysts, the cost of a Hadoop data management system, including hardware, software, and other expenses, comes to about $1,000 a terabyte–about one-fifth to one-twentieth the cost of other data management technologies

6) Explain Map-Reduce Framework in Details.

MAPREDUCE

MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner.

What is MapReduce?

MapReduce is a parallel programming model for writing distributed applications devised at Google for efficient processing of large amounts of data (multi-terabyte data-sets), on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. The MapReduce program runs on Hadoop which is an Apache open-source framework.

The term MapReduce actually refers to the following two different tasks that Hadoop programs perform:

- **The Map Task:** This is the first task, which takes input data and converts it into a set of data, where individual elements are broken down into tuples (key/value pairs).

- **The Reduce Task:** This task takes the output from a map task as input and combines those data tuples into a smaller set of tuples. The reduce task is always performed after the map task.

Typically both the input and the output are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

The MapReduce framework consists of a single master **JobTracker** and one slave **TaskTracker** per cluster-node. The master is responsible for resource management, tracking resource consumption/availability and scheduling the jobs component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves TaskTracker execute the tasks as directed by the master and provide task-status information to the master periodically.

The JobTracker is a single point of failure for the Hadoop MapReduce service which means if JobTracker goes down, all running jobs are halted.

The Algorithm

 Generally MapReduce paradigm is based on sending the computer to where the data resides!

 MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.

o **Map stage**: The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.

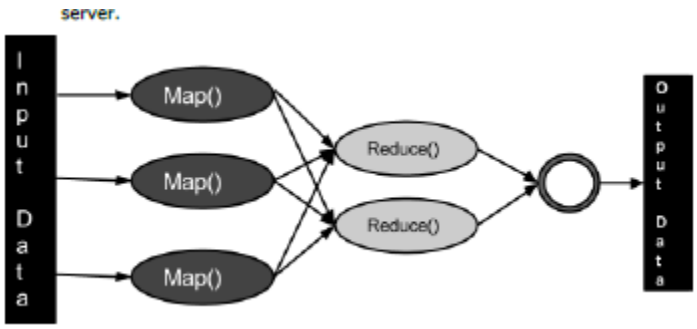
o **Reduce stage**: This stage is the combination of the **Shuffle** stage and the **Reduce** stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

 During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.

 The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.

 Most of the computing takes place on nodes with data on local disks that reduces the network traffic.

 After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.



Terminology

 PayLoad - Applications implement the Map and the Reduce functions, and form the core of the job.

 Mapper - Mapper maps the input key/value pairs to a set of intermediate key/value pair.

 NamedNode - Node that manages the Hadoop Distributed File System (HDFS).

 DataNode - Node where data is presented in advance before any processing takes place.

 MasterNode - Node where JobTracker runs and which accepts job requests from clients.

 SlaveNode - Node where Map and Reduce program runs.

 JobTracker - Schedules jobs and tracks the assign jobs to Task tracker.

 Task Tracker - Tracks the task and reports status to JobTracker.

 Job - A program is an execution of a Mapper and Reducer across a dataset.

 Task - An execution of a Mapper or a Reducer on a slice of data.

 Task Attempt - A particular instance of an attempt to execute a task on a SlaveNode.

Unit 4

1) Distinguish Supervised Vs. Unsupervised Learning.

Supervised Learning

- > we are given a data set and already know what our correct output should look like, having the idea that there is a relationship between the input and the output.

->Supervised learning problems are categorized into **"regression"** and **"classification"** problems.

->In a **regression** problem, we are trying to predict results within a **continuous**output, meaning that we are trying to map input variables to some **continuous** function. In a **classification** problem, we are instead trying to predict results in a **discrete** output. In other words, we are trying to map input variables into **discrete** categories.

Unsupervised Learning

->Unsupervised learning, on the other hand, allows us to approach problems with little or no idea what our results should look like.

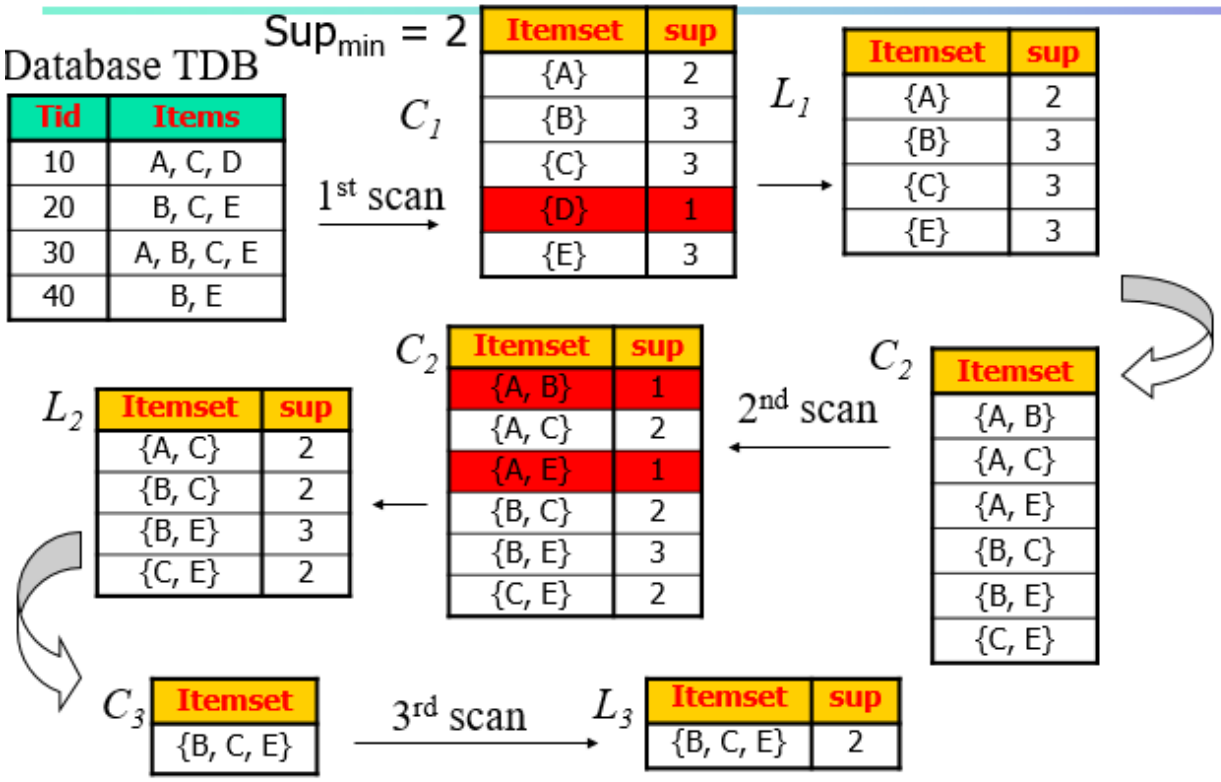
->We can derive this structure by **clustering** the data based on relationships among the variables in the data.

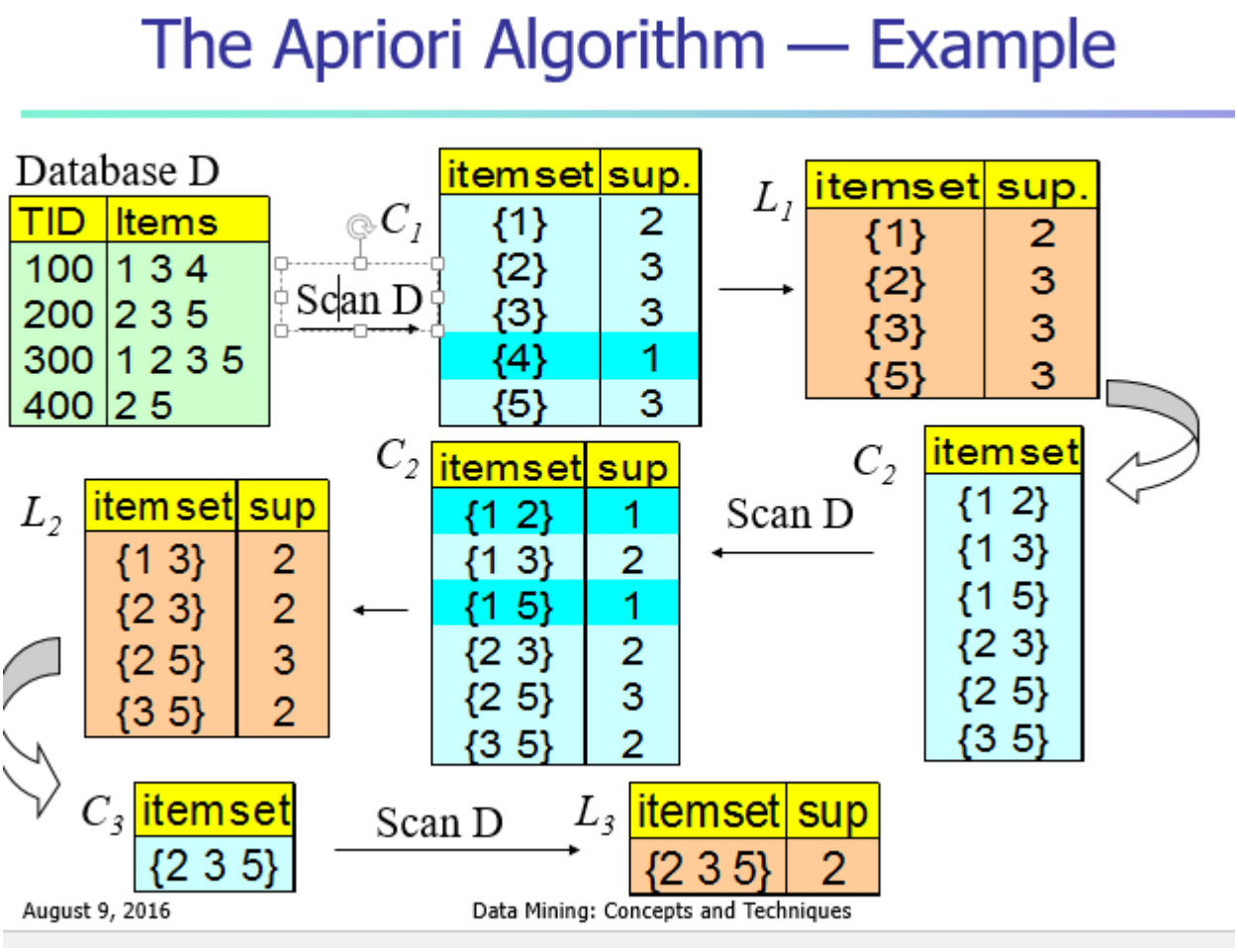
->With unsupervised learning there is no feedback based on the prediction results, i.e., there is no teacher to correct you. It's not just about clustering. For example, associative memory is unsupervised learning.

2. Explain the steps of the "Apriori Algorithm" for mining Frequent Item set.

- **Method:**
  - **Initially, scan DB once to get frequent 1-itemset**
  - **Generate length (k+1) candidate itemsets from length k frequent itemsets**
  - **Test the candidates against DB**
  - **Terminate when no frequent or candidate set can be generated**

The Apriori Algorithm—An Example





3. Write the steps of the K-means Clustering Algorithm. Also state its limitation.

The *K-Means* Clustering Method

- **Given *k*, the *k-means* algorithm is implemented in four steps:**
  - **Partition objects into *k* nonempty subsets**
  - **Compute seed points as the centroids of the clusters of the current partition (the centroid is the center, i.e., *mean point*, of the cluster)**
  - **Assign each object to the cluster with the nearest seed point**
  - **Go back to Step 2, stop when no more new assignment**

OR

Method: The k-means algorithm is implemented as follows.

1) arbitrarily choose k objects as the initial cluster centers;
2) repeat
3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
4) update the cluster means, i.e., calculate the mean value of the objects for each cluster;
5) until no change;

Figure 8.1: k-means algorithm.

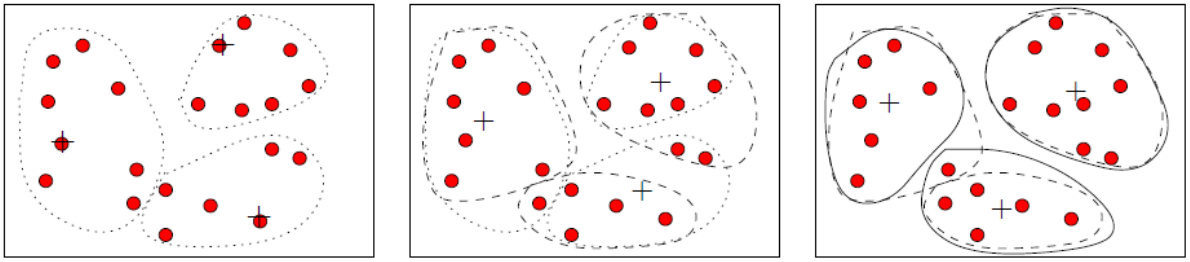


Figure 8.2: Clustering of a set of points based on the k-means method.

- **2 4 10 12 3 20 30 11 25 23**
- **Enter initial mean 1:2**
- **Enter initial mean 2:16**
- **Cluster 1:2 4 3**
- **m1=3**
- **Cluster 2:10 12 20 30 11 25 23**
- **m2=18**
- **----**
- **Cluster 1:2 4 10 3**
- **m1=4**
- **Cluster 2:12 20 30 11 25 23**
- **m2=20**
- **----**
- **Cluster 1:2 4 10 3 11**
- **m1=6**
- **Cluster 2:12 20 30 25 23**
- **m2=22**
- **----**
- **Cluster 1:2 4 10 12 3 11**
- **m1=7**
- **Cluster 2:20 30 25 23**
- **m2=24**
- **----**
- **Cluster 1:2 4 10 12 3 11**

- **m1=7**
- **Cluster 2:20 30 25 23**
- **m2=24**
- **----**

Limitation

- Applicable only when *mean* is defined, then what about categorical data?
- Need to specify *k*, the *number* of clusters, in advance
- Unable to handle noisy data and *outliers*
- Not suitable to discover clusters with *non-convex shapes*

4. Write a Short note on Hierarchical Clustering.

5. Explain how the topology of a neural network is designed.

Defining a Network Topology

- **First decide the network topology: # of units in the *input layer*, # of *hidden layers* (if > 1), # of units in *each hidden layer*, and # of units in the *output layer***
- **Normalizing the input values for each attribute measured in the training tuples to [0.0—1.0]**
- **One input unit per domain value, each initialized to 0**
- **Output, if for classification and more than two classes, one output unit per class is used**
- **Once a network has been trained and its accuracy is unacceptable, repeat the training process with a *different network topology* or a *different set of initial weights***

Backpropagation

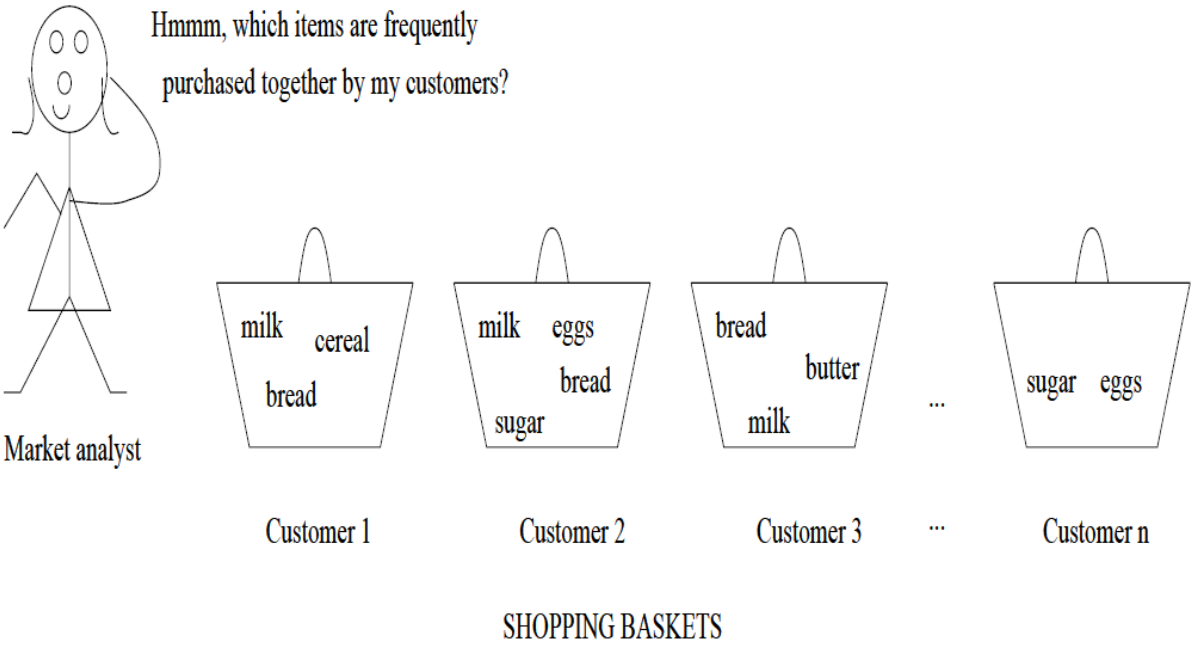
- **Iteratively process a set of training tuples & compare the network's prediction with the actual known target value**
- **For each training tuple, the weights are modified to minimize the mean squared error between the network's prediction and the actual target value**
- **Modifications are made in the "backwards" direction: from the output layer, through each hidden layer down to the first hidden layer, hence "backpropagation"**
- **Steps**
  - **Initialize weights (to small random #s) and biases in the network**
  - **Propagate the inputs forward (by applying activation function)**
  - **Backpropagate the error (by updating weights and biases)**
  - **Terminating condition (when error is very small, etc.)**

7. Write & Explain Apriori algorithm for discovering frequent item sets for mining Boolean Association Rules.

8. What is Market Basket Analysis? Explain Association Rules with Confidence & Support.

6.1.1 Market basket analysis: A motivating example for association rule mining

Suppose, as manager of an *AllElectronics* branch, you would like to learn more about the buying habits of your customers. Specifically, you wonder *"Which groups or sets of items are customers likely to purchase on a given trip to the store?"*. To answer your question, market basket analysis may be performed on the retail data of customer transactions at your store. The results may be used to plan marketing or advertising strategies, as well as catalog design. For instance, market basket analysis may help managers design different store layouts. In one strategy, items that are frequently purchased together can be placed in close proximity in order to further encourage the sale of such items together. If customers who purchase computers also tend to buy financial management software at the same time, then placing the hardware display close to the software display may help to increase the sales of both of these



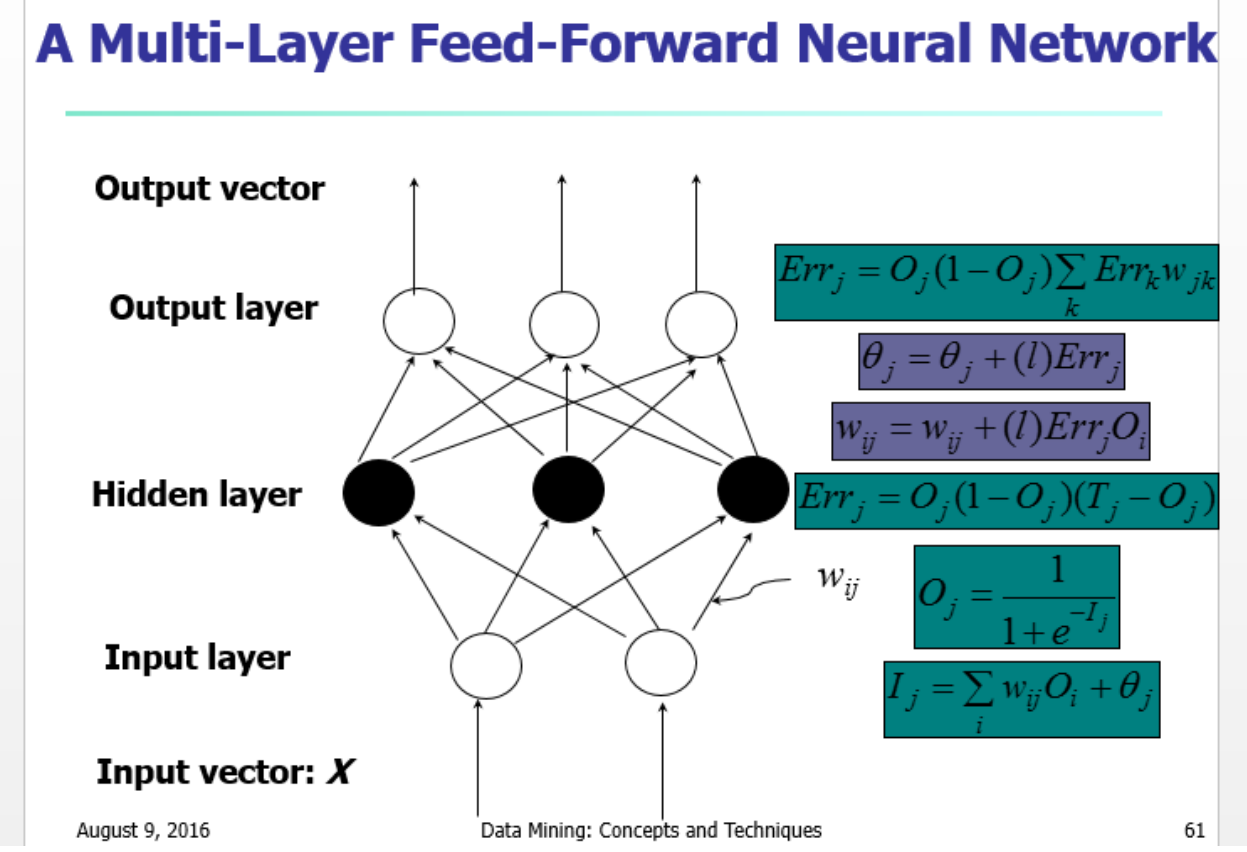
items. In an alternative strategy, placing hardware and software at opposite ends of the store may entice customers who purchase such items to pick up other items along the way. For instance, after deciding on an expensive computer, a customer may observe security systems for sale while heading towards the software display to purchase financial management software, and may decide to purchase a home security system as well. Market basket analysis can also help retailers to plan which items to put on sale at reduced prices. If customers tend to purchase computers and printers together, then having a sale on printers may encourage the sale of printers *as well as* computers.

If we think of the universe as the set of items available at the store, then each item has a Boolean variable representing the presence or absence of that item. Each basket can then be represented by a Boolean vector of values assigned to these variable. The Boolean vectors can be analyzed for buying patterns which reflect items that are frequent *associated* or purchased together. These patterns can be represented in the form of **association rules**. For example, the information that customers who purchase computers also tend to buy financial management software at the same time is represented in association Rule (6.1) below.

$$computer \Rightarrow financial\_management\_software \qquad [support = 2\%, confidence = 60\%] \qquad (6.1)$$

Rule **support** and **confidence** are two measures of rule interestingness that were described earlier in Section 1.5. They respectively reflect the usefulness and certainty of discovered rules. A support of 2% for association Rule (6.1) means that 2% of all the transactions under analysis show that computer and financial management software are purchased together. A confidence of 60% means that 60% of the customers who purchased a computer also bought the software. Typically, association rules are considered interesting if they satisfy both a **minimum support threshold** and a **minimum confidence threshold**. Such thresholds can be set by users or domain experts.

9. Explain Multilayer feed-forward neural network with example.



How A Multi-Layer Neural Network Works?

- The **inputs** to the network correspond to the attributes measured for each training tuple
- Inputs are fed simultaneously into the units making up the **input layer**
- They are then weighted and fed simultaneously to a **hidden layer**
- The number of hidden layers is arbitrary, although usually only one
- The weighted outputs of the last hidden layer are input to units making up the **output layer**, which emits the network's prediction
- The network is **feed-forward** in that none of the weights cycles back to an input unit or to an output unit of a previous layer
- From a statistical point of view, networks perform **nonlinear regression**: Given enough hidden units and enough training samples, they can closely approximate any function

6) Differentiate Agglomerative Vs. Divisive.

Agglomerative Clustering:

(Leaves to trunk)

- We start out with all sample units in *n* clusters of size 1.
- Then, at each step of the algorithm, the pair of clusters with the shortest distance are combined into a single cluster.
- The algorithm stops when all sample units are combined into a single cluster of size *n*.

Divisive Clustering:

(Trunk to leaves)

- We start out with **all** sample units in a single cluster of size *n*.
- Then, at each step of the algorithm, clusters are partitioned into a pair of daughter clusters, selected to maximize the distance between each daughter.
- The algorithm stops when sample units are partitioned into *n* clusters of size 1.

UNIT I (Introduction to Big Data)

1. What is Bigdata? and discuss in detail why big data is more important with real time examples?

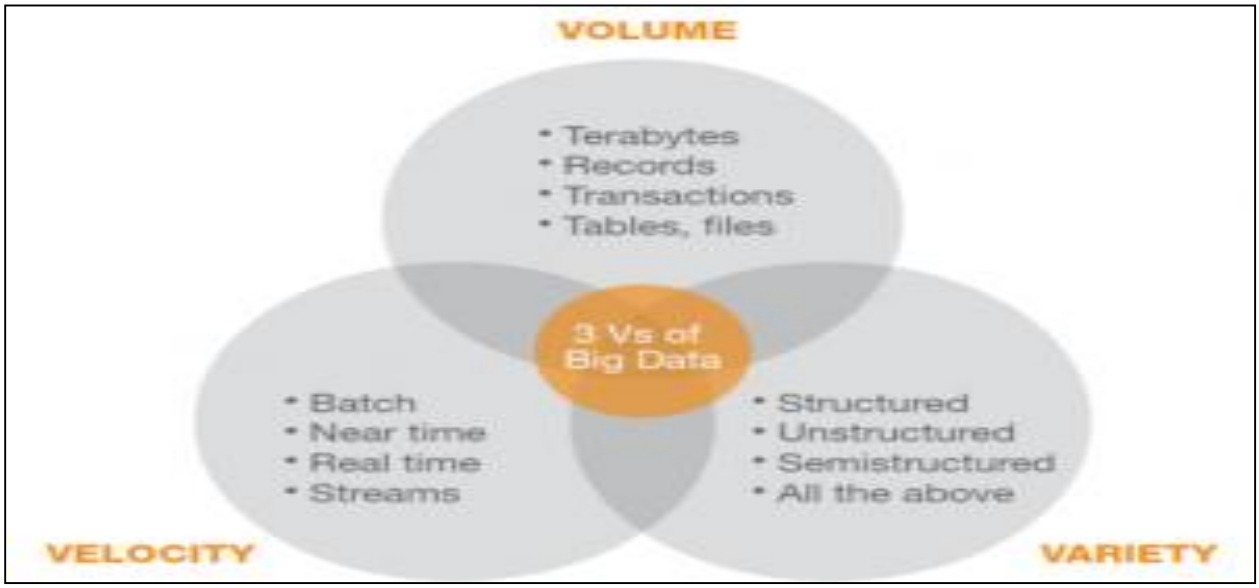
Big data is large data set. It is refered to as data set whose size is beyond the ability of typical database software tools to capture, store, manage and analyze.

Real time example Stoke market Social Media

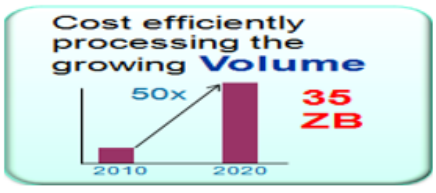
2. Discuss Bigdata in terms of three dimensions: volume, variety and velocity.

3 dimensions / characteristics of Big data

- 3Vs (volume, variety and velocity) are three defining properties or dimensions of big data.
- **Volume refers to the amount of data,**
- **variety refers to the number of types of data and**
- **velocity refers to the speed of data processing.**

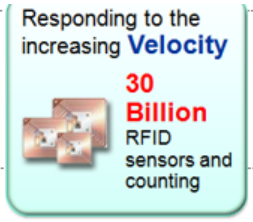


- **Volume:**
  The size of available data has been growing at an increasing rate.



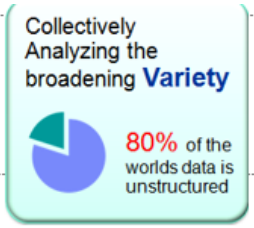
A text file is a few kilo bytes, a sound file is a few mega bytes while a full length movie is a few giga bytes. More sources of data are added on continuous basis. For companies, in the old days, all data was generated internally by employees. Currently, the data is generated by employees, partners and customers.
Example : Smart phones

Click to add title



- **Velocity:**
- Data is increasingly accelerating the velocity at which it is created and at which it is integrated. **We have moved from batch to a real-time business**
- Initially, companies analyzed data using a batch process. One takes a chunk of data, submits a job to the server and waits for delivery of the result. That scheme works when the incoming data rate is slower than the batch-processing rate and when the result is useful despite the delay. With the new sources of data such as social and mobile applications, the batch process breaks down. The data is now streaming into the server in real time, in a continuous fashion and the result is only useful if the delay is very short.

Click to add title



- **Variety:**
- Variety presents an equally difficult challenge. The growth in data sources has fuelled the growth in data types. In fact, 80% of the world's data is unstructured. Yet most traditional methods apply analytics only to structured information.
- The variety of data sources continues to increase. It includes
- ■ Internet data (i.e., click stream, social media, social networking links)
- ■ Primary research (i.e., surveys, experiments, observations)
- ■ Secondary research (i.e., competitive and marketplace data, industry reports, consumer data, business data)
- ■ Location data (i.e., mobile device data, geospatial data)
- ■ Image data (i.e., video, satellite image, surveillance)
- ■ Supply chain data (i.e., EDI, vendor catalogs and pricing, quality information)
- ■ Device data (i.e., sensors, PLCs, RF devices, LIMs, telemetry)

3. Discuss Industry Examples of Bigdata in detail

4. Discuss big data in healthcare and medicine.

5. Discuss Nuances of Big data.

6. Explain Nuts and Bolts of Big Data in details.

- **Assembling a Big Data solution is sort of like putting together an Collection set. There are various pieces and elements that must be put together in the proper fashion to make sure everything works adequately, and there are almost endless combinations of configurations that can be made with the components at hand.**
- **With Big Data, the components include platform pieces, servers, virtualization solutions, storage arrays, applications, sensors, and routing equipment. The right pieces must be picked and integrated in a fashion that offers the best performance, high efficiency, affordability, ease of management and use, and scalability.**

THE STORAGE DILEMMA

- Big Data consists of data sets that are too large to be acquired, handled, analyzed, or stored in an appropriate time frame using the traditional infrastructures.
- design new storage platforms that incorporate block- and file-based systems to meet the needs of Big Data and associated analytics. Meeting the challenges posed by Big Data means focusing on some key storage ideologies and understanding how those storage design elements interact with Big Data demands.

Capacity

- Big Data can mean petabytes of data. Big Data storage systems must therefore be able to quickly and easily change scale to meet the growth of data collections. These storage systems will need to add capacity in modules or arrays that are transparent to users, without taking systems down.
- Most Big Data environments are turning to scale-out storage (the ability to increase storage performance as capacity increases) technologies to meet that criterion.

- Big Data storage systems to expand file counts into the billions without suffering the overhead problems that traditional file systems encounter**.**

Security

- Many types of data carry security standards that are driven by compliance laws and regulations. The data may be financial, medical, or government intelligence and may be part of an analytics set yet still be protected.

Latency

- Latency produces "stale" data. That is another case in which scale-out architectures solve problems.
- The technology enables the cluster of storage nodes to increase in processing power and connectivity as they grow incapacity. Object-based storage systems can parallel data streams, further improving output.
- **Virtualization of server resources,**

  which is a common methodology used to expand compute resources without the purchase of new hardware, drives high IOPS requirements, just as it does in traditional IT environments. Those high IOPS performance requirements can be met with solid-state storage devices, which can be implemented in many different formats, including simple server-based cache to all-flash-based scalable storage systems.

Access

- Storage infrastructures that include global file systems can address this issue, since they allow multiple users on multiple hosts to access files from many different back-end storage systems in multiple locations.

Flexibility

- Big Data storage infrastructures also need to account for data migration challenges, at least during the start-up phase. Ideally, data migration will become something that is no longer needed in the world of Big Data, simply because the data are distributed in multiple locations.

Persistence

- Big Data applications often involve regulatory compliance requirements, which dictate that data must be saved for years or decades.

- Examples are medical information, which is often saved for the life of the patient, and financial information, which is typically saved for seven years. However, Big Data users are often saving data longer because they are part of a historical record or are used for time-based analysis**.**

Cost

- Big Data can be expensive. Given the scale at which many organizations are operating their Big Data environments, cost containment is imperative. That means more efficiency as well as less expensive components.
- **Storage duplication**

  has already entered the primary storage market and, depending on the data types involved, could bring some value for Big Data storage systems.
- Other Big Data storage technologies that can improve efficiencies are thin provisioning, snapshots, and cloning.
- **Thin provisioning** operates by allocating disk storage space in a flexible manner among multiple users based on the minimum space required by each user at any given time.
- **Snapshots** streamline access to stored data and can speed up the process of data recover.
- **Disk cloning** is copying the contents of a computer's hard drive.

8. Explain Features of Big Data.

- **Support for batch and real-time analytics.**
- Most of the existing platforms for processing data were designed for handling **transactional Web applications and have little support for business analytics applications**. That situation has driven Hadoop to become the de facto standard for handling batch processing. However, real-time analytics is altogether different, requiring something more than Hadoop can offer.

Alternative approaches.

- Transforming Big Data application development into something more mainstream may be the best way to leverage what is offered by Big Data. This means creating a built-in stack that integrates with Big Data databases from the **No SQL world** and creating **Map Reduce frameworks such as Hadoop and distributed processing.**

Available Big Data mapping tools.

- Batch-processing projects are being serviced with frameworks such as **Hive, which provide an SQL-like facade for handling complex batch processing with Hadoop.** However, other tools are starting to show promise. An example is **JPA, which provides a more standardized JEE abstraction that fits into real-time Big Data applications.** The **Google app Engine uses Data Nucleus along with Big table to achieve the same goal**.

Big Data abstraction tools.

- There are several choices available to abstract data, ranging from open source tools to commercial distributions of specialized products.
- One to pay attention to is **Spring Data from Spring Source, which is a high-level abstraction tool that offers the ability to map different data stores of all kinds into one common abstraction through annotation and a plug-in approach.**

Moving away from SQL

- Here Big Data platforms must be able to support schema-less semantics, which in turn means that the data mapping layer would need to be extended to support document semantics. **Examples are MongoDB, CouchBase, Cassandra, and the Giga Spaces document API.**

In-memory processing

- If the goal is to **deliver the best performance and reduce latency**, then one must consider using **RAM-based devices and perform processing in-memory.**
- Big Data platforms need to provide a **seamless integration between RAM and disk-based devices** in which data that are written in RAM would be synched into the disk asynchronously.

Built-in support for event-driven data distribution

- Big Data applications (and platforms) must also be able to work with event-driven processes.

- With Big Data, this means there must be data awareness incorporated, which makes it easy to route messages based on data affinity and the content of the message.

Support for public, private, and hybrid clouds

- Big Data applications consume large amounts of computer and storage resources. This has led to the use of the cloud and its elastic capabilities for running Big Data applications, which in turn can offer a more economical approach to processing Big Data jobs.

Consistent management

- The typical **Big Data application stack incorporates several layers, including the database itself, the Web tier, the processing tier, caching layer, the data synchronization and distribution layer, and reporting tools.** A major disadvantage for those managing Big Data applications is that each of those layers comes with different management, provisioning, monitoring, and troubleshooting tools.

9. Discuss the Security, Compliance, Auditing and Protection with respect to Big Data.

- How can the data be protected?
- **access:** Data can be easily protected, but only if you **eliminate access to the data.** That's not a pragmatic solution, to say the least. The key is to control access, but even then, knowing the ***who, what, when***, and ***where*** of data access is only a start.
- **availability**: controlling where the data are stored and how the data are distributed.
- **Performance :** Higher levels of **encryption, complex security methodologies, and additional security layers** can all improve security.
- **Liability :** Accessible data carry with them liability, such as the **sensitivity of the data, the legal requirements connected to the data, privacy issues, and intellectual property concerns.**

PRAGMATIC STEPS TO SECURING BIGDATA

- If you do not need certain information, it should be destroyed, because it represents a risk to the organization. That risk grows every day for as long as the information is kept.
- **Health care probably provides the best example for those charged with compliance as they examine how Big Data creation, storage, and flow work in their organizations.**
- In the medical industry, the primary problem is that **unsecured Big Data stores are filled with content that is collected and analyzed** in real time and is often extraordinarily sensitive: **intellectual property, personal identifying information, and other confidential information.** The disclosure of this type of data, by either attack or human error, can be disturbing to a company and its reputation.

- **Companies may also move Big Data to the cloud for disaster recovery, replication, load balancing, storage, and other purposes.**
  A combination of technologies has been assembled to meet four important goals

Control access by process, not job function.

- Server and network administrators, cloud administrators, and other employees often have access to more information than their jobs require because the systems simply lack the appropriate access controls. Just because a user has operating system–level access to a specific server does not mean that he or she needs, or should have, access to the Big Data stored on that server.

Secure the data at rest

- All Big Data, especially **sensitive information, should remain encrypted**, whether it is stored on a disk, on a server, or in the cloud

Protect the cryptographic keys and store them separately from the data

- Cryptographic keys are the **gateway to the encrypted data**. If the keys are left unprotected, the data are easily compromised.
- Storing the cryptographic keys on a separate, hardened server, either on the premises or in **the cloud, is the best practice for keeping data safe and an important step in regulatory compliance.** The bottom line is to treat key security with as much, if not greater, rigor than the data set itself.

Create trusted applications and stacks to protect data from rogue users

- You may **encrypt your data to control access,**
- Encrypting more than just the data and hardening the security of your overall environment—including applications, services, and configurations—gives you peace of mind that your sensitive information is protected from malicious users and rogue employees.
- **Automation** offers the ability to **decrease compliance and security costs** and still provide the higher levels of assurance, which validates where the data are and where they are going.

basic rules that should be used to enable security

- **Ensure that security does not impede performance or availability.**
- Big Data is all about handling volume while providing results, being able to deal with the velocity and variety of data , and allowing organizations to capture, analyze, store, or move data in real time. Security controls that limit any of these processes are a non starter for organizations serious about Big Data.
- **Pick the right encryption scheme.**
- Some data security solutions **encrypt at the file level or lower** , such as including specific data values, documents, or rows and columns.
- To maintain the high levels of performance required to analyze Big Data, consider a **transparent data encryption solution** optimized for Big Data.

Ensure that the security solution can evolve with your changing requirements

- The flexibility **to migrate between cloud providers and models based** on changing business needs is a requirement, and this is no different with Big Data technologies.
- When evaluating security, you should consider a solution that is **platform-agnostic** and can work with any Big Data file system or database, including Hadoop, Cassandra, and Mongo DB.

10. Explain the Challenges of Big Data.

- Capturing data
- Curation
- Storage
- Searching
- Sharing
- Transfer
- Analysis
- Presentation