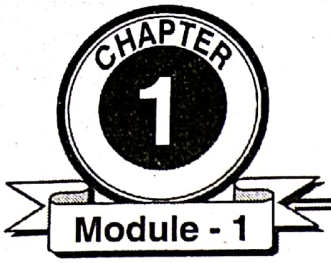

Introduction to Big Data

Syllabus

Introduction to Big Data, Big Data characteristics, Types of Big Data, Traditional vs. Big Data business approach, Case Study of Big Data Solutions

1.1 Introduction to Big Data Management

- We all are surrounded by huge data. People upload/download videos, audios, images from variety of devices. Sending text messages, multimedia messages, updating their Facebook, Whatsapp, Twitter status, comments, online shopping, online advertising etc. generates huge data.
- As a result of multiple processing machines have to generate and keep huge data too. Due to this exponential growth of data, Data analysis becomes very much required task for day to day operations.
- The term 'Big Data' means huge volume, high velocity and a variety of data.
- This big data is increasing tremendously day by day.
- Traditional data management systems and existing tools are facing difficulties to process such a Big Data.
- R is one of the main computing tools used in statistical education and research. It is also widely used for data analysis and numerical computing in other fields of scientific research.

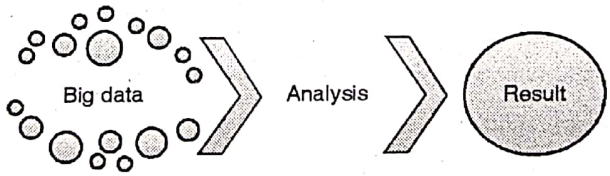

Fig. 1.1.1

1.2 Big Data

MU - May 17

Q What is Big Data ? **(May 17, 2 Marks)**
Q. Write a short note on Big Data.

- We all are surrounded by huge data. People upload/download videos, audios, images from variety of devices.
- Sending text messages, multimedia messages, updating their Facebook, Whatsapp, Twitter status, comments, online shopping, online advertising etc.
- Big data generates huge amount data.
- As a result machines have to generate and keep huge data too. Due to this exponential growth of data the analysis of that data becomes challenging and difficult.

- The term 'Big Data' means huge volume, high velocity and a variety of data. This big data is increasing tremendously day by day. Traditional data management systems and existing tools are facing difficulties to process such a Big Data.
- Big data is the most important technologies in modern world. It is really critical to store and manage it. Big is a collection of large datasets that cannot be processed using traditional computing techniques.
- Big Data includes huge volume, high velocity and extensible variety of data. The data in it may be structured data, Semi Structured data or unstructured data. Big data also involves various tools, techniques and frameworks.

1.3 Big Data Characteristics - Four Important V of Big Data

MU - May 16, Dec. 16, Dec. 17

Q. What are the three Vs of big data? **(May 16, Dec. 16, 2 Marks)**

Q. Describe any five characteristics of Big Data. **(Dec. 17, 5 Marks)**

Big data characteristics are as follows :

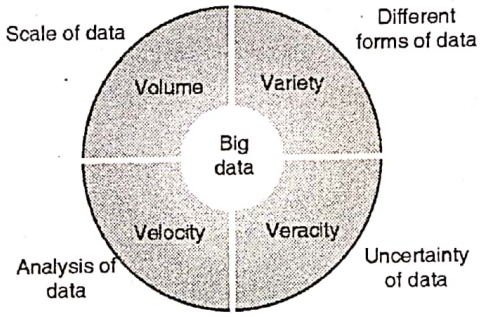

Fig. 1.3.1 : Big data characteristics

1. Volume

- Huge amount of data is generated during big data applications.
- The amount of data generated as well as storage volume is very big in size.

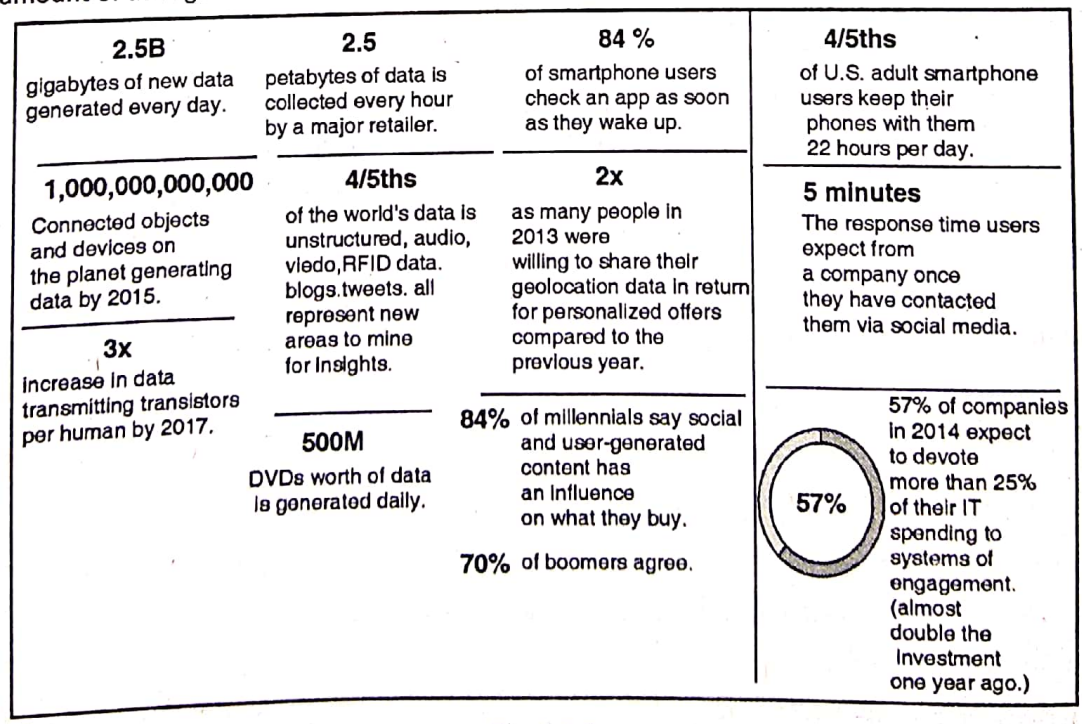

Fig. 1.3.2

2. Velocity

- For time critical applications the faster processing is very important. E.g. share marketing, video streaming
- The huge amount of data is generated and stored requires higher processing speed of processing data.
- The amount of digital data will be doubled in every 18 months and it repeats may be in less time in future.

3. Variety

The type and nature of data is having great variety.

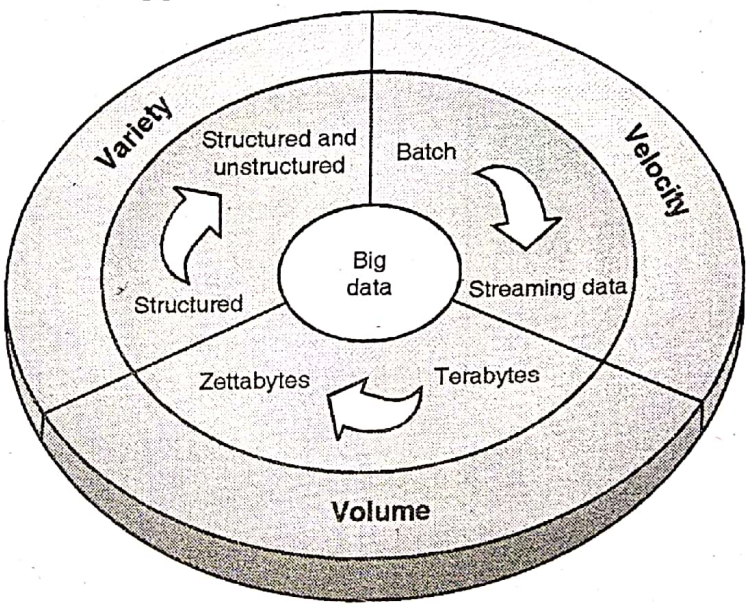

Fig. 1.3.3

4. Veracity

- The data captured is is not in certain format.
- Data captured can vary greatly.
- So accuracy of analysis depends on the veracity of the source data.

5. Additional characteristics

(a) Programmable

- It is possible with big data to explore all types by programming logic.
- Programming can be used to perform any kind of exploration because of the scale of the data.

(b) Data driven

- The data driven approach is possible for scientists.
- As data collected is huge amounts.

(c) Multi Attributes

- It is possible to deal with many gigabytes of data that consist of thousands of attributes.
- As all data operations are now happening on a larger scale.

(d) Iterative

The more computing power can iterate on your models until you get them as per your own requirements.

1.4 Types of Big Data

Q. Write a short note on Types of Big Data.

1. Introduction

Hadoop, Hive, Pig, Cascading, Cascalog, mrjob, Caffeine, S4, MapR, Acunu, Flume, Kafka, Azkaban, Oozie, Greenplum

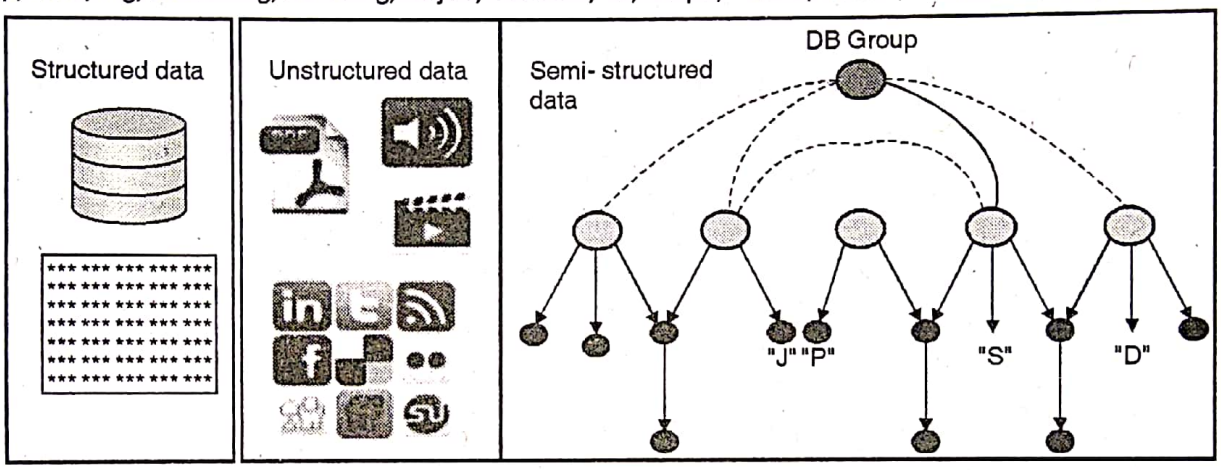

Fig. 1.4.1 : Types of big data

2. Structured data

- Structured data is generally data that has a definate length and format for big data.
- Like RDBMS tables have fixed number of colums and data can be increased by adding rows.

Example :

Structured data include marks data as numbers, dates or data like words and numbers. Structured data is very simple to dealing with, and easy to store in a database.

Sources of structured data

The data can be generated by human or it will be generated by machine.

(i) Human generated data

1. **Sensor data :** Radio frequency ID tags, medical devices, and Global Positioning System data.
2. **Web log data :** All kinds of data about their activity.
3. **Point-of-sale data :** Data associated with sales.
4. **Financial data :** Stock-trading data or banking transaction data.

(ii) Machine generated data

1. **Input data :** Survey data, responses sheets and so on.
2. **Click-stream data :** Data is generated every time you click a Link on a website.
3. **Gaming-related data :** Every move you make in a game can be recorded.

(iii) **Tools generates structured data**

(i) Data Marts (ii) RDBMS

(iii) Greenplum (iv) TeraData

3. Un-structured data

- Unstructured data is generally data collected in any available form without restricting them for any formats.
- Like audio, video data, Web blog data etc.

Example :

Unstructured data include video recording of CCTV surveillance.

Sources of unstructured data

The data can be generated by human or it will be generated by machine.

(i) **Human generated data**

1. **Satellite images :** This includes weather data or the data from satellite
2. **Scientific data :** This includes seismic imagery, weather forecasting data etc.
3. **Photographs and video :** This includes security, video etc.
4. **Radar or sonar data :** This includes vehicular, oceanographic data

(ii) **Machine generated data**

1. **Text data :** The documents, logs, survey results, and e-mails in company.
2. **Social media :** Generated from the social media platforms such as YouTube, Facebook etc.
3. **Mobile data :** This includes data such as text messages and location information etc.
4. **Website content :** Site data like YouTube, Flickr, or Instagram.

(iii) **Tools generates structured data**

1. Hadoop
2. HBase
3. Hive
4. Pig
5. Cloudera
6. MapR

4. Semi-structured data

- Along with structured and unstructured data, there's also a semi-structured data.
- Semi-structured data is information that doesn't reside in a RDBMS.
- It may organized in tree pattern which is easier to analyze in some cases.
- Examples of semi-structured data might include XML documents and NoSQL databases.

5. Hybrid data

- There are systems which will make use of both types of data to achieve competitive advantages.
- Structured data is offering simplicity whereas unstructured data will give lot of data about topic.

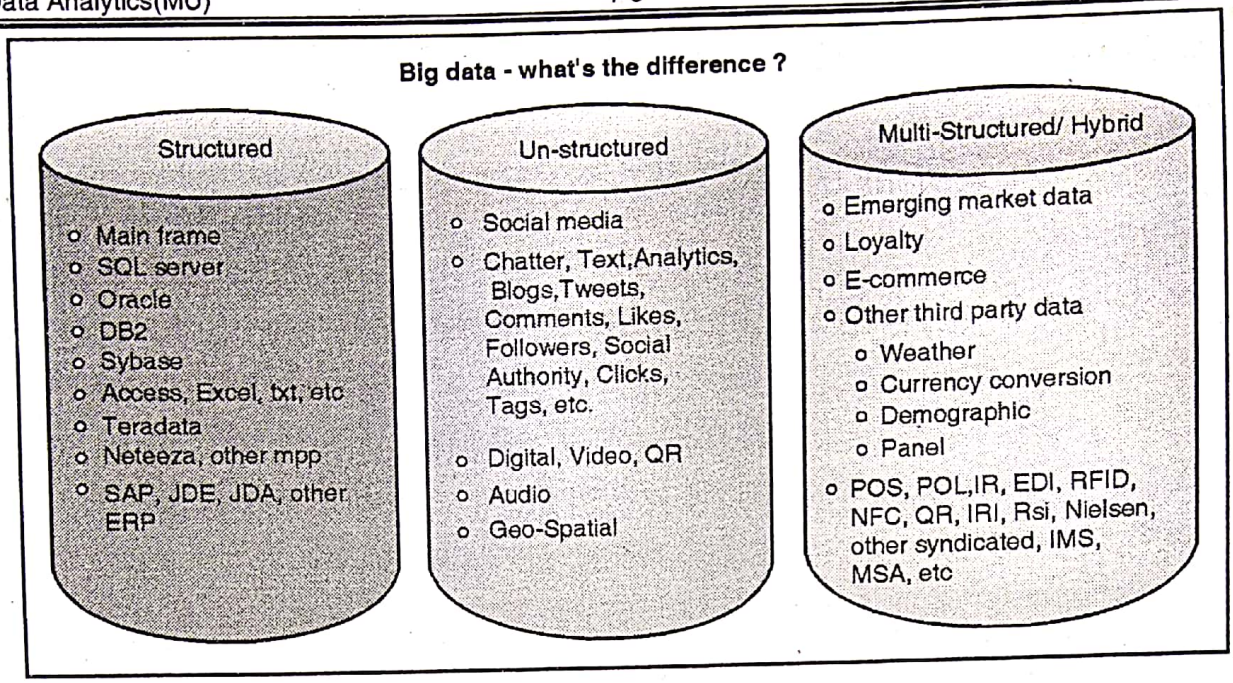

Fig. 1.4.2

1.5 Big Data vs. Traditional Data Business Approach

MU - Dec. 18

Q. Compare Traditional approach and traditional big data approach. **(Dec. 18, 3 Marks)**

The modern world is generating massive volumes of data at very fast rates. As a result, big data analytics is becoming a powerful tool for businesses looking to mining valuable data for competitive advantage.

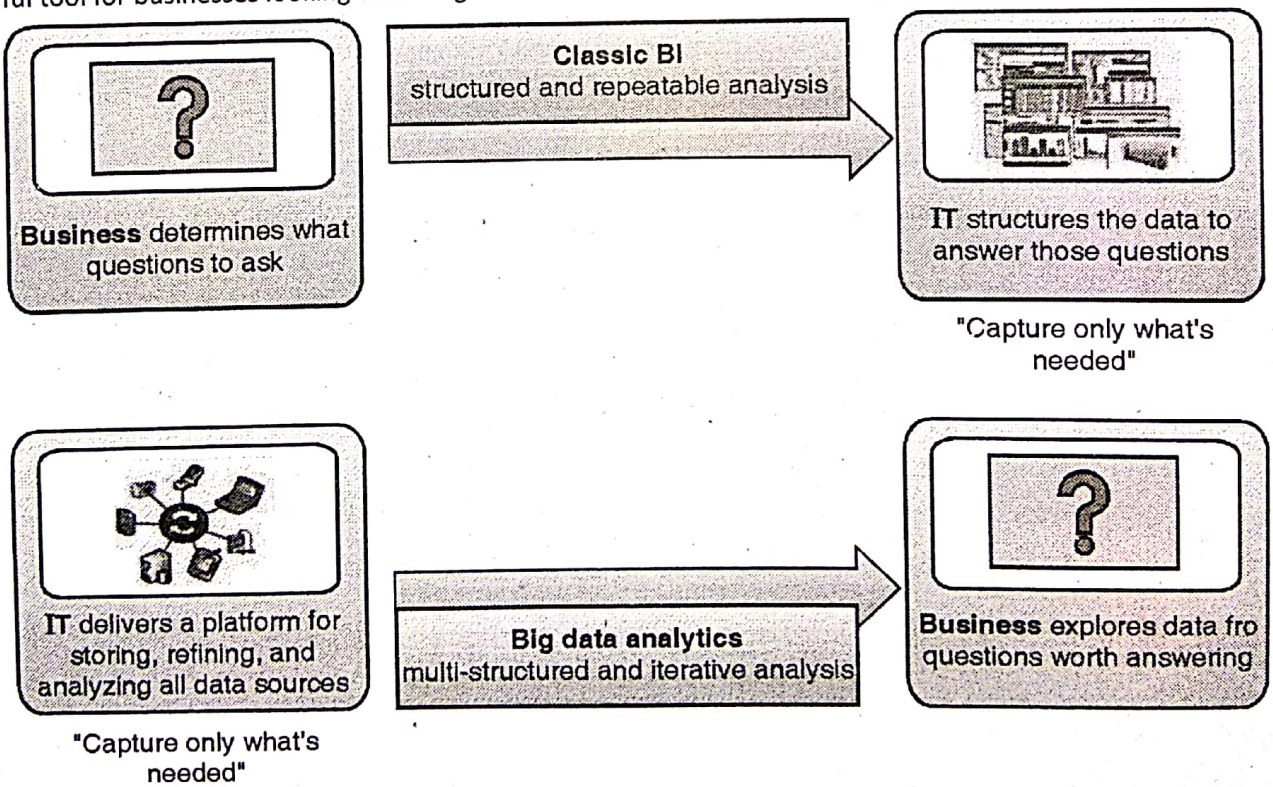

Fig. 1.5.1

1. Traditional business intelligence

- There are many systems distributed throughout the organization.
- The traditional data warehouse and business intelligent approach required extensive data analysis work with each of the systems and extensive transfer of data.
- Traditional Business Intelligence (BI) systems offer various levels and types of analyses on structured data but they are not designed to handle unstructured data.
- For these systems Big Data may creates a big problems due to data that flows in either structured or unstructured way.
- This makes them limited when it comes to delivering Big Data benefits.
- Many of the data sources are incomplete, do not use the same definitions, and not always available.
- Saving all the data from each system to a centralized location makes it unfeasible.

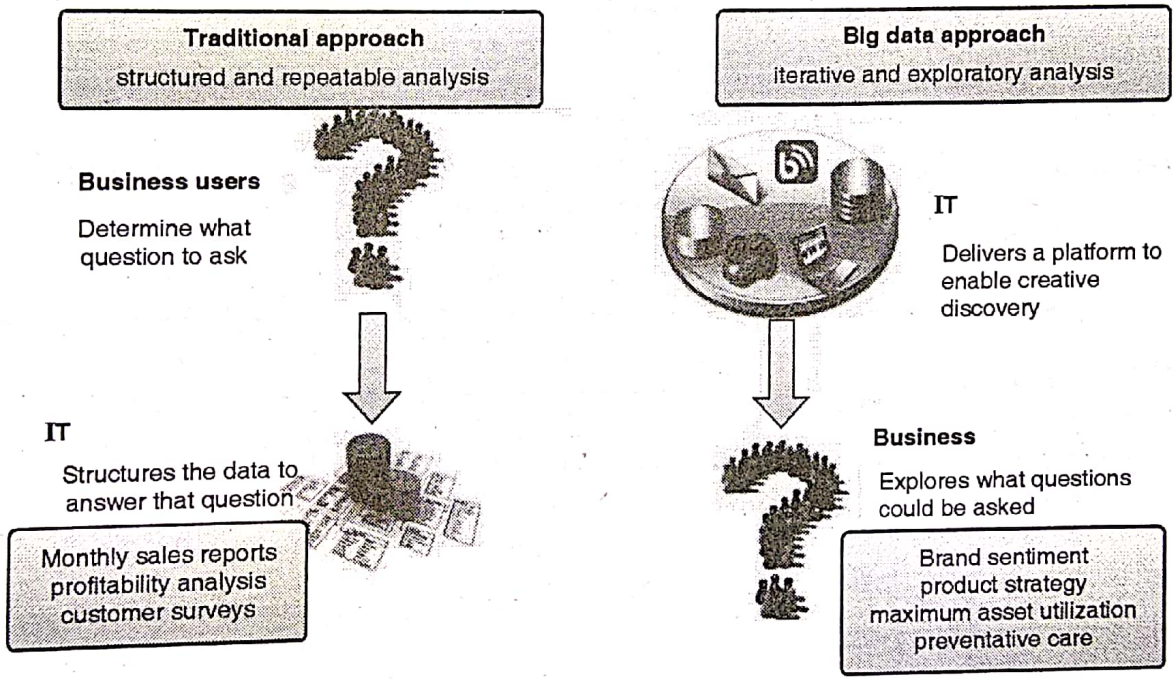

Fig. 1.5.2 : Traditional business intelligence

2. Big data analysis

- Big data means large or complex data set that traditional data processing applications may no be able to process efficientely.
- The Big data analytics involves data analysis, data capture, search, sharing, storage, transfer, visualization, querying and information security.
- The term generally only used for predictive analytics.

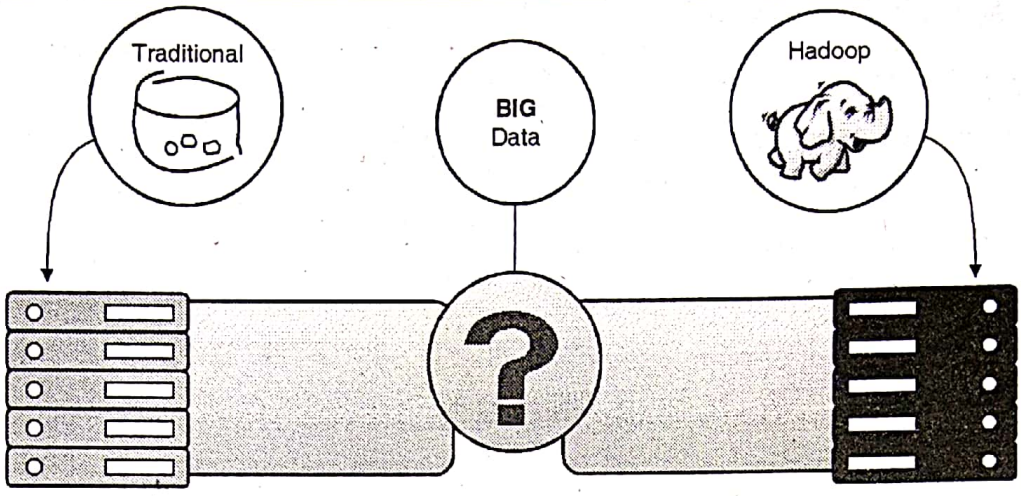

Fig. 1.5.3 : Big data analysis

- The efficiency of big data may lead to more confident decision making, and better decisions which can result in greater operational efficiency, cost reduction and reduced risk.
- Cloud-based platform can be used for the business world's big data problems.
- There can be some situations where running workloads on a traditional database may be the better solution.

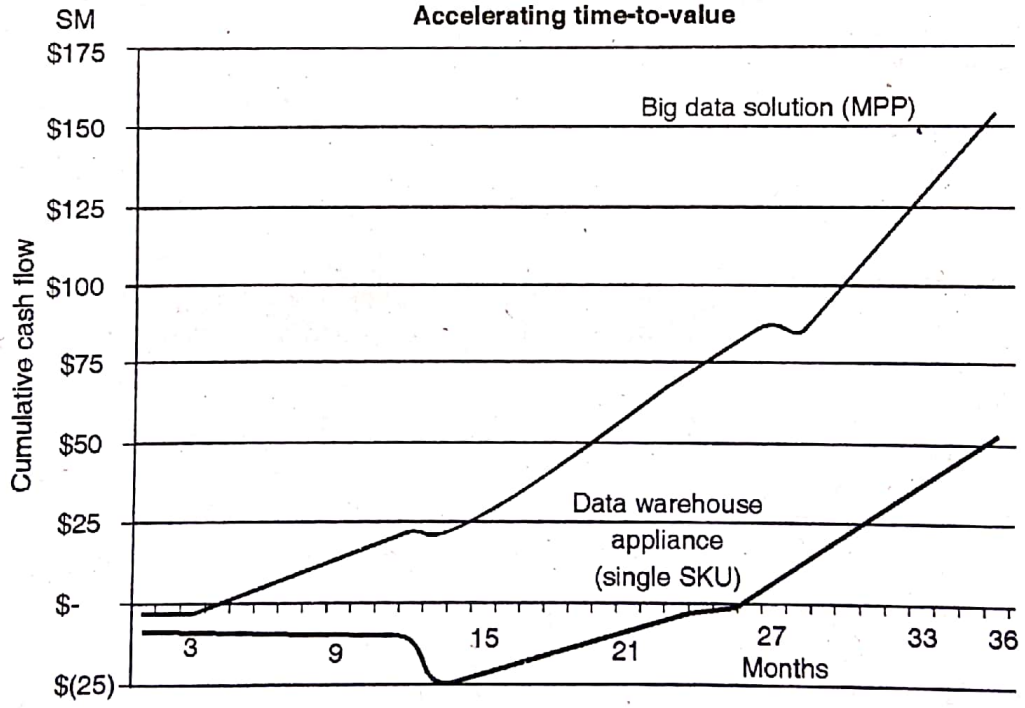

Fig. 1.5.4

3. Comparison of traditional and big data

Sr. No.	Dimension	Traditional	Big Data
1	Data source	Mainly internal.	Both inside and outside organization including traditional.
2	Data structure	Pre-defined structure.	Unstructured in nature.

Sr. No.	Dimension	Traditional	Big Data
3	Data relationship	By default, stable and interrelationship.	Unknown relationship.
4	Data location	Centralized.	Physically highly distributed.
5	Data analysis	After the complete build.	Intermediate analysis, as you go.
6	Data reporting	Mostly canned with limited and pre-defined interaction paths.	Reporting in all possible direction across the data in real time mode.
7	Cost factor	Specialized high end hardware and software.	Inexpensive commodity boxes in cluster mode.
8	CAP theorem	Consistency - Top priority.	Availability - Top priority.

4. Comparison between RDBMS and Hadoop

Sr. No.		Traditional RDBMS	Hadoop / Map Reduce
1	Data size	Gigabytes (Terabytes)	Petabytes (Hexabytes)
2	Access	Interactive and Batch	Batch – NOT interactive
3	Updates	Read/Write many times	Write once, Read many times
4	Structure	Static schema	Dynamic schema
5	Integrity	High (ACID)	Low
6	Scaling	Non linear	Linear
7	Query response time	Can be near immediate	Has latency (due to batch processing)

1.6 Tools used for Big Data

Q. Explain various tools used in Big Data.

1. **MapReduce**

   Hadoop, Hive, Pig, Cascading, Cascalog, mrjob, Caffeine, S4, MapR, Acunu, Flume, Kafka, Azkaban, Oozie, Greenplum

2. **Storage**

   S3, Hadoop Distributed File System

3. **Servers**

   EC2, Google App Engine, Elastic, Beanstalk, Heroku

4. **NoSQL**

   ZooKeeper Databases, MongoDB, Cassandra, Redis, BigTable, Hbase, Hypertable, Voldemort, Riak, CouchDB

5. **Processing**

   R, Yahoo! Pipes, Mechanical Turk, Solr/Lucene, ElasticSearch, Datameer, BigSheets, Tinkerpop

1.7 Data Infrastructure Requirements

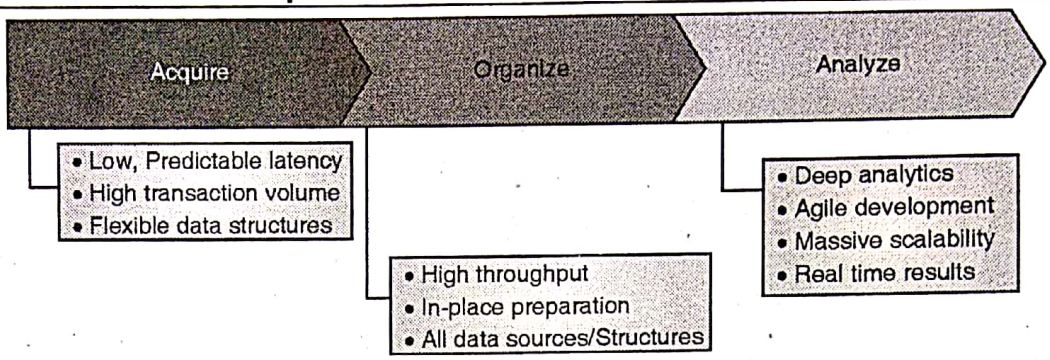

Fig. 1.7.1 : Data infrastructure requirement

1. **Acquiring data**

   High volume of data and transactions are the basic requirements of big data. Infrastructure should support the same. Flexible data structures should be used for the same. The amount of time required for this should be as less as possible.

2. **Organizing data**

   As the data may be structured, semi structured or unstructured it should be organized in a fast and efficient way.

3. **Analyzing data**

   Data analysis should be faster and efficient. It should support distributed computing.

1.8 Case Studies of Big Data Solutions

MU - May 16

Q. Give two examples of big data case studies. Indicate which vs are satisfied by these case studies. **(May 16, 4 Marks)**

There are many cases, in which big data solutions can be used effectively,

1. **Healthcare and public health industry**

   - The computing power of big data analytics enables us to predict disease, allow us to find new cures and better understand and predict disease patterns.
   - Like entire DNA strings can be decoded in minutes
   - Smart watches can be used to apply to predict symptoms for various diseases.
   - Big data techniques are already being used to monitor babies in a specialist premature and sick baby unit. By recording and analyzing every heart beat and breathing pattern of every baby, the unit was able to develop algorithms that can now predict infections 24 hours before any physical symptoms appear.
   - Big data analytics allow us to monitor and predict the developments of epidemics and disease. Integrating data from medical records with social media analytics enables us to monitor few diseases.

2. **Sports**

   - All sports popularly performing all analysis in big data analytics.
   - The Cricket and football matches you will observe many predictions which found correct maximum times.

- Example is IBM SlamTracker tool for tennis tournaments
- Also video analytics track the performance of every player in a cricket game, and sensor technology in sports equipment such as basket ball allows us to get feedback even using smart phones.
- Many elite sports teams also track athletes outside of the sporting environment using smart technology to track nutrition and sleep, as well as social media conversations.

3. Science and research

- Science and research is currently being transformed by the new possibilities big data.
- Experiments involve testing lot of possibilities of test cases and generate huge amounts of data.
- The many advanced lab uses computing powers of thousands of computers distributed across many data centers worldwide to analyze the data,
- It will help to leveraged performance areas of science and research.

4. Security enforcement

- Big data is applied for improving national security enforcement.
- The National Security Agency (NSA) in the U.S. uses big data analytics to foil terrorist plots.
- The big data techniques are used to detect and prevent cyber attacks.
- Police forces use big data tools to catch criminals and even predict criminal activity and credit card companies use big data use it to detect fraudulent transactions.

5. Financial trading

- Automated Trading and High-Frequency Trading (HFT) is a new area where big data can play a role.
- The big data algorithms can be used to make trading decisions.
- Today, the majority of equity trading now takes place via data algorithms that increasingly take into account signals from social media networks and news websites to make buy and sell decisions in split seconds.

Review Questions

Q. 1 Write a short note on Big Data.

Q. 2 Explain various applications of Big Data applications.

Q. 3 Give all characteristics of Big Data.

Q. 4 Explain three vs of Big Data.

Q. 5 Explain various types of big data in details.

Q. 6 Why to use Big Data over traditional business approach ?

Q. 7 Compare Traditional approach and traditional big data approach.

Q. 8 Explain various needs of Big Data.

Q. 9 Explain various tools used in Big Data.

Q. 10 Write a short note on :

(a) Types of Big Data

(b) Traditional vs Big Data business approach.

□□□

CHAPTER 2

Module - 1

Introduction to Hadoop

Syllabus

Concept of Hadoop, Core Hadoop Components, Hadoop Ecosystem

2.1 Hadoop

MU - May 17

Q. What is Hadoop? How Big Data andHadoop are linked? **(May 17, 4 Marks)**

Q. Write a short note on Hadoop.

- Hadoop is an open-source, big data storage and processing software framework. Hadoop stores and process big data in a distributed fashion on large clusters of commodity hardware. Massive data storage and faster processing are the two important aspects of Hadoop.

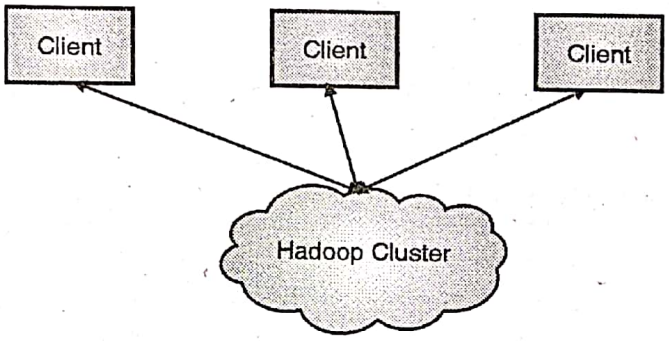

Fig. 2.1.1 : Hadoop cluster

- As shown in Fig. 2.1.1 Hadoop cluster is a set of commodity machines networked together in one location i.e. cloud.
- These cloud machines are then used for Data storage and processing. From individual clients users can submit their jobs to cluster. These clients may be present at some remote locations from the Hadoop cluster.
- Hadoop run applications on systems with thousands of nodes involving huge storage capabilities. As distributed file system is used by Hadoop data transfer rates among nodes are very faster.
- As thousands of machines are there in a cluster user can get uninterrupted service and node failure is not a big issue in Hadoop even if a large number of nodes become inoperative.
- Hadoop uses distributed storage and transfers code to data. This code is tiny and consumes less memory also.
- This code executes with data there itself. Thus the time to get data and again restore results is saved as the data is locally available. Thus interprocess communication time is saved which makes it faster processing.
- The redundancy of data is important feature of Hadoop due to which node failures are easily handled.
- In Hadoop user need not to worry about partitioning the data, data and task assignment to nodes, communication between nodes. As Hadoop handles it all, user can concentrate on data and operations on that data.

2.1.1 Hadoop - Features

1. Low cost

As Hadoop is an open-source framework, it is free. It uses commodity hardware to store and process huge data. Hence it is not much costly.

2. High computing power

Hadoop uses distributed computing model. Due to this task can be distributed amongst different nodes and can be processed quickly. Cluster have thousands of nodes which gives high computing capability to Hadoop.

3. Scalability

Nodes can be easily added and removed. Failed nodes can be easily detected. For all these activities very little administration is required.

4. Huge and flexible storage

Massive data storage is available due to thousands of nodes in the cluster. It supports both structured and unstructured data. No preprocessing is required on data before storing it.

5. Fault tolerance and data protection

If any node fails the tasks in hand are automatically redirected to other nodes. Multiple copies of all data are automatically stored. Due to this even if any node fails that data is available on some other nodes also.

2.1.2 Hadoop and Traditional RDBMS

Sr. No.	Hadoop	RDBMS
1.	Hadoop stores both structured and unstructured data.	RDBMS stores data in a structural way.
2.	SQL can be implemented on top of Hadoop as the execution engine.	SQL (Structured Query Language) is used.
3.	Scaling out is not that much expensive as machines can be added or removed with ease and little administration.	Scaling up (upgradation) is very expensive.
4.	Basic data unit is key/value pairs.	Basic data unit is relational tables.
5.	With MapReduce we can use scripts and codes to tell actual steps in processing the data.	With SQL we can state expected result and database engine derives it.
6.	Hadoop is designed for offline processing and analysis of large-scale data.	RDBMS is designed for online transactions.

2.2 Hadoop System Principles

1. Scaling out

In Traditional RDBMS it is quite difficult to add more hardware, software resources i.e. scale up. In Hadoop this can be easily done i.e. scale down.

2. Transfer code to data

In RDBMS generally data is moved to code and results are stored back. As data is moving there is always a security threat. In Hadoop small code is moved to data and it is executed there itself. Thus data is local. Thus Hadoop correlates preprocessors and storage.

3. Fault tolerance

Hadoop is designed to cope up with node failures. As large number of machines are there, a node failure is very common problem.

4. Abstraction of complexities

Hadoop provides proper interfaces between components for proper working.

5. Data protection and consistency

Hadoop handles system level challenges as it supports data consistency.

2.3 Hadoop Physical Architecture

MU - May 17, Dec. 18

Q. Explain Physical architecture of Hadoop. (May 17, Dec. 18, 4 Marks)

- Running Hadoop means running a set of resident programs. These resident programs are also known as daemons.
- These daemons may be running on the same server or on the different servers in the network.
- All these daemons have some specific functionality assigned to them. Let us see these daemons.

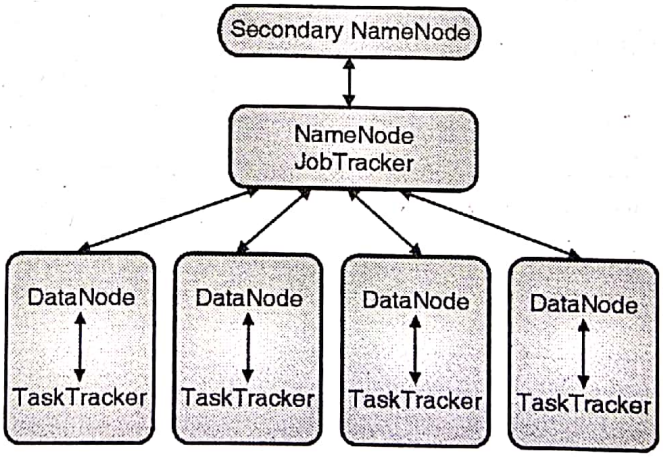

Fig. 2.3.1 : Hadoop cluster topology

NameNode

1. The NameNode is known as the master of HDFS.
2. DataNode is known as the slave of HDFS.
3. The NameNode has JobTracker which keeps track of files distributed to DataNodes.
4. NameNode directs DataNode regarding the low-level I/O tasks.
5. NameNode is the only single point of failure component.

DataNode

1. DataNode is known as the slave of HDFS.
2. The DataNode takes client block addresses from NameNodes.

3. Using this address client communicates directly with the DataNode.
4. For replication of data a DataNode may communicate with other DataNodes.
5. DataNode continually informs local change updates to NameNodes.
6. To create, move or delete blocks DataNode receives instructions from the local disk.

Secondary NameNode (SNN)

1. State monitoring of cluster HDFS is done by SNN.
2. Every cluster has one SNN.
3. SNN resides on its own machine also.
4. On the same server any other DataNode or TaskTracker daemons cannot run.
5. The SNN takes snapshots of the HDFS metadata at intervals by communicating constantly with NameNode.

JobTracker

1. JobTracker determines files to process, node assignments for different tasks, tasks monitoring etc.
2. Only one JobTracker daemon per Hadoop cluster is allowed.
3. JobTracker runs on a server as a master node of the cluster.

TaskTracker

1. Individual tasks assigned by JobTracker are executed by TaskTracker.
2. There is a single TaskTracker per slave node.
3. TaskTracker may handle multiple tasks parallelly by using multiple JVMs.
4. TaskTracker constantly communicates with the JobTracker. Within a specified amount of time if the TaskTracker fails to respond to JobTracker then it is assumed that the TaskTracker has crashed. Rescheduling of corresponding tasks are done to other nodes in the cluster.

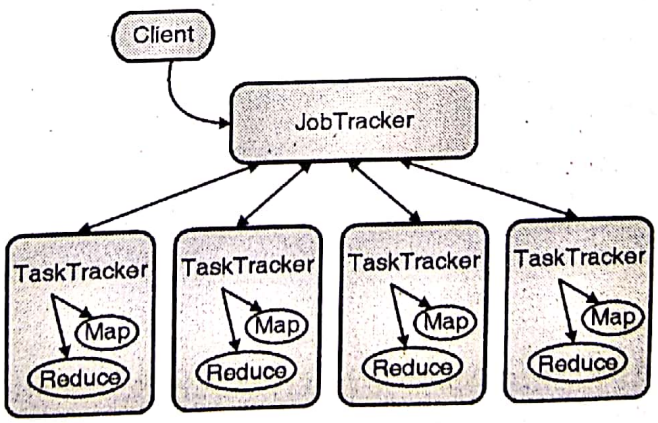

Fig. 2.3.2 : Jobtracker and tasktracker interaction

2.4 Hadoop Core Components

Q. Explain components of Core Hadoop.

- HDFS is a file system for Hadoop.
- It runs on clusters on commodity hardware.
  - HDFS - Hadoop distributed file system (storage)
  - Map reduce (processing)

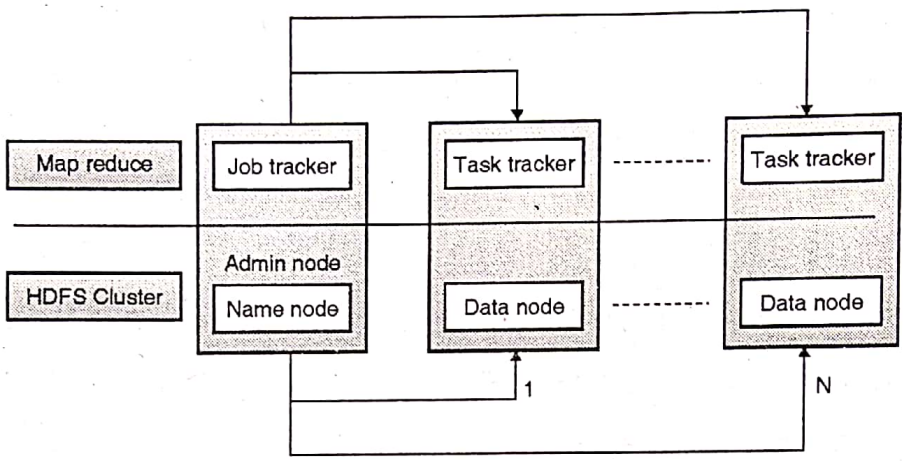

Fig. 2.4.1 : Hadoop core components

2.4.1 HDFS (Hadoop Distributed File System)

MU - Dec 17

Q. Describe the structure of HDFS in a hadoop ecosystem using a diagram. (Dec 17, 5 Marks)

- HDFS is a file system for Hadoop.
- It runs on clusters on commodity hardware.
- HDFS has following important characteristics :
  - Highly fault-tolerant
  - High throughput
  - Supports application with massive data sets
  - Streaming access to file system data
  - Can be built out of commodity hardware.

HDFS Architecture

- For distributed storage and distributed computation Hadoop uses a master/slave architecture. The distributed storage system in Hadoop is called as the Hadoop Distributed File System or HDFS. In HDFS a file is chopped into 64MB chunks and then stored, known as blocks.
- As previously discussed HDFS cluster has Master (NameNode) and Slave (DataNode) architecture. Name Node manages the namespace of the filesystem.

- In this namespace the information regarding file system tree, metadata for all the files and directories in that tree etc. is stored. For this it creates two files the namespace image and the edit log and stores information in it on consistent basis.
- A *client* interacts with HDFS by communicating with the Name Node and Datanodes. The user does not know about the assignment of Name Node and Data Node for functioning. i.e. Which NameNode and DataNodes are assigned or will be assigned.

1. NameNode

- The NameNode is known as the master of HDFS.
- DataNode is known as the slave of HDFS.
- The NameNode has Job Tracker which keeps track of files distributed to DataNodes.
- NameNode directs DataNode regarding the low-level I/O tasks.
- NameNode is the only single point of failure component.

2. DataNode

- DataNode is known as the slave of HDFS.
- The DataNode takes client block addresses from NameNodes.
- Using this address client communicates directly with the DataNode.
- For replication of data a DataNode may communicate with other DataNodes.
- DataNode continually informs local change updates to NameNodes.
- To create, move or delete blocks DataNode receives instructions from the local disk.

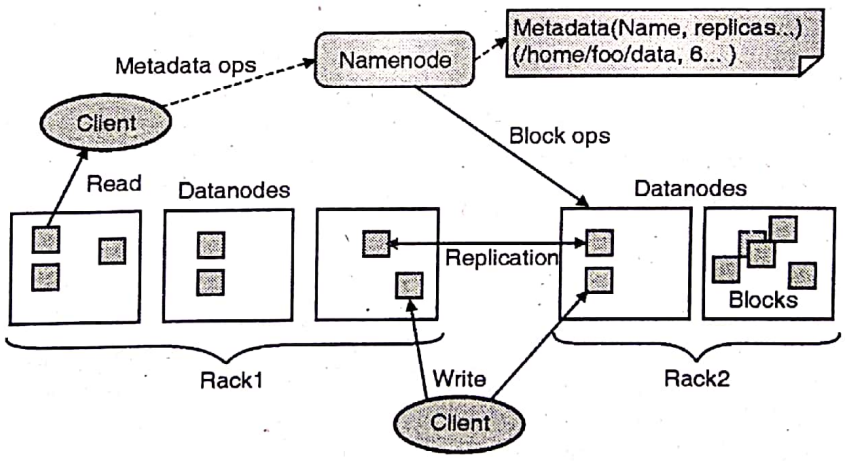

Fig. 2.4.2 : HDFS architecture

2.4.2 MapReduce

MU - May 17

Q. What is MapReduce? Explain How MapReduce work ? (May 17, 8 Marks)

- MapReduce is a software framework. In Mapreduce an application is broken down into number of small parts.

- These small parts are also as called fragments or blocks. These blocks then can be run on any node in the cluster.
- Data Processing is done by MapReduce. MapReduce scales and runs an application to different cluster machines.
- Required configuration changes for scaling and running for these applications are done by MapReduce itself. There are two primitives used for data processing by Mapreduce known as ***mappers*** and ***reducers***.
- Mapping and reducing are the two important phases for executing an application program. In the mapping phase MapReduce takes the input data, filters that input data and then transforms each data element to the mapper.
- In the reducing phase, the reducer processes all the outputs from the mapper, aggregates all the outputs and then provides a final result.
- MapReduce uses *lists* and *key/value pairs* for processing of data.

MapReduce core functions

1. **Read input**

   Divides input into small parts / blocks. These blocks then get assigned to a Map function

2. **Function mapping**

   It converts file data to smaller, intermediate <key, value> pairs.

3. **Partition, compare and sort**

   - **Partition function :** With the given key and number of reducers it finds the correct reducer.
   - **Compare function :** Map intermediate outputs are sorted according to this compare function

4. **Function reducing**

   Intermediate values are reduced to smaller solutions and given to output.

5. **Write output**

   Gives file output.

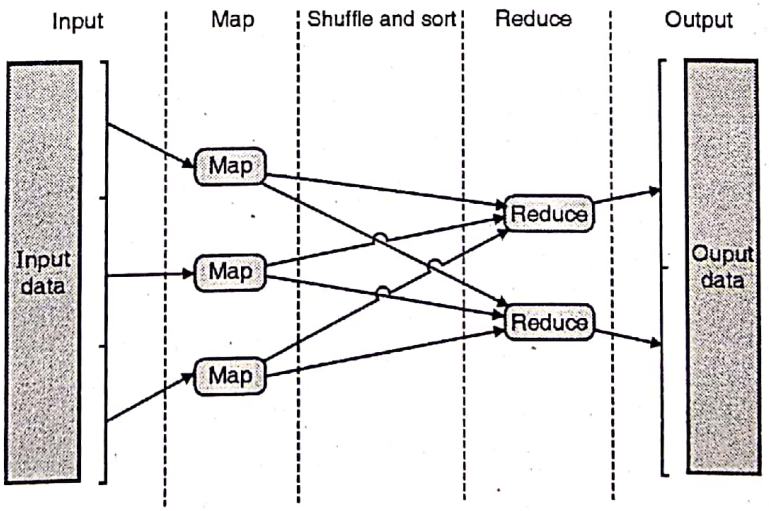

Fig. 2.4.3 : The general MapReduce dataflow

To understand how it works let us see one example,

File 1 : "Hello Sachin Hello Sumit"

File 2 : "Goodnight Sachin Goodnight Sumit"

Count occurrences of each word across different files.

Three operations will be there as follows,

(i) Map

Map1	**Map2**
< Hello, 1 >	< Goodnight, 1 >
< Sachin, 1 >	< Sachin, 1 >
< Hello, 1 >	< Goodnight, 1 >
< Sumit, 1 >	< Sumit, 1 >

(ii) Combine

Combine Map1	**Combine Map2**
< Sachin, 1 >	< Sachin, 1 >
< Sumit, 1 >	< Sumit, 1 >
< Hello, 2 >	< Goodnight, 2 >

(iii) Reduce

< Sachin, 2 >

< Sumit, 2 >

< Goodnight, 2 >

< Hello, 2 >

2.4.3 Hadoop - Limitation

MU - May 17, Dec.18

Q. State Limitations of Hadoop. **(May 17, Dec.18, 2 Marks)**

Q. What are the limitations of Hadoop?

- Hadoop can perform only batch processing and sequential access.
- Sequential access is time consuming.
- So a new technique is needed to get rid of this problem.

2.5 Hadoop - Ecosystem

MU - Dec. 16, May 17, May 18, Dec. 18

Q. Give Hadoop Ecosystem and briefly explain its components. **(Dec. 16, 10 Marks)**

Q. Explain Hadoop Ecosystem with core components. **(May 17, Dec. 18, 4 Marks)**

Q. What do you mean by the Hadoop Ecosystem?Describe any three components of a typical Hadoop Ecosystem. **(May 18, 10 Marks)**

Q. Explain Hadoop Ecosystem.

1. Introduction

- Hadoop can perform only batch processing and sequential access.
- Sequential access is time consuming.
- So a new technique is needed to get rid of this problem.
- The data in today's world is growing rapidly in size as well as scale up and shows no signs of slowing down.
- Statistics show that every year amount of data generated is more than previous years.
- The amount of unstructured data is much more than structured information stored in rows and columns.
- Big Data actually comes from complex, unstructured formats, everything from web sites, social media and email, to videos, presentations, etc.
- The pioneers in this field of data is Google, which designed scalable frameworks like MapReduce and Google File System.
- Apache open source has started with initiative by the name Hadoop, it is a framework that allows for the distributed processing of such large data sets across clusters of machines.

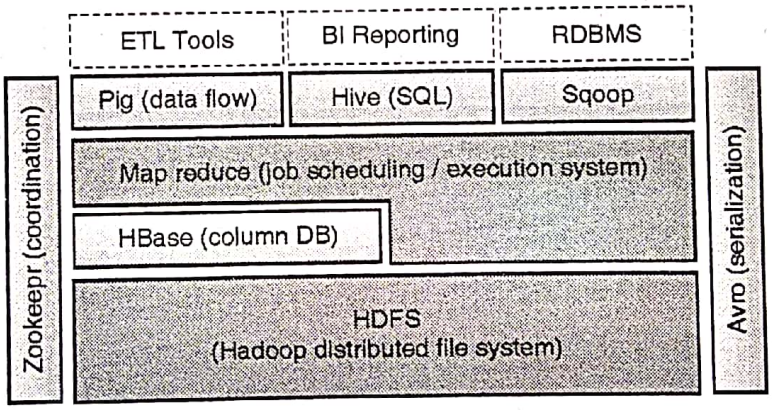

Fig. 2.5.1 : Hadoop ecosystem

2. Ecosystem

- Apache Hadoop, has 2 core projects,
  - Hadoop MapReduce
  - Hadoop Distributed File System
- Hadoop MapReduce is a programming model and software for writing applications which can process vast amounts of data in parallel on large clusters of computers.
- HDFS is the primary storage system, it creates multiple replicas of data blocks and distributes them on compute nodes throughout a cluster to enable reliable, extremely rapid computations.
- Other Hadoop-related projects are Chukwa, Hive, HBase, Mahout, Sqoop and ZooKeeper.

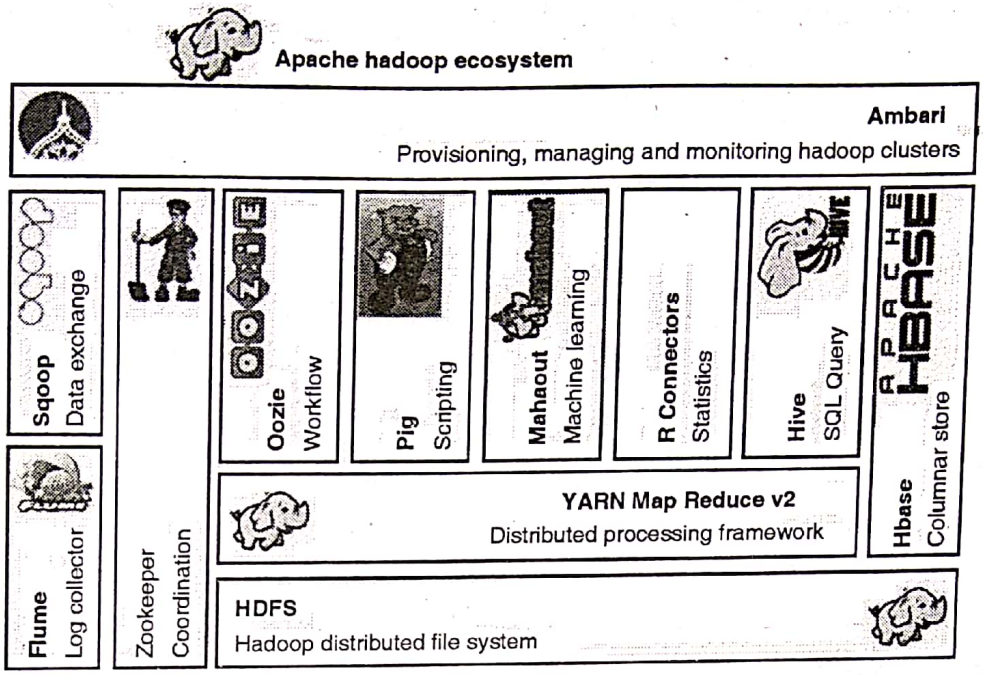

Fig. 2.5.2

2.6 ZooKeeper

1. ZooKeeper is a distributed, open-source coordination service for distributed applications used by Hadoop.
2. This system is a simple set of primitives that distributed applications can build upon to implement higher level services for synchronization, configuration maintenance, and groups and naming.

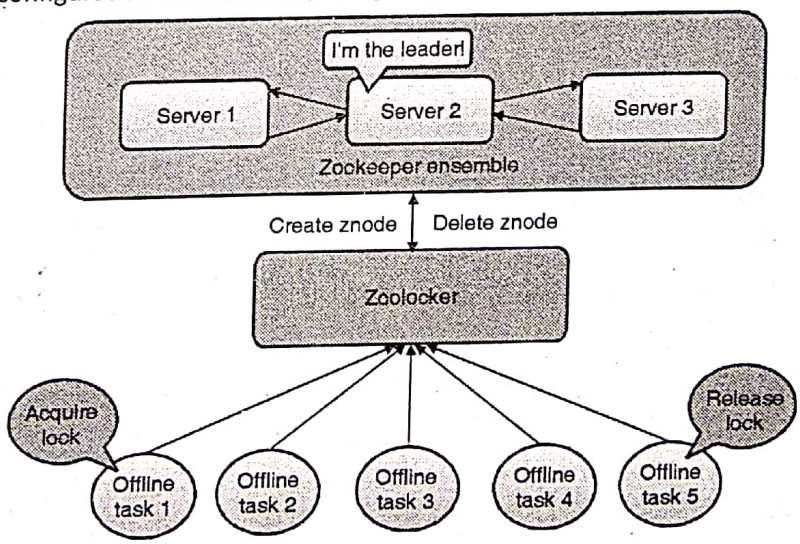

Fig. 2.6.1

3. This Coordination services are prone to errors such as race conditions and deadlock.
4. The main goal behind ZooKeeper is to use distributed applications.
5. ZooKeeper will allows distributed processes to coordinate with each other using shared hierarchical namespace organized as a standard file system.
6. The name space made up of of data registers called znodes, and these are similar to files and directories.
7. ZooKeeper data is kept in-memory, which means it can achieve high throughput and low latency.

2.7 HBase

- HBase is a distributed column-oriented database.
- HBase is hadoop application built on top of HDFS.
- HBase is suitable for huge datasets where real-time read/write random access is required.
- HBase is not a relational database. Hence does not support SQL.
- It is an open-source project and is horizontally scalable.
- Cassandra, couchDB, Dynamo and MongoDB are some other databases similar to HBase.
- Data can be entered in HDFS either directly or through HBase.
- Consistent read and writes, Automatic failure support is provided.
- It can be easily integrated with JAVA.
- Data is replicated across cluster. Useful when some node fails.

2.7.1 Comparison of HDFS and HBase

Sr. No.	HDFS	HBase
1.	HDFS is a distributed file system suitable for storing large files.	HBase is a database built on top of the HDFS.
2.	HDFS does not support fast individual record lookups.	HBase provides fast lookups for larger tables.
3.	It provides high latency batch processing.	Low latency random access.

2.7.2 Comparison of RDBMS and HBase

Sr. No.	RDBMS	HBase
1.	RDBMS uses schema. Data is stored according to its schema.	HBase is schema-less. Only column families are defined.
2.	Scaling is difficult.	Horizontally scalable.

Sr. No.	RDBMS	HBase
3.	RDBMS is transactional.	No transactions are there in HBase.
4.	It has normalized data.	It has de-normalized data.
5.	It is good for structured data.	It is good for semi-structured as well as structured data.
6.	It is row oriented database.	It is column oriented database.
7.	It is suitable for Online Transaction Process (OLTP).	It is suitable for Online Analytical Processing (OLAP).

2.7.3 HBase Architecture

- The Master performs administration, cluster management, region management, load balancing and failure handling.
- Region Server hosts and manages servers, region splitting, read/write request handling, client communication etc.
- Region contains Write Ahead Log (WAL). It may have multiple regions. Region is made up of Memstore and HFiles in which data is stored. Zookeeper is required to manage all the services.

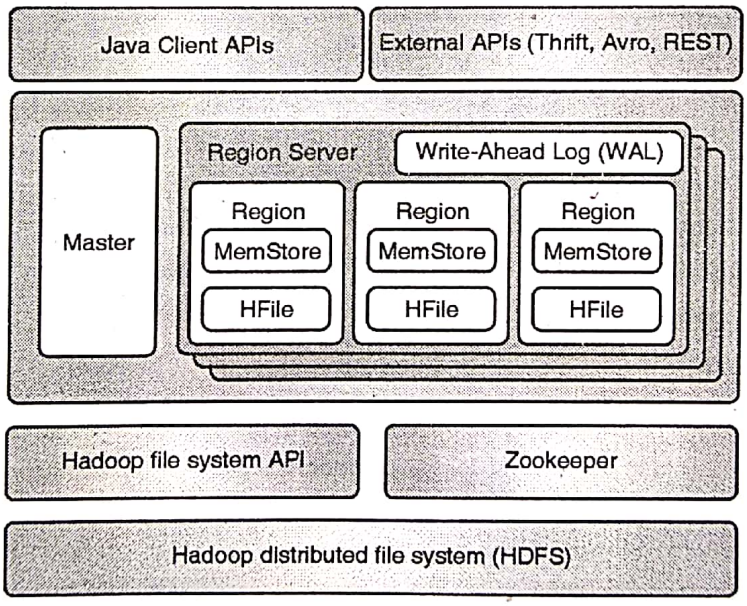

Fig. 2.7.1 : HBase database architecture

2.7.4 Region Splitting Methods

1. Pre splitting

Regions are created first and split points are assigned at the time of table creation. Initial set of region split points are to be used very carefully otherwise load distribution will be heterogeneous which may hamper clusters performance.

2. Auto splitting

This is by default action. It splits region when one of the stores crosses the max configured value.

3. Manual splitting

Split regions which are not uniformly loaded.

2.7.5 Region Assignment and Load Balancing

This information cannot be changed further as these are the standard procedures.

On startup

1. On startup Assignment Manager is invoked by Master.
2. From META the information about existing region assignments is taken by the AssignmentManager.
3. If the RegionServer is still online then the assignment is kept as it is.
4. If the RegionServer is not online then for region assignment the LoadBalancerFactory is invoked. The DefaultLoadBalancer will randomly assign the region to a RegionServer.
5. META is updated with this new RegionServer assignment. The RegionServer starts functioning upon region opening by the RegionServer.

When region server fails

1. Regions become unavailable when any RegionServer fails.
2. The Master finds which RegionServer is failed.
3. The region assignments done by that RegionServer then becomes invalid. The same process is followed for new region assignment as that of startup.

Region assignment upon load balancing

When there are no regions in transition, the cluster load is balanced by a load balancer by moving regions around. Thus redistributes the regions on the cluster. It is configured via hbase.balancer.period. The default value is 300000 (5 minutes).

2.7.6 HBase Data Model

- The Data Model in HBase is made of different logical components such as Tables, Rows, Column Families, Columns, Cells and Versions.
- It can handle semi-structured data that may be varied in terms of data type, size and columns. Thus partitioning and distributing data across the cluster is easier.

Row Key	Movies		Shows	
Screen	MovieName	Ticket	Time	Day
01	Harry Potter 1	200	6.00	Saturday
02	Harry Potter 2	250	3.00	Sunday

Column Families

Fig. 2.7.2 : HBase data model

1. Tables

Tables are stored as a logical collection of rows in Regions.

2. Rows

Each row is one instance of data. Each table row is identified by a rowkey. These rowkeys are unique and always treated as a byte[].

3. Column Families

Data in a row are grouped together as Column Families. These are stored in HFiles.

4. Columns

- A Column Family is made of one or more columns.
- A Column is accessed by, column family : columnname.
- There can be multiple Columns within a Column Family and Rows within a table can have varied number of Columns.

5. Cell

A Cell stores data as a combination of rowkey, Column Family and the Column (Column Qualifier).

6. Version

On the basis of timestamp different data versions are created. By default is the number of versions are 3 but it can be configured to some other value as well.

2.8 HIVE

- Hive is a data warehouse infrastructure tool.
- It processes structured data in HDFS. Hive structures data into tables, rows, columns and partitions.
- It resides on top of Hadoop.
- It is used to summarize big Data, analysis of big data.
- It is suitable for Online Analytical Application Processing.
- It supports ad hoc querries. It has its own SQL type language called HiveQL or HQL.
- SQL type scripts can be created for MapReduce operations using HIVE.
- Primitive datatypes like Integers, Floats, Doubles, and Strings are supported by HIVE.
- Associative Arrays, Lists, Structs etc. can be used.
- Serialize API and Deserialized API are used to store and retrieve data.
- HIVE is easy to scale and has faster processing.

2.8.1 Architecture of HIVE

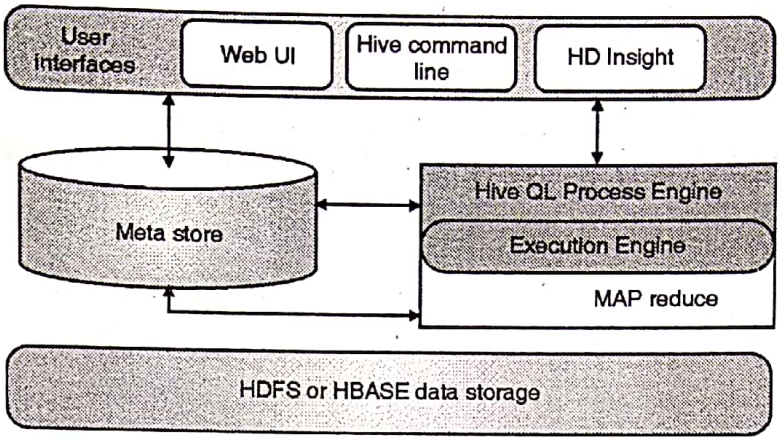

Fig. 2.8.1 : Hive architecture

This information cannot be changed further as these are the standard components.

1. **User interface**

Hive supports Hive Web UI, Hive command line, and Hive HD through which user can easily process queries.

2. **Meta store**

Hive stores Meta data, schema etc. in respective database servers known as metasores.

3. **HiveQL process engine**

HiveQL is used as querying language to get information from Metastore. It is an alternative to MapReduce Java program. HiveQL query can be written for MapReduce job.

4. **Execution engine**

Querry processing and result generation is the job of Execution engine. It is same as that of MapReduce results.

5. **HDFS or HBASE**

Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.

2.8.2 Working of HIVE

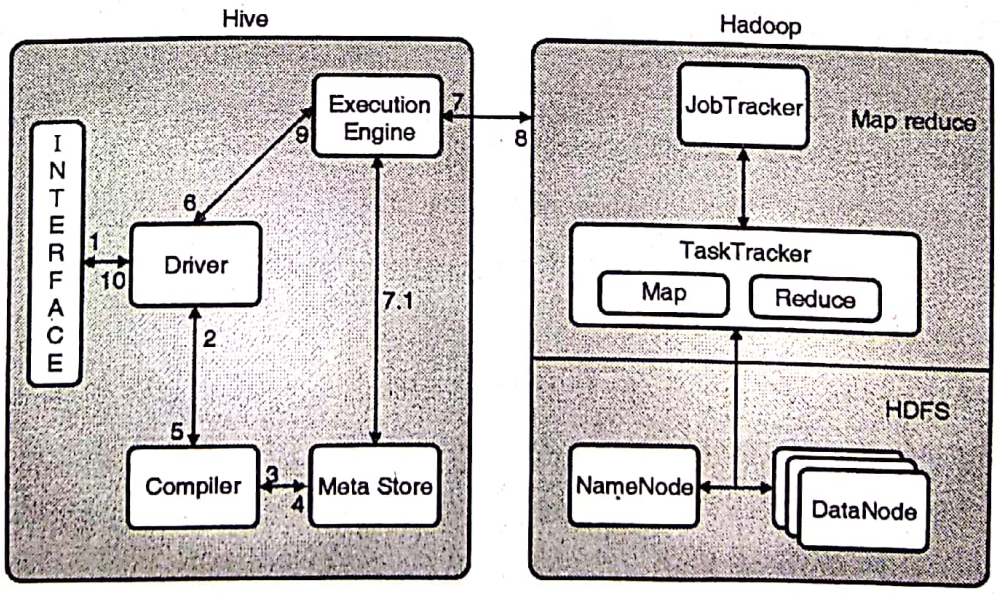

Fig. 2.8.2 : Hive and Hadoop communication

1. **Execute Query :** Command Line or Web UI sends query to JDBC or ODBC Driver to execute.
2. **Get Plan :** With the help of query compiler driver checks the syntax and requirement of query.
3. **Get Metadata :** The compiler sends metadata request to Metastore for getting data.
4. **Send Metadata :** Metastore sends the required metadata as a response to the compiler.
5. **Send Plan :** The compiler checks the requirement and resends the plan to the driver. Thus the parsing and compiling of a query is complete.
6. **Execute Plan :** The driver sends the execute plan to the execution engine.
7. **Execute Job :** The execution engine sends the job to JobTracker. JobTracker assigns it to TaskTracker.

7.1 **Metadata Operations :** The execution engine can execute metadata operations with Metastore.

8. **Fetch Result :** The execution engine receives the results from Data nodes.
9. **Send Results :** The execution engine sends those resultant values to the driver.
10. **Send Results :** The driver sends the results to Hive Interfaces.

2.8.3 HIVE Data Models

The Hive data models contain the following components,

1. Databases
2. Tables
3. Partitions
4. Buckets or clusters

Partitions

Table is divided into a smaller parts based on the value of a partition column. Then on these slices of data querries can be made for faster processing.

Buckets

Buckets give extra structure to the data that may be used for efficient queries. Different data required for querries joined together. Thus querries can be evaluated quickly.

Review Question

Q. 1 Write a short note on Hadoop.

Q. 2 What is Hadoop?

Q. 3 Explain components of Core Hadoop.

Q. 4 Explain Hadoop Ecosystem.

Q. 5 Explain physical architecture of Hadoop.

Q. 6 What are the limitations of Hadoop?

CHAPTER 3

Module - 2

Hadoop HDFS and MapReduce

Syllabus

Distributed File Systems : Physical Organization of Compute Nodes, Large-Scale File-System Organization, MapReduce : The Map Tasks, Grouping by Key, The Reduce Tasks, Combiners, Details of MapReduce Execution, Coping With Node Failures, Algorithms Using MapReduce : Matrix-Vector Multiplication by MapReduce, Relational-Algebra Operations, Computing Selections by MapReduce, Computing Projections by MapReduce, Union, Intersection, and Difference by MapReduce, Hadoop Limitations

3.1 Distributed File Systems

- In the early days of computing, most of the computations were done on a standalone computer.
- A standalone computer is also called a compute node. It consists of a single processor along with its main memory, cache memory and a local disk.
- As the computations got more complex and the data started getting bigger, the need for more computation power was felt. This gave rise to the concept of parallel computing.
- But the problem with parallel computing is that it requires many processors and specialized hardware. This makes the whole parallel processing arrangement very costly.
- A cost-effective solution to this problem is to use the already existing standalone computers and combine their individual computing resources to create a big pool of computing resource and yet allow the compute nodes to operate more or less independently.
- This new paradigm of computing is known as cluster computing. Fig. 3.1.1 shows cluster of compute nodes in general.

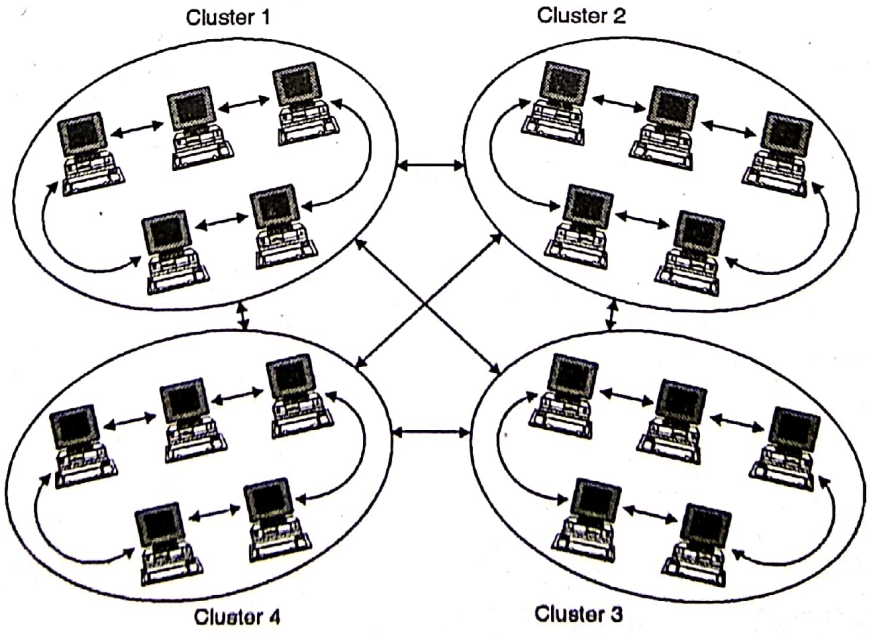

Fig. 3.1.1 : Cluster of compute nodes

3.1.1 Physical Organization of Compute Nodes

- The compute nodes are arranged in racks with each rack holding around 8 to 64 compute nodes as depicted in Fig. 3.1.2.

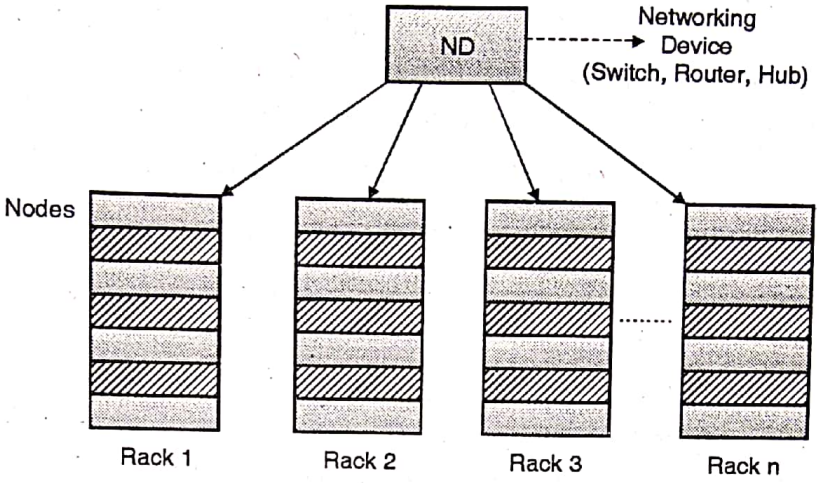

Fig. 3.1.2 : Compute nodes arranged in racks

- There are two levels of connections intra-rack and inter-rack. The compute nodes in a single rack are connected through a gigabit Ethernet and this is known as intra-rack connection. Additionally, the racks are connected to each other with another level of network or a switch which is known as the inter-rack connection.
- One major problem with this type of setup is that there are a lot of interconnected components and more the number of components, higher is the probability of failure. For example, single node failure or an entire rack failure.
- To make the system more robust against such type of failures the following steps are taken :
- Duplicate copies of files are stored at several compute nodes. This is done so that even if a compute node crashes, the file is not lost forever. This feature is known as "Data Replication". Fig. 3.1.3 shows data replication. Here, the data item D_1 is originally stored on Node 1 and a copy each is stored on Node 2 as well as Node 3. It means in total we have three copies of the same data item D_1. This is known as "Replication Factor" (RF).

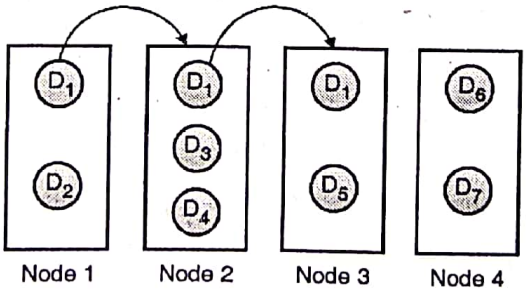

Fig. 3.1.3 : Data replication

- Computations are subdivided into tasks such that even if one task fails to complete execution, it may be restarted without affecting other tasks.

3.1.2 Large-Scale File-System Organization

- The conventional file systems present on standalone computers cannot take full advantage of cluster computing. For this reason a new type of file system is needed. This new file system is called Distributed File System or DFS.

- The types of files which are most suited to be used with DFS are :
  - Very large sized files having size in TBs or more.
  - Files with very less number of update operations compared to read and append operations.
- In DFS, files are divided into smaller units called "chunks". A chunk is usually of 64 MB in size. Each chunk is normally replicated and stored in three different compute nodes. It is also ensured that these three compute nodes are members of different racks so that in the event of rack failure at least one copy of the chunk is available.
- Both the chunk size and the replication factor can be adjusted by the user based on the demands of the application.
- All the chunks of a file and their locations are stored in a separate file called master node or name node. This file acts as an index to find the different chunks of a particular file. The master node is also replicated just like the individual chunks.
- The information about the master nodes and their replicas are stored in a directory. This directory in turn is replicated in a similar fashion and all the participants of the DFS are aware of the locations where the directory copies reside.
- There are many different implementations of the DFS described above such as :
  - Google File System (GFS),
  - Hadoop Distributed File System (HDFS),
  - Colossus, which is an improved version of GFS.

3.2 MapReduce

MU - Dec. 17, Dec. 18

Q. Explain concept of MapReduce using an example. **(Dec. 17, 5 Marks)**

Q. What is the MapReduce? Explain the role of combiner with the help of an example. **(Dec. 18, 10 Marks)**

- **MapReduce** can be used to write applications to process large amounts of data, in parallel, on large clusters of commodity hardware (a commodity hardware is nothing but the hardware which is easily available in the local market) in a reliable manner.
- **MapReduce** is a processing technique as well as a programming model for distributed computing based on java programming language or java framework.
- The MapReduce algorithm contains two important functions, namely Map and Reduce :
  - The Map tasks accept one or more chunks from a DFS and turn them into a sequence of key-value pairs. How the input data is converted into key-value pairs is determined by the code written by the user for the Map function.
  - A master controller collects and sorts the key-value pairs produced by the Map tasks. These sorted keys are then divided among the Reduce tasks. This distribution is done in such a way so that all the key-value pairs having the same key are assigned to the same Reduce task.
  - The Reduce tasks combine all of the values associated with a particular key. The code written by the user for the Reduce function determines how the combination is done.

3.2.1 The Map Tasks

- The input for a Map task is an element and the output is zero or more key-value pairs. An element could be anything such as a tuple or an entire document.
- Some useful assumptions are made which are:
  - An element is stored entirely in one chunk. That means one element cannot be stored across multiple chunks,
  - The types of keys and values both are arbitrary,
  - Keys need not be unique.
- Let us understand the MapReduce operations with an example. Let us suppose we are given a collection of documents and the task is to compute the counts of the number of times each word occurs in that collection.
- Here each document is an input element. One Map task will be assigned one or more chunks and that Map task will process all the documents in its assigned chunk(s).
- The output will be of the form:

  $(w_1, 1), (w_2, 1), (w_3, 1), \ldots, (w_n, 1)$

  Where $w_1, w_2, w_3, \ldots, w_n$ are the words in the document collection that is assigned to the Map task.
- If a particular word w appears c times, then the output will have c number of (w, 1) pairs.

3.2.2 Grouping by Key

- After the successful completion of all the Map tasks, the grouping of the key-value pairs is done by the master controller.
- The number of Reduce tasks r is set by the user in advance. The master controller uses a hash function that maps each key to the range 0 to r-1. In this step all the key-value pairs are segregated in r files according to the hash function output. These r files will be the input to the Reduce tasks.
- The master controller then performs the grouping by key procedure to produce a sequence of key/list-of-values pairs for each key k which are of the form $(k, [v_1, v_2, v_3, \ldots, v_n])$, where $(k, v_1), (k, v_2), (k, v_3), \ldots, (k, v_n)$ are the key-value pairs which were produced by all of the Map tasks.

3.2.3 The Reduce Tasks

- The input to a Reduce task is one or more keys and their list of associated values. And the output produced by the Reduce task is a sequence of zero or more key-value pairs which may be different from the key-value pairs produced by the Map tasks. But in most cases both the key-value pairs produced by the Reduce tasks and Map tasks are of the same type.
- In the final step the outputs produced by all of the Reduce tasks are combined in a single file.
- In our word count example, the Reduce tasks will add all the values for each key. The outputs will be of the form (w, s) pairs, where w is a word and sis the number of times it appears in the collection of documents.
- Fig. 3.2.1 shows the various MapReduce phases for the word frequency counting example.

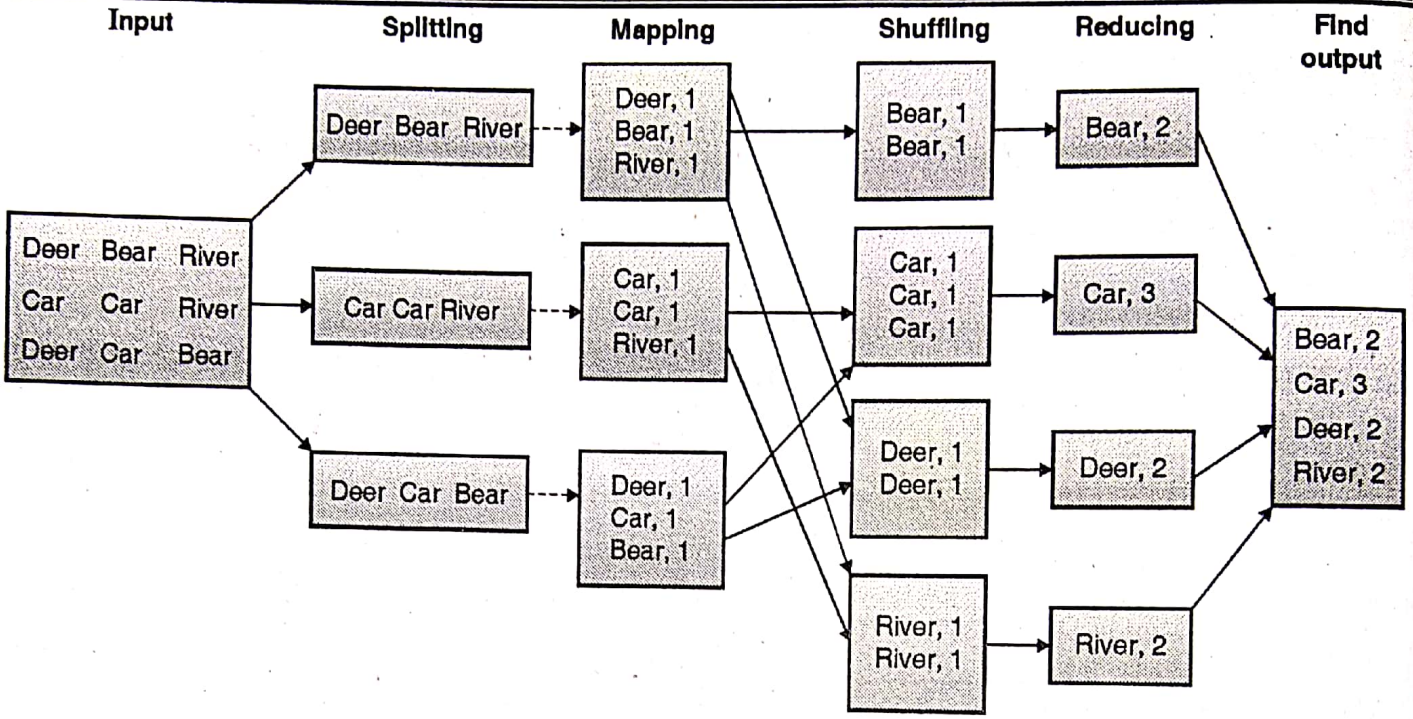

Fig. 3.2.1 : MapReduce process for word frequency count

3.2.4 Combiners

MU - May 17

Q. What are Combiners? When Should one use combiner in mapreduce job? **(May 17, 5 Marks)**

Q. What are combiners? Explain the position and significance of combiners.

- A combiner is a type of mediator between the mapper phase and the reducer phase. The use of combiners is totally optional. As a combiner sits between the mapper and the reducer, it accepts the output of map phase as an input and passes the key-value pairs to the reduce operation.
- Combiners are also known as semi-reducers as they reside before the reducer. A combiner is used when the Reduce function is commutative and associative. This means that the values can be combined in any order without affecting the final result. For example, addition is an operation which is both commutative as well as associative.
- In case of a huge dataset, when a MapReduce technique is applied, then the mapper phase generates an enormous amount of key-value pairs and these intermediate results need to be handed over to the reducer for the reduction process. But to transfer such a large number of key-value pairs from the mapper to the reducer, sufficiently large communication network bandwidth is required which generally leads to network congestion.
- So to avoid such congestion we can put some of the work of reducers in to the Map tasks. For example in a single Map task if c number of (w, 1) key-value pairs appear having the same key, they can be combined into a pair (w, c), where w is the key and c is the number of times the word w appears in the set of documents handled by this Map task.
- Fig. 3.2.2 shows position and working mechanism of combiner. Combiner reside between map phase and reduce phase which is responsible for making a group of values of same key which is received from mapper phase.

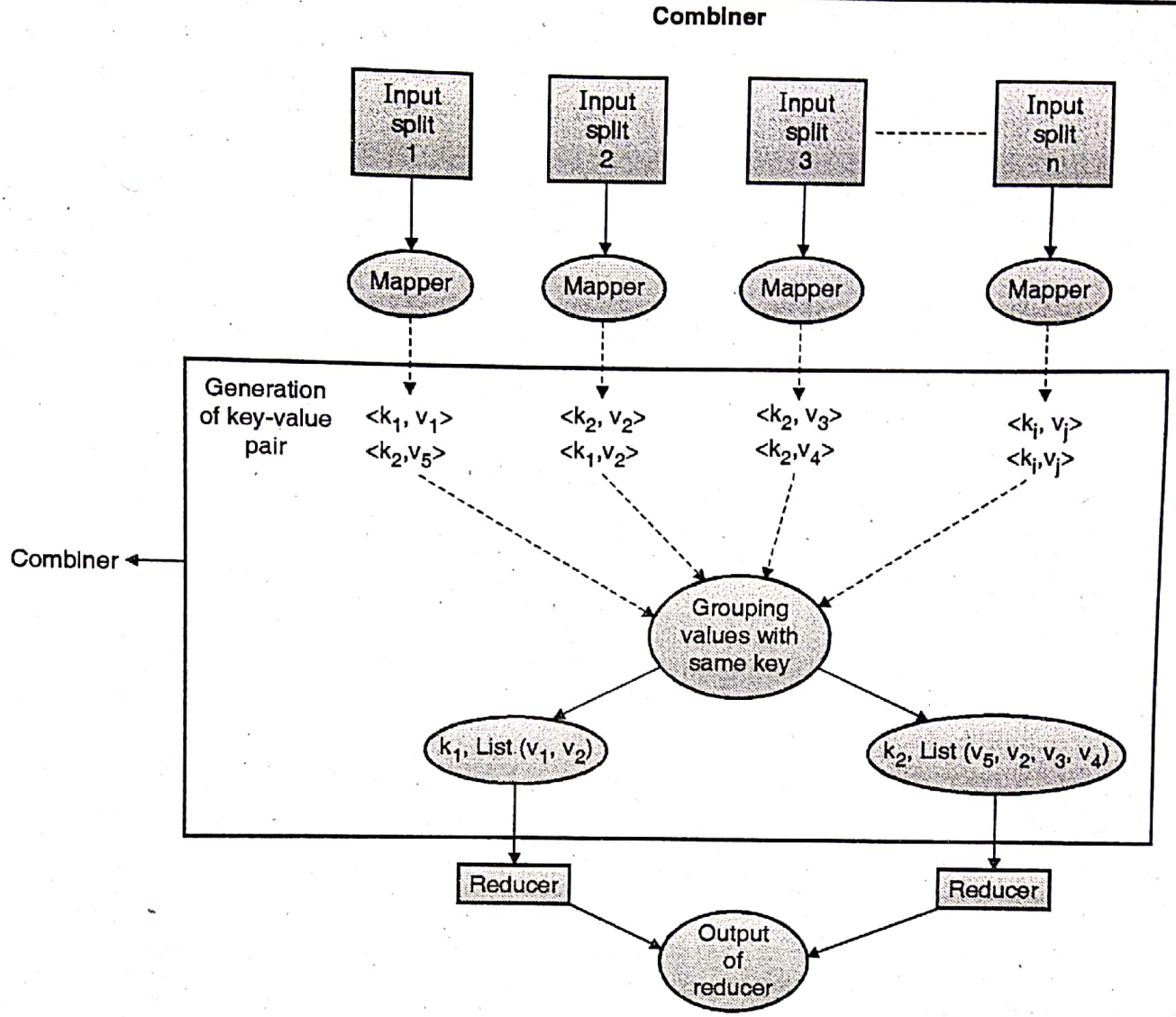

Fig. 3.2.2 : Position and Working Mechanism of Combiner

3.2.5 Details of MapReduce Execution

- In this section we will discuss in details how a MapReduce based program is executed. The user program first creates a Master controller process with the help of fork command using the library provided by the MapReduce system as depicted in Fig. 3.2.3.
- In addition to the Master process, the user program also forks a number of worker processes. These processes run on different compute nodes.
- The Master has the following responsibilities :
  - Creation of Map and Reduce tasks,
  - Assignment of Map and Reduce tasks to the Workers,
  - Keeping track of the Map and Reduce tasks (idle/executing/completed).
- A Worker process can be assigned either a Map task or a Reduce task but not both the tasks.

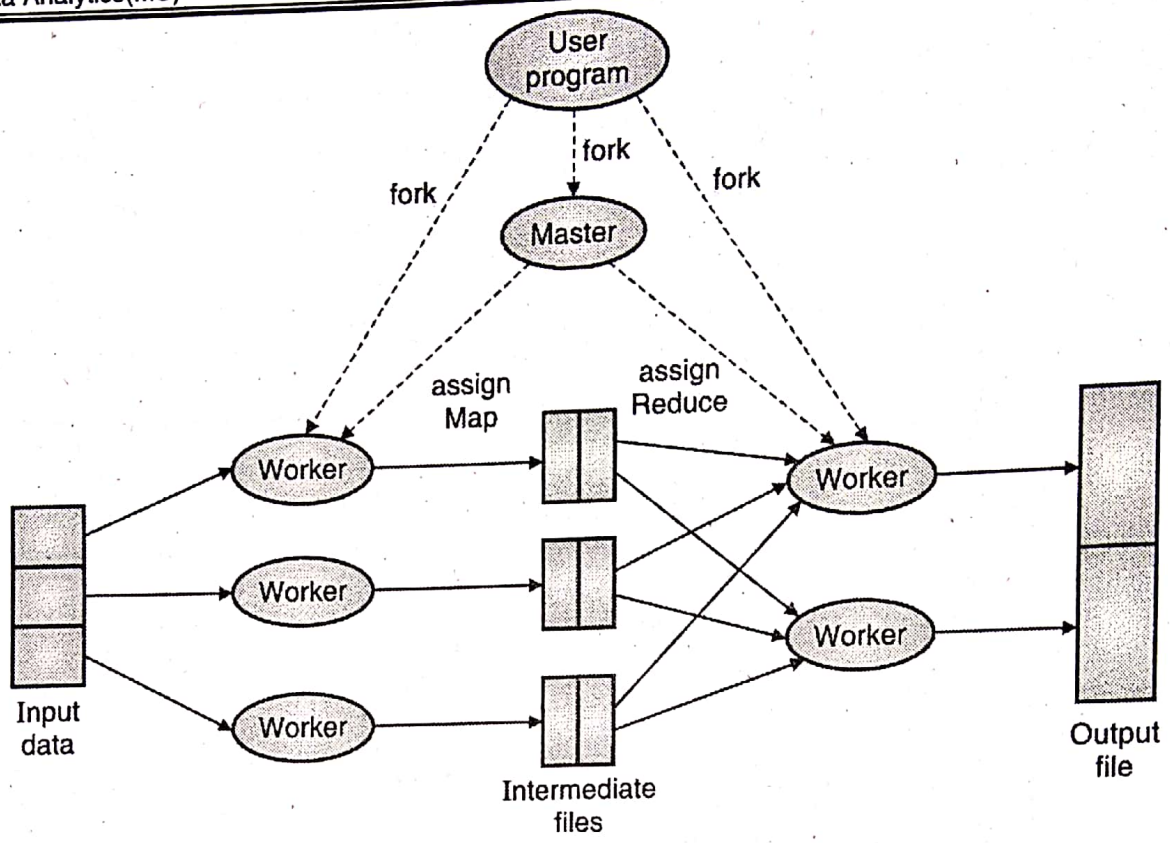

Fig. 3.2.3 : MapReduce program execution

- Every Map task creates one intermediate file in its local compute node for each of the Reduce tasks. So for example in the Fig. 3.2.3 there are two Reduce tasks, thus each of the Map tasks create two intermediate files which will be passed on as input to the Reduce tasks.
- All the Reduce tasks together will produce a single file as the final output of the MapReduce based program.

3.2.6 Coping with Node Failures

- There are three types of node failures :
  - Master node failure,
  - Map worker node failure,
  - Reduce worker node failure.
- If the Master node fails then the entire process has to be restarted. This is the worst kind of failure.
- If a Map worker node fails then the Master will assign the tasks to some other available worker node even if the task had completed.
- If a Reduce worker node fails then the tasks are simply rescheduled on some other Reduce worker later.

3.3 Algorithms using MapReduce

- MapReduce is not the solution for every problem. The distribute file system only makes sense for extremely large files which are very rarely updated.
- The main motivation behind the creation of Google's implementation of MapReduce was their PageRank algorithm which requires very large dimensional matrix-vector multiplications.
- In this section we will study a few algorithms which fit nicely into the MapReduce style of computing.

3.3.1 Matrix-Vector Multiplication by MapReduce

MU - May 16, May 19

Q. Explain matrix-vector multiplication algorithm by Map Reduce. (May 16, 10 Marks)

Q. Write pseudo code for Matrix vector Multiplication by MapReduce. Illustrate with an example showing all the steps. (May 19, 10 Marks)

- Let us consider a Matrix M of size n × n. Let m_{ij} denote the element in row i and column j. Let us also consider a vector v of length n and the j^{th} element of the vector is represented as v_j.
- The matrix-vector multiplication will produce another vector x whose ith element x_i is given by the formula:

$$x_i = \sum_j^n = 1\ m_{ij}v_j$$

- In real life applications such as Google's PageRank the dimensions of the matrices and vectors will be in trillions. Let us at first take the case where although the dimension n is large but it is also able to fit entirely in the main memory of the compute node.
- The Map task at each Map worker node works on a chunk of M and the entire v and produces the key-value pairs $(i, m_{ij}v_j)$. All the sum terms of the component x_i of the result vector of matrix-vector multiplication will be getting the same key i.
- The Reduce tasks sum all the values for a particular key and produce the result (i, x_i).
- In the case the where the vector v is too large to fit into the main memory of a compute node, an alternative approach as shown in Fig. 3.3.1 is taken. The matrix M is divided into vertical stripes and the vector v is divided into horizontal stripes having the following characteristics:
  - The number of stripes in M and the number of stripes in v must be equal,
  - The width of all the stripes in M must be equal,
  - The height of all the stripes in v must be equal.
  - The size of a stripe in v must be such that it can fit conveniently in main memory of a compute node.
- Now it is sufficient to multiply the jth stripe of M with the jth stripe of v. A chunk of a stripe of M and the corresponding entire stripe of v is assigned to each Map task and the calculations proceed as described earlier.

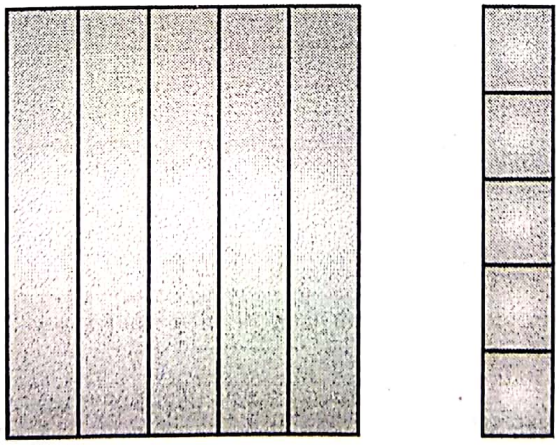

Fig. 3.3.1 : Division of matrix and vector into stripes

3.3.2 Relational-Algebra Operations

Q. What is relational Algebra? Explain different operation performed by Relational algebra.

- A relation is a table. A row of this table is called a tuple and the column headers are called attributes. The schema of a relation is the set of its attributes. A relation is represented by $R(A_1, A_2, \ldots, A_n)$, where R is the name of the relation and $A_1, A_2, \ldots, A_n$ are the attributes of R.
- Relational algebra defines the standard operations that can be performed on relations. In this section we are going to discuss the following relational algebra operations :
  - Selection,
  - Projection,
  - Union,
  - Intersection,
  - Difference.

1. Selection operation

- In case of a selection operation a constraint which is denoted by 'C' is applied on every tuple in the relation.
- Only those tuples which satisfy the specified constraint 'C' will be retrieved and shown by the system as output.
- Selection operation in the relational algebra is represented by $\sigma_C(R)$.

Where, $\sigma \rightarrow$ represents select operation

$C \rightarrow$ represents condition/constraint

$R \rightarrow$ represents the relation.

2. Projection operation

- Consider 'S' to be a subset of columns/attributes for a relation R.
- E.g. If a relation contains a total of10 columns then consider only first 5 columns for processing.
- From all the tuples/rows only the specified attributes/columns are retrieved and shown as output.
- The projection operation is represented in relational algebra as

$$\pi_S (R)$$

Where, $\pi \rightarrow$represents project operation

$S \rightarrow$ represents subset

$R \rightarrow$represents the relation

3. Union, Intersection and difference operations

- All the three operations operate on the rows of two different relations. The basic requirement is that both of these relations must be having the same schema.

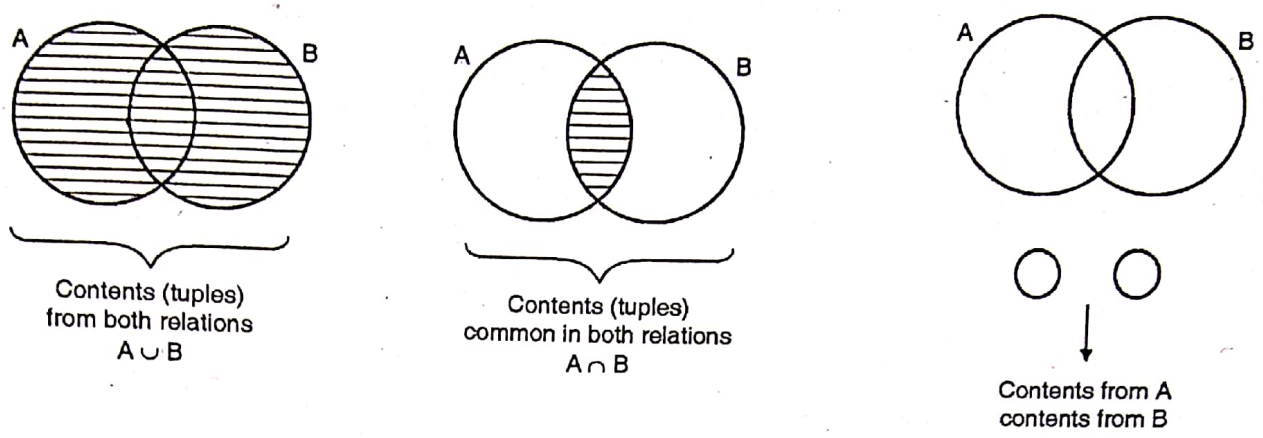

Fig. 3.3.2 : Union, Intersection and difference Venn diagrams

3.3.3 Computing Selections by MapReduce

- MapReduce is way too powerful for selection operations. Selections can be done completely either in the Map phase alone or in the Reduce phase alone. Here we shall discuss the former.

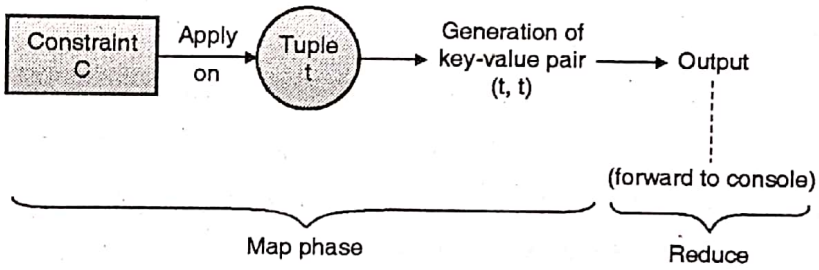

Fig. 3.3.3 : MapReduce Implementation of selection operation

- In the Map task, the constraint C is applied on each tuple t of the relation.
- If C is satisfied by a tuple t, then the key-value pair (t, t) is produced for that particular tuple. Observe that here both the key as well as the value are the tuple t itself.
- If C is not satisfied by t, then the Map function will produce no output for that tuple t.
- As the processing is already finished in the Map function the Reduce function is the identity function. It will simply forward on the key-value pairs to the output for display.
- The output relation is obtained from either the key part or the value part as both are containing the tuple t.

3.3.4 Computing Projections by MapReduce

- In the Map task, from each tuple t in R the attributes not present in S are eliminated and a new tuple t' is constructed. The output of the Map tasks is the key-value pairs (t', t').
- The main job of the Reduce task is to eliminate the duplicate t's as the output of the projection operation cannot have duplicates.

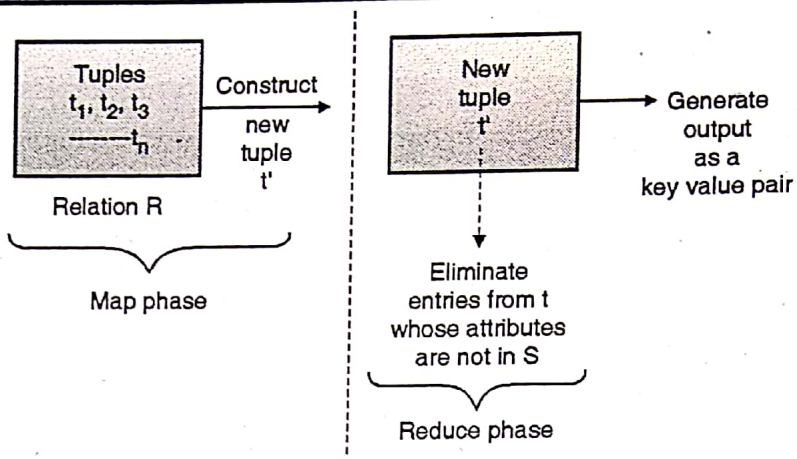

Fig. 3.3.4 : MapReduce implementation of projection operation

3.3.5 Union, Intersection and Difference by MapReduce

Q. Explain Union, Intersection and Difference operation with MapReduce techniques.

Union with MapReduce

- For the union operation R ∪ S, the two relations R and S must have the same schema. Those tuples which are present in either R or S or both must be present in the output.
- The only responsibility of the Map phase is of converting each tuple t into the key-value pair (t, t).
- The Reduce phase eliminates the duplicates just as in the case of projection operation. Here for a key t there can be either 1 value if it is present in only one of the relations or t can have 2 values if it is present in both the relations. In either case the output produced by the Reduce task will be (t, t).

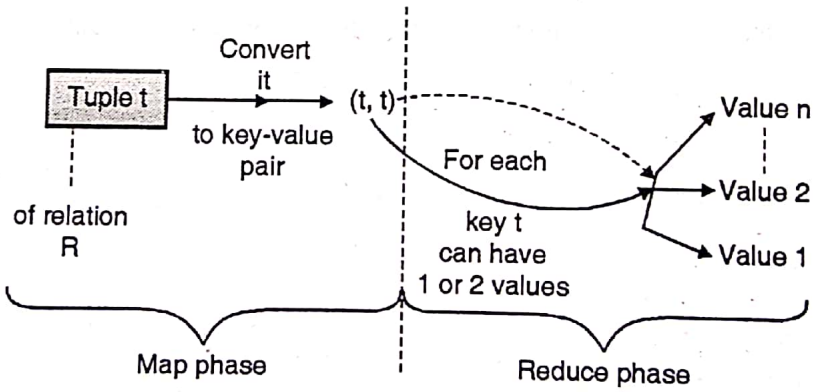

Fig. 3.3.5 : Union operation with MapReduce

Intersection with MapReduce

- For the intersection operation R ∩ S, both the relations R and S must have the same schema. Only those tuples which are present in both R and S should be present in the output.
- The responsibility of the Map phase is same as that of union operation i.e. conversion of a tuple 't' in a given relation 'R' into the key-value pair format (t, t).
- Reducer will produce the output (t, t) only if both R and S have the tuple t. This can be done by checking the number of values associated with the key t. If the key t has a list of two values [t, t], then the Reduce task will produce the output (t, t). If the key t has only one value [t], then the Reducer will produce nothing as output.

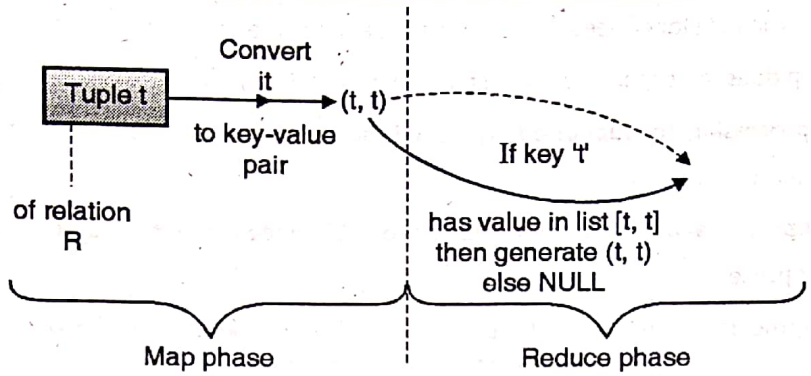

Fig. 3.3.6 : Intersection operation with MapReduce

Difference with MapReduce

- For the difference operation R – S, both the relations R and S must have the same schema. The tuples which are present only in R and not in S will be present in the output.
- The Map phase will produce the key-value pairs (t, R) for every tuple in R and (t, S) for every tuple in S.
- The Reduce phase will produce the output (t, t) only if the associated value of a key t is [R].

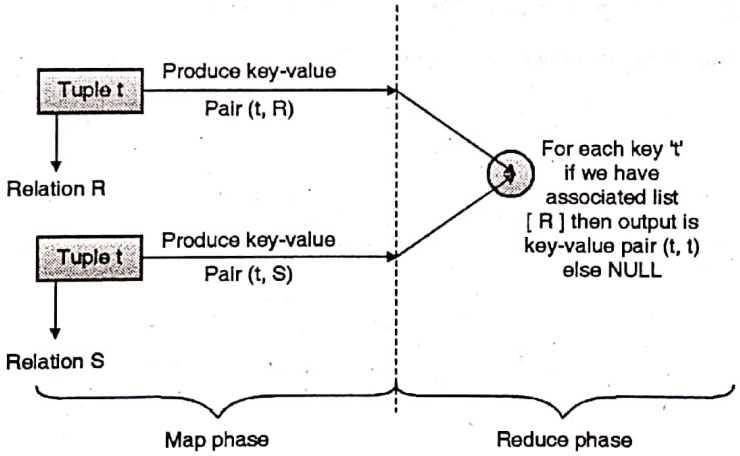

Fig. 3.3.7 : Difference operation with MapReduce

3.4 Hadoop Limitations

- Hadoop is a collection of open source projects created by Doug Cutting and Mike Cafarella in 2006. It was inspired by Google's MapReduce programming framework. Hadoop consist of the following core modules :
  - Hadoop Distributed File System (HDFS) which is the storage module and
  - MapReduce programming model which is the processing module.
- The various limitations of Hadoop are :
  - **Not fit for small files :** Hadoop was designed to work with big sized files and it cannot efficiently handle small files even if there are a huge number of small files. A file is considered small if its size is less than the HDFS block size which by default is 128 MB.
  - **Slow speed :** Hadoop works with massive amounts of distributed data which slows down the processing.

- **Not easy to use :** The developer needs to write the code for all the operations which makes it very difficult to use.
- **Security :** Hadoop does not support encryption which makes it vulnerable.
- **Real-time data processing not supported :** Hadoop is designed to support only batch processing and hence real-time processing feature is missing.
- **No iteration support :** Hadoop is not designed to support the feeding of the output of one stage to the input of the next stage of processing.
- **No caching :** Intermediate results are not cached and this brings down the performance.

Review Questions

Q. 1 Explain the large-scale file-system organization in detail.

Q. 2 Explain MapReduce framework with suitable example.

Q. 3 What are the different phases involved in MapReduce technique ? Explain with example.

Q. 4 What are combiners ? Explain the position and significance of combiners.

Q. 5 What is Relational Algebra ? Explain the different operations performed by Relational Algebra.

Q. 6 Explain selection and projection operation using MapReduce functionality.

Q. 7 Explain Union, Intersection and Difference operations with MapReduce techniques.