

BIG DATA AND ANALYTICS
(Effective from the academic year 2018 -2019)
SEMESTER – VII

Course Code	18CS72	CIE Marks	40
Number of Contact Hours/Week	4:0:0	SEE Marks	60
Total Number of Contact Hours	50	Exam Hours	03

CREDITS –4

Course Learning Objectives: This course (18CS72) will enable students to:

- Understand fundamentals of Big Data analytics
- Explore the Hadoop framework and Hadoop Distributed File system
- Illustrate the concepts of NoSQL using MongoDB and Cassandra for Big Data
- Employ MapReduce programming model to process the big data
- Understand various machine learning algorithms for Big Data Analytics, Web Mining and Social Network Analysis.

Module 1	Contact Hours
Introduction to Big Data Analytics: Big Data, Scalability and Parallel Processing, Designing Data Architecture, Data Sources, Quality, Pre-Processing and Storing, Data Storage and Analysis, Big Data Analytics Applications and Case Studies. Text book 1: Chapter 1: 1.2 -1.7 RBT: L1, L2, L3	10
Module 2	
Introduction to Hadoop (T1): Introduction, Hadoop and its Ecosystem, Hadoop Distributed File System, MapReduce Framework and Programming Model, Hadoop Yarn, Hadoop Ecosystem Tools. Hadoop Distributed File System Basics (T2): HDFS Design Features, Components, HDFS User Commands. Essential Hadoop Tools (T2): Using Apache Pig, Hive, Sqoop, Flume, Oozie, HBase. Text book 1: Chapter 2 :2.1-2.6 Text Book 2: Chapter 3 Text Book 2: Chapter 7 (except walk throughs) RBT: L1, L2, L3	10
Module 3	
NoSQL Big Data Management, MongoDB and Cassandra: Introduction, NoSQL Data Store, NoSQL Data Architecture Patterns, NoSQL to Manage Big Data, Shared-Nothing Architecture for Big Data Tasks, MongoDB, Databases, Cassandra Databases. Text book 1: Chapter 3: 3.1-3.7 RBT: L1, L2, L3	10
Module 4	
MapReduce, Hive and Pig: Introduction, MapReduce Map Tasks, Reduce Tasks and MapReduce Execution, Composing MapReduce for Calculations and Algorithms, Hive, HiveQL, Pig. Text book 1: Chapter 4: 4.1-4.6 RBT: L1, L2, L3	10

Module 5	
<p>Machine Learning Algorithms for Big Data Analytics: Introduction, Estimating the relationships, Outliers, Variances, Probability Distributions, and Correlations, Regression analysis, Finding Similar Items, Similarity of Sets and Collaborative Filtering, Frequent Itemsets and Association Rule Mining.</p> <p>Text, Web Content, Link, and Social Network Analytics: Introduction, Text mining, Web Mining, Web Content and Web Usage Analytics, Page Rank, Structure of Web and analyzing a Web Graph, Social Network as Graphs and Social Network Analytics:</p> <p>Text book 1: Chapter 6: 6.1 to 6.5</p> <p>Text book 1: Chapter 9: 9.1 to 9.5</p>	10
<p>Course Outcomes: The student will be able to:</p> <ul style="list-style-type: none"> • Understand fundamentals of Big Data analytics. • Investigate Hadoop framework and Hadoop Distributed File system. • Illustrate the concepts of NoSQL using MongoDB and Cassandra for Big Data. • Demonstrate the MapReduce programming model to process the big data along with Hadoop tools. • Use Machine Learning algorithms for real world big data. • Analyze web contents and Social Networks to provide analytics with relevant visualization tools. 	
<p>Question Paper Pattern:</p> <ul style="list-style-type: none"> • The question paper will have ten questions. • Each full Question consisting of 20 marks • There will be 2 full questions (with a maximum of four sub questions) from each module. • Each full question will have sub questions covering all the topics under a module. • The students will have to answer 5 full questions, selecting one full question from each module. 	
<p>Textbooks:</p> <ol style="list-style-type: none"> 1. Raj Kamal and Preeti Saxena, "Big Data Analytics Introduction to Hadoop, Spark, and Machine-Learning", McGraw Hill Education, 2018 ISBN: 9789353164966, 9353164966 2. Douglas Eadline, "Hadoop 2 Quick-Start Guide: Learn the Essentials of Big Data Computing in the Apache Hadoop 2 Ecosystem", 1stEdition, Pearson Education, 2016. ISBN-13: 978-9332570351 	
<p>Reference Books:</p> <ol style="list-style-type: none"> 1. Tom White, "Hadoop: The Definitive Guide", 4th Edition, O'Reilly Media, 2015. ISBN-13: 978-9352130672 2. Boris Lublinsky, Kevin T Smith, Alexey Yakubovich, "Professional Hadoop Solutions", 1stEdition, Wrox Press, 2014 ISBN-13: 978-8126551071 3. Eric Sammer, "Hadoop Operations: A Guide for Developers and Administrators", 1stEdition, O'Reilly Media, 2012. ISBN-13: 978-9350239261 4. Arshdeep Bahga, Vijay Madisetti, "Big Data Analytics: A Hands-On Approach", 1st Edition, VPT Publications, 2018. ISBN-13: 978-0996025577 	

Module 1

Introduction to Big Data Analytics

Contents

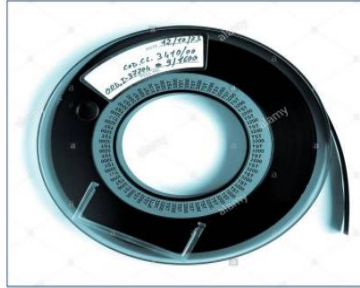
- Big Data
- Scalability and Parallel Processing
- Designing Data Architecture
- Data Sources
- Quality
- Pre-Processing and Storing
- Data Storage and Analysis
- Big Data Analytics Applications and Case Studies

Introduction

History of data storage



Ledger Books



Magnetic tapes(230 KB)



Floppy Disk(1.44 MB)



11/4/2021
4 November 2021

Compact Disk(700 MB)



Padmavathi H G – 18CS72
Mrs. Padmavathi H G – 18CS72

Digital Video Disk(1.4 GB to 17 GB)

5
5



SD Card(64 MB to 1 TB)



USB Flash Drive(8 MB to 2 TB)



Blue Ray Disk(50 GB)



Cloud Data Storage

11/4/2021

Padmavathi H G – 18CS72

6

The amount of data in today's Digital Universe is

2,700,000,000,000,000,000 KB

Approximately 2.7 zettabytes of digital data exist today.

8 bits = 1Byte

1024 bytes = 1KB

1024 KB = 1MB

1024MB = 1GB

1024GB = 1TB

1000 (1024) terabytes = 1 petabyte

1000 petabytes = 1 exabyte

1000 exabytes = 1 zettabyte

30 billion 4K movies can be stored in 1 zettabyte.

90% of 2.7 zettabytes of data has been created in the last 10 to 15 years.

How can so much of data get created?



11/4/2021

8

Approximately 40 exabytes of data gets generated by a single smart phone every month.

- There are approximately 6000000000 smart phones users across the world.
- Monthly generation of world-wide data is,
40 exabytes x 6000000000

BIG DATA

Data generated on the internet per minute



2.1 million snaps shared



3.8 million searches



1 million logins



4.5 million videos watched



188 million emails sent

11/4/2021

Padmavathi H G – 18CS72

10

Definitions of Data

“Data is information, usually in the form of facts or statistics that one can analyze or use for further calculations.”

“Data is information that can be stored and used by a computer program.”

“Data is information presented in numbers, letters, or other form.”

“Data is information from series of observations, measurements or facts.”

“Data is information from series of behavioral observations, measurements or facts.”

Definition of Web

“Web is large scale integration and presence of data on web servers.”

Is Web and Internet the same???

- Web is a part of the Internet that stores web data in the form of documents and other web resources.



Definition of Web Data

“Web Data is the data present on web servers in the form of text, images, videos, audios and multimedia files for web users.”

• Examples of Web data



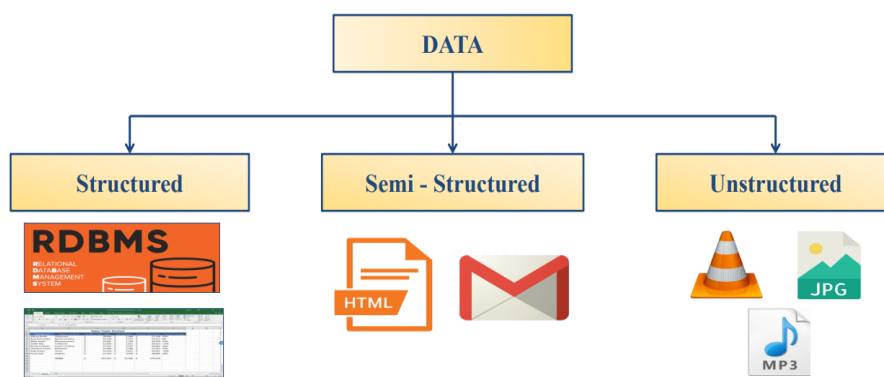
11/4/2021



Padmavathi H G – 18CS72

14

Classification of Data



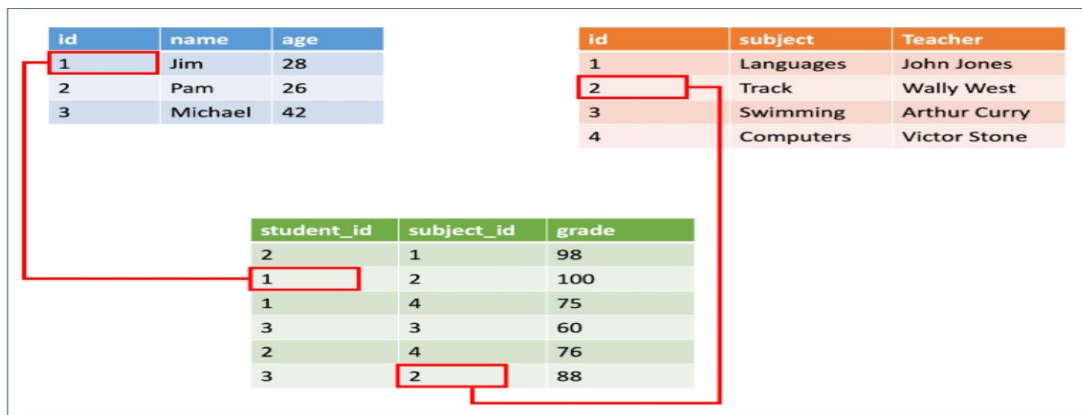
11/4/2021

Padmavathi H G – 18CS72

15

Structured Data

- Structured data conform and associate with data schemas and data models.
- Structured data are found in tables (rows and columns).
- Databases that hold tables in this form are called Relational Databases.
- In structured data, all row in a table has the same set of columns.
- SQL (Structured Query Language) programming language used for structured data.



Structured data enables the following

- Data insert, delete, update and append.
- Indexing to enable faster data retrieval.
- Scalability which enables increasing or decreasing capacities and data processing operations such as storing, processing and analytics.
- Transaction's processing which follows ACID rules (Atomicity, Consistency, Isolation and Durability).
- Encryption and Decryption for data security.

Semi-Structured Data

- Semi-structured data contain tags and other markers.
- The tags are used to separate semantic elements thereby creating a hierarchy.
- Here the data does not conform and associate with formal data model structures.
- It does not confine into a rigid structure such as that needed for relational databases.
- Examples of semi-structured data are XML and JSON documents.

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Note

To: Tove

From: Jani

Reminder

Don't forget me this weekend!

*An XML file and its
corresponding output*

Unstructured Data

- Unstructured data is information that either does not organize in a pre-defined manner or not

have a pre-defined data model.

- It is a data that is present in absolute raw form.
- This type of data does not possess data features such as a table or a database.
- The data here is text-heavy but may also contain numbers, dates, and facts as well.
- Word, PDF, Videos and audio files are classical examples of unstructured data.

Some examples of unstructured data

- Mobile data: Text messages, chat messages, tweets, blogs and comments.
- Website content data: YouTube videos, browsing data, e-payments, user-generated maps.
- Social media data: Images and videos from Instagram, Facebook, LinkedIn, Flickr (upload, access, organize, edit and share photos from any device from anywhere in the world).
- Satellite images, atmospheric data, surveillance, traffic videos.

Big Data Definitions

“Big Data is high volume, high velocity and/or high-variety information asset that

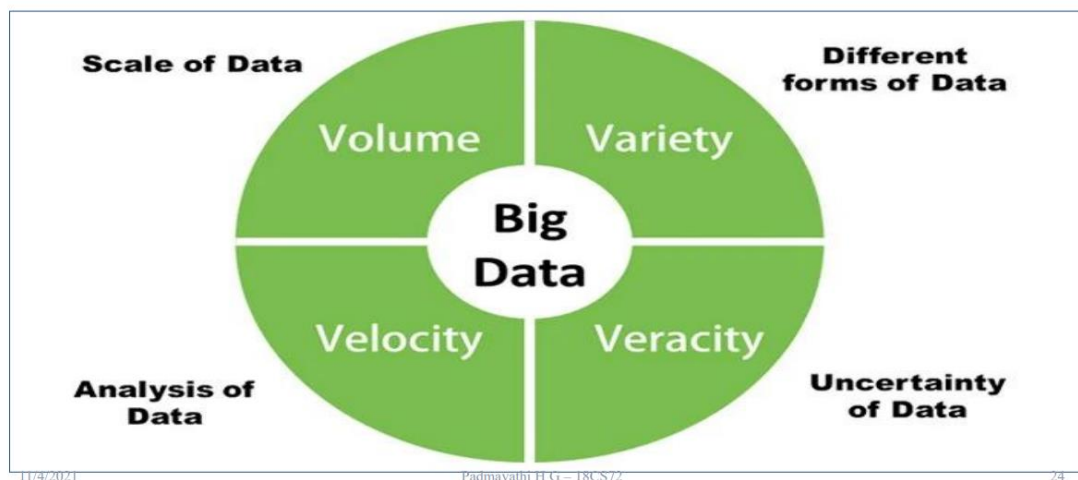
requires new forms of processing for enhanced decision making, insight discovery and process optimization.”

“Big Data is a collection of data sets so large or complex that traditional data processing applications are inadequate.”

“Big Data is data of a very large size, typically to the extent that its manipulation and management present significant logistical challenges.”

"Big Data refers to data sets whose size is beyond the ability of typical database software tool to capture, store, manage and analyze."

Big Data Characteristics



Big Data Characteristics



11/4/2021

Padmavathi H G – 18CS72

Volume

- Volume defines the amount or quantity of data, which is generated from various applications.
- This data is generated from different sources such as IoT devices, social media, videos, financial transactions and customer logs.
- Volume determines the processing considerations needed for handling that data.
- Currently distributed systems are used to store data in several locations and brought together by frameworks like Hadoop, Storm, Hive and Spark.

Velocity

- The term Velocity refers to the speed of generation of data.
- Velocity is a measure of how fast the data is generated and processed.

- To meet the demands and the challenges of processing Big Data, the velocity of generation of data plays a crucial role.
- This is because only after analysis and processing, the data can meet the demands of the clients/users.
- There is no point in making huge investments if we end up waiting for data.

Variety

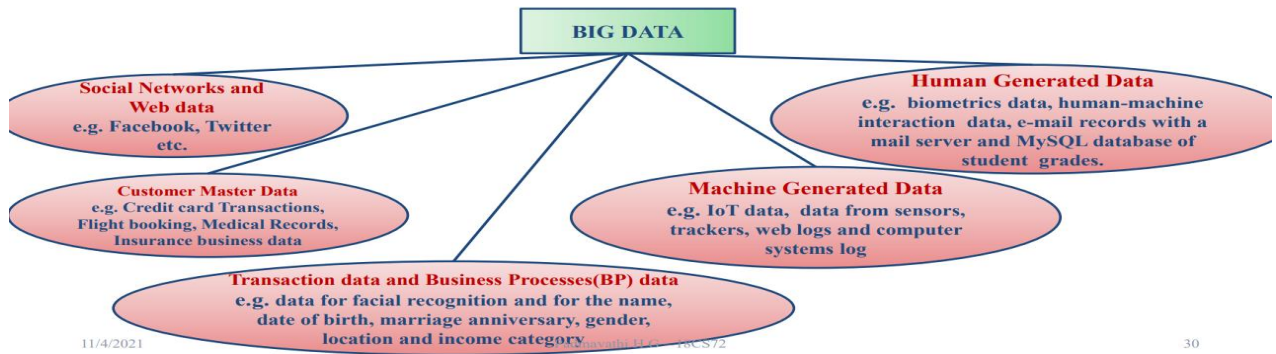
- One of the most important Big Data characteristics is its variety.
- It refers to the different sources of data and their nature.
- Earlier, it was only available in spread sheets and databases.
- Nowadays, data is present in photos, audio files, videos, text files, and PDFs.
- Out of all the data available, 80% comprises of unstructured data while the rest is structured and semi structured.

Veracity

- Veracity refers to the quality of data captured.
- It is a measure of how accurate or truthful a data set may be.
- The quality of data is dependent on certain factors such as
 - Where the data has been collected from
 - How it was collected
 - How it will be analysed.
- Low veracity data, usually contains a high percentage of non-valuable, „noisy“ and meaningless data.

Big Data Types

The following classification is developed by a team from IBM



Big Data Classification

Various classification methods for traditional data

Basis of Classification	Examples
Data sources	<ul style="list-style-type: none"> Data storage such as records RDBMs Distributed databases Row-oriented In- memory data tables Column-oriented In-memory data tables Data warehouse
Data formats	<ul style="list-style-type: none"> Structured Semi-structured
Data Stores structure	<ul style="list-style-type: none"> Enterprise or cloud servers Data warehouse Row-oriented data for OLTP Column-oriented for OLAP Records Graph databases Hashed entries for key/value pairs

Basis of Classification	Examples
Processing data rates	<ul style="list-style-type: none"> Batch Near-time Real-time

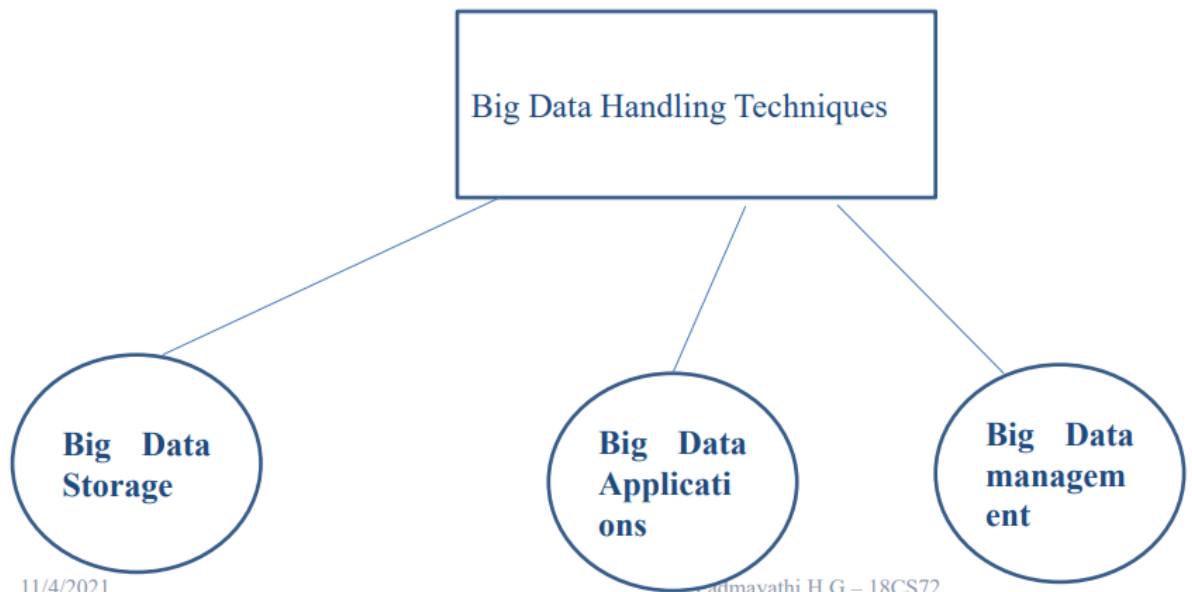
Various classification methods for Big data

Basis of Classification	Examples
Big Data sources	<ul style="list-style-type: none"> Distributed file system Operational Data Store (ODS) Data warehouse NoSQL database (MongoDB, Cassandra) Sensors data External data such as web, social media, weather data, health records
Big Data formats	<ul style="list-style-type: none"> Unstructured Semi-structured Multistructured

Basis of Classification	Examples
Processing Big data rates	<ul style="list-style-type: none"> • High volume, velocity, variety and veracity • Batch • Near-time • Real-time, Streaming
Big Data processing methods	<ul style="list-style-type: none"> • Batch processing (using MapReduce, Hive or Pig), • Real-time processing (using SparkStreaming, SparkSQL, Apache Drill)
Big Data analysis methods	<ul style="list-style-type: none"> • Statistical analysis • Predictive analysis • Regression analysis • Machine learning algorithms • Clustering algorithms • Classifiers • Social network analysis • Location-based analysis • Diagnostic analysis

Big Data Handling Techniques

Following are the techniques deployed for Big Data storage, applications, data management and mining and analytics.



Big Data Handling Techniques

Big Data Storage	<ul style="list-style-type: none">• Huge data volumes storage• Data distribution• High-speed networks• High performance computing
Big Data Applications	<ul style="list-style-type: none">• Applications which are open source, reliable, scalable, distributed file system, distributed database, parallel and distributed computing systems such as <i>Hadoop</i> or <i>Spark</i>.
Big Data management	<ul style="list-style-type: none">• NoSQL• Column-oriented database• Graph database• Other form of databases used as per needs of the applications

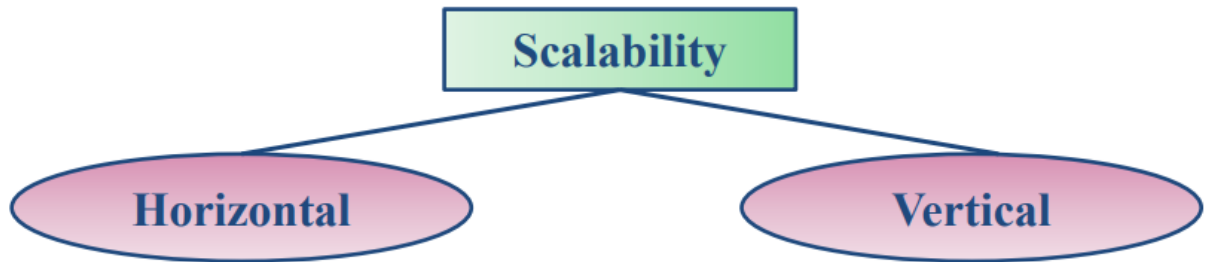
SCALABILITY AND PARALLEL PROCESSING

- Traditional data stores use RDBMS tables or data warehouse to store and manage data.
- Big Data needs processing of larger data volume and therefore needs intensive computations.
- Processing complex applications with large datasets (TB to PB datasets) need hundreds of computing nodes.
- Therefore, Big Data processing and analytics requires scaling up of computing resources.
- Scalability enables increase or decrease in the capacity of data storage, processing and analytics.

Analytics Scalability to Big Data

- The Scalability of an application can be measured by the number of requests or tasks it can effectively support simultaneously.
- The point at which an application can no longer handle additional requests effectively is the limit of its scalability.
- This limit is reached when a critical hardware resource runs out, requiring different or more machines.
- Scaling these resources can include any combination of adjustments to CPU and physical memory, hard disk and/or the network bandwidth.

SCALABILITY AND PARALLEL PROCESSING

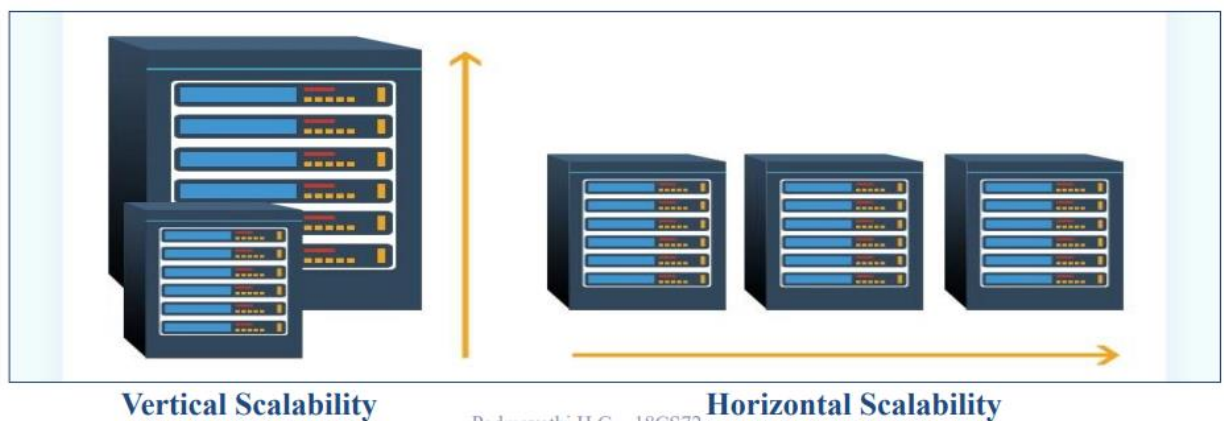


Horizontal Scalability

- Horizontal scalability means increasing the number of systems working in coherence and scaling out the workload.
- It is also referred to as Scaling out.
- Scaling out means using more resources and distributing the processing and storage tasks in parallel.

Vertical Scalability

- Vertical scalability means scaling up the given system's resources and increasing the system's analytics, reporting and visualization capabilities.
- It is also referred to as Scaling up.



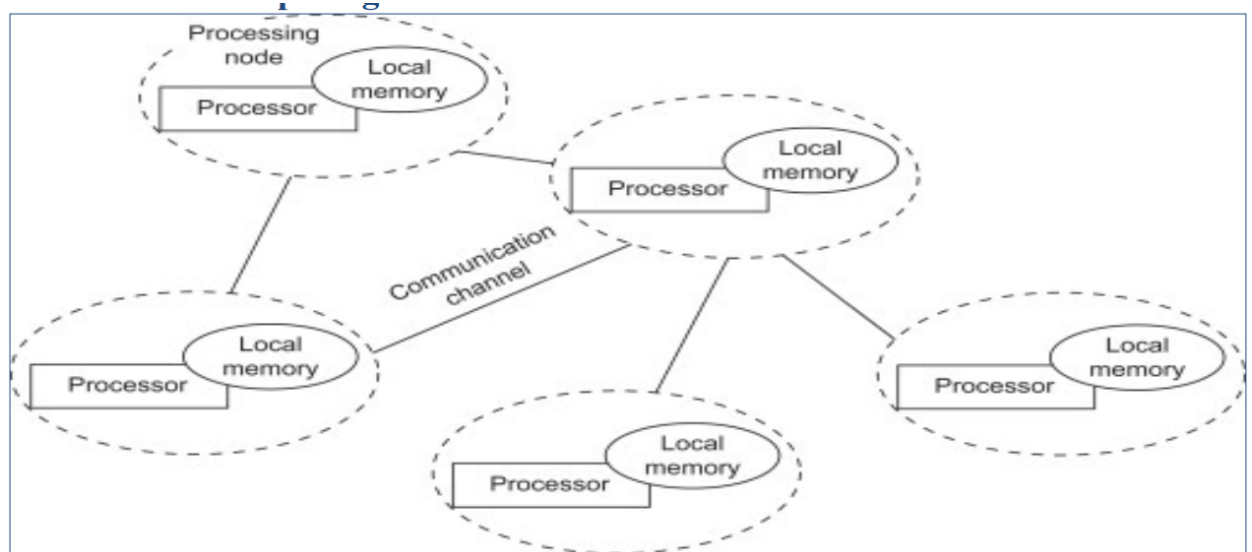
Padmavathi H G – 18CS72

Scaling up and scaling out are definitely beneficial for carrying out analytics.

- However, buying faster CPUs, bigger and faster RAM modules, hard disks and motherboards will be expensive.

- Also, if more CPUs add in a computer, but the software does not exploit the advantage of them, then this results in wastage of resources.
- We next discuss alternative ways for scaling up and out processing of analytics software and Massively Parallel Processing Platforms
- Scaling uses parallel processing systems.
- It is impractical or impossible to efficiently execute programs that are large and complex on a single computer with limited memory.
- Such scenarios require use of Massive Parallel Processing(MPPs) platforms.
- Parallelization of tasks can be done at several levels:
 - Distributing separate tasks onto separate threads on the same CPU.
 - Distributing separate tasks onto separate CPUs on the same computer.
 - Distributing separate tasks onto separate computers.
- When designing an algorithm or problem, we need to draw the advantage of availability of **multiple computing systems**.
- Multiple compute resources are used in parallel processing systems.
- The computational problem is broken into discrete pieces of sub-tasks that can be processed simultaneously.
- The system executes multiple program instructions or sub-tasks at any moment in time.
- Total time taken will be much less than with a single compute resource.

Distributed Computing Model

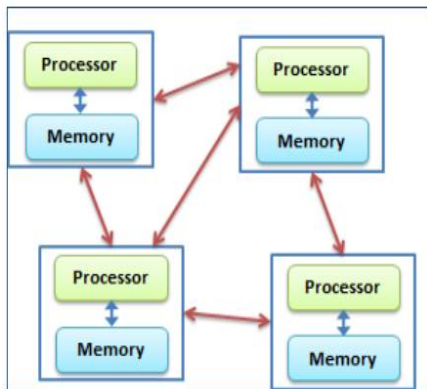


At a basic level, a Distributed System is a collection of computers that work together to form a single computer for the end-user.

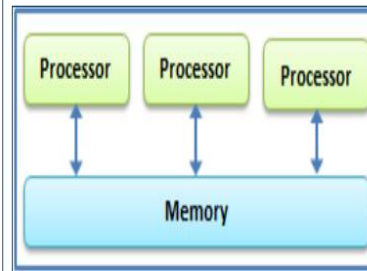
- The computers that are in a distributed system can be physically close together and connected by a local network, or they can be geographically distant and connected by a wide area network.
- These connected systems are able to fail independently without damaging the whole system. • Hence a distributed system can be thought of as a set of interdependent and autonomous computers or nodes.

• **Processing Big Data requires,**

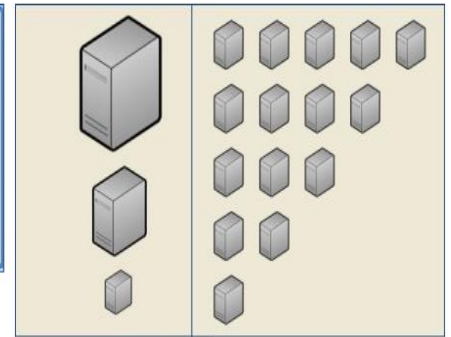
- Distributed computing
- Parallel computing
- Scalable computing



Distributed Computing



Parallel Computing



Scalable Computing

- A Distributed Computing model uses **Table 1.2** Distributed computing paradigms

- *Cloud*
- *Grid*
- *Clusters*

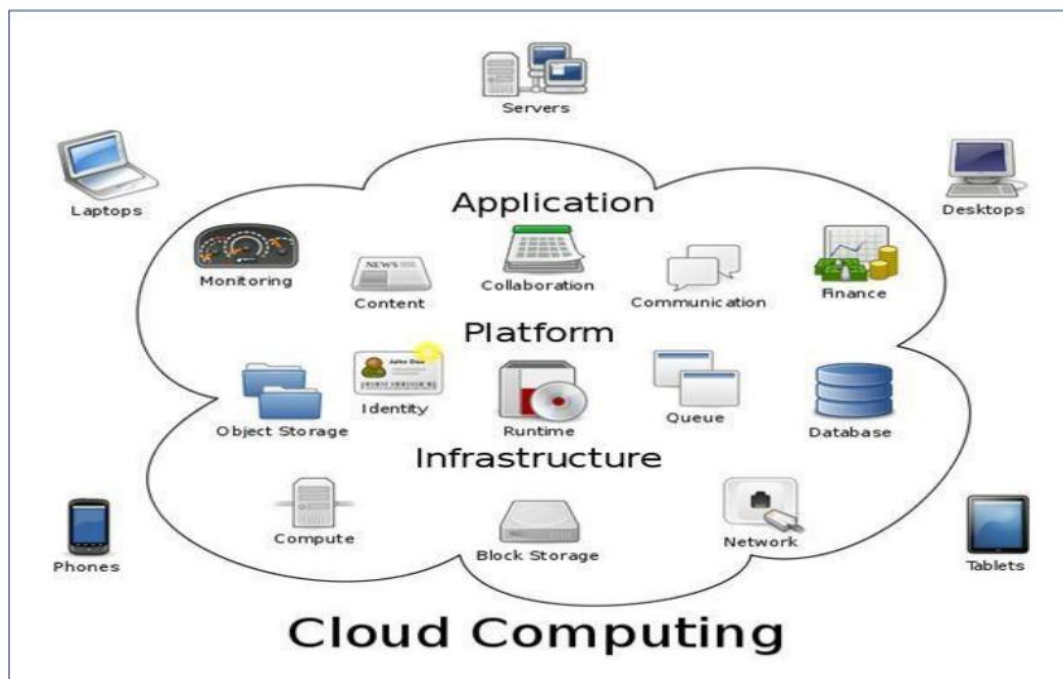
Distributed computing on multiple processing nodes/clusters	Big Data > 10 M	Large datasets below 10 M	Small to medium datasets up to 1 M
Distributed computing	Yes	Yes	No
Parallel computing	Yes	Yes	No
Scalable computing	Yes	Yes	No
Shared nothing (No in-between data sharing and inter-processor communication)	Yes	Limited sharing	No
Shared in-between between the distributed nodes/clusters	No	Limited sharing	Yes

11/4/2021

Dr. J. ... 4/11/2021 10:00:00

46

Cloud Computing



Dr. J. ... 4/11/2021 10:00:00

Wikipedia defines cloud computing as “Cloud computing is a type of Internet-based computing that provides shared processing resources and data to the computers and other devices on demand.”

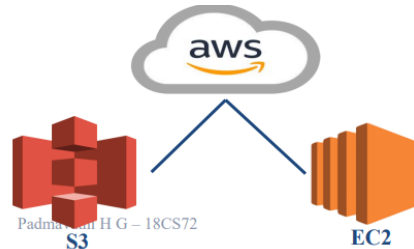
- In the simplest terms, cloud computing means storing and accessing data and programs over

the internet instead of your computer's hard drive.

- Some examples of cloud resources are



11/4/2021



48

Some of the important features of cloud computing are:

- On-demand service

- Resource pooling

- Scalability

- Accountability

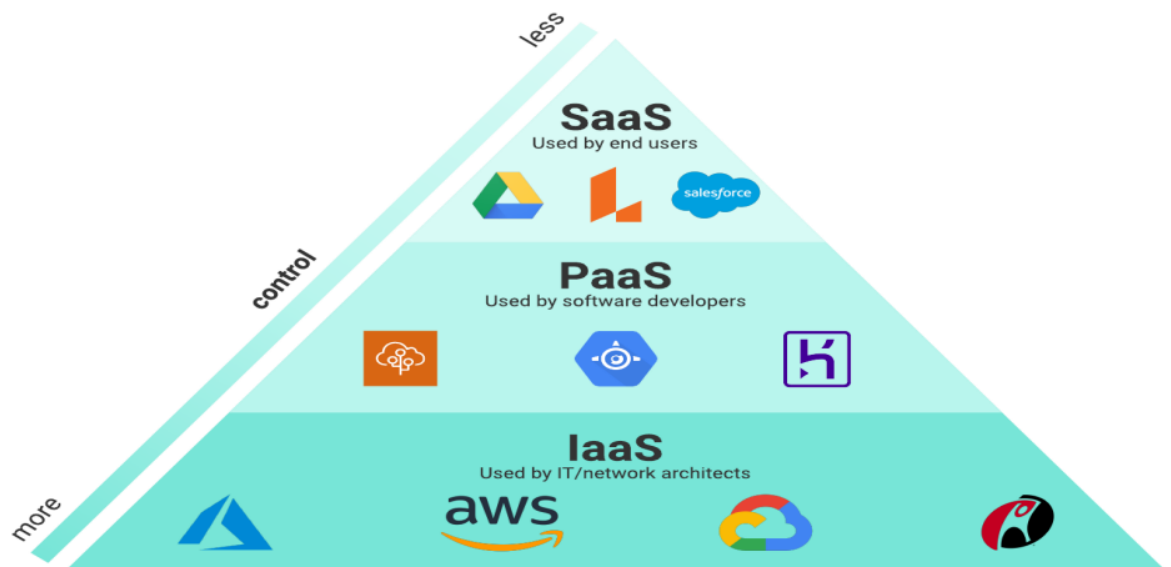
- Broad network access.

- Cloud services can be accessed from Anywhere and at Anytime through the Internet.

- A local private cloud can also be set up on a local cluster of computers.

- Development platform, Database, Software applications are some computing infrastructures provided by cloud computing.

- Cloud computing services can be classified into three fundamental types.



Software as a Service (SaaS)

- Providing software applications as a service to end users is known as Software as a Service.
- Software applications are hosted by a service provider and made available to customers over the Internet.
- In this model, an Independent Software Vendor (ISV) may contract a third-party cloud provider to host the application.
- With larger companies, such as Microsoft, the cloud provider might also be the software vendor.
- Google workspace, Microsoft Office 365, Adobe Creative Cloud are some popular examples of SaaS.

Platform as a Service (PaaS)

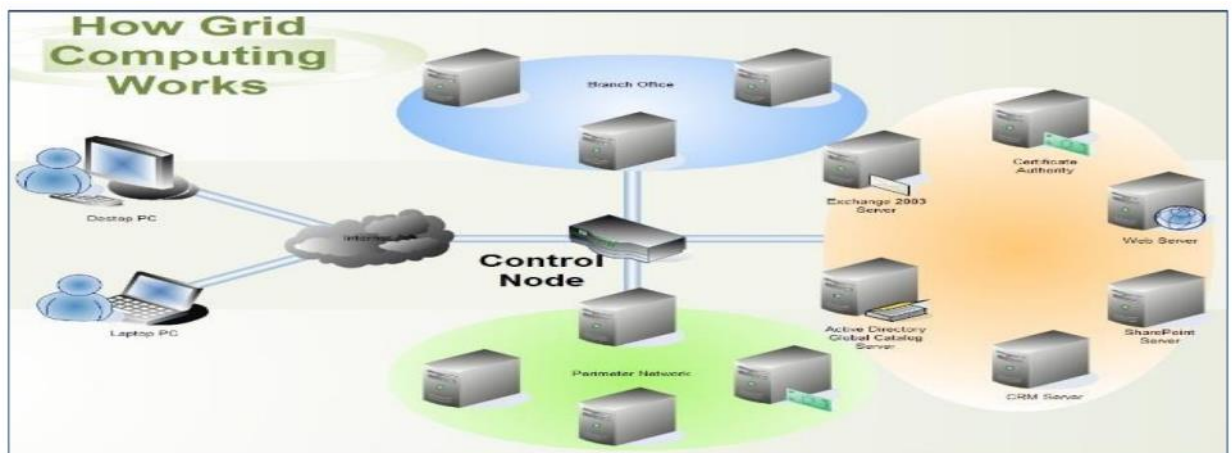
- It implies providing the runtime environment to allow developers to build applications and services.
- Computing platforms which typically includes operating system, programming language execution environment, database, web server is provided.
- Using PaaS users can build, compile and run their programs without worrying about the underlying infrastructure.
- Google App Engine, AWS Elastic Beanstalk, Heroku, AWS Elastic Beanstalk are popular examples of PaaS.

Infrastructure as a Service (IaaS)

- Providing access to resources, such as hard disks, network connections, databases storage, data center and virtual server spaces is Infrastructure as a Service (IaaS).
- In simple terms, IaaS is the equivalent to servers and hardware that we might have in our workplace.
- IaaS enables end users to scale and shrink resources on the basis of demand.
- This helps in reducing the need for high, up-front capital expenditures or unnecessary “owned” infrastructure.
- Amazon Web Services (AWS), Cisco Meta cloud, Microsoft Azure, Google Compute Engine (GCE)

Grid Computing

- Grid Computing refers to a group of computing resources from several locations that are connected with each other to achieve a common task.
- Alternately Grid computing is defined as a group of networked computers which work together to perform large tasks, such as analyzing huge sets of data
- Grid is a group of computers that might spread over remotely.



Grid Computing is a subset of Distributed computing.

- The nodes in the grid consist of machines with different platforms running on various different OS.
- A Grid computing network mainly consists of these three types of machines
 - Control Node

A computer, usually a server or a group of servers which administrates the whole network and keeps the

account of the resources in the network pool.

– Provider

The computer which contributes it's resources in the network resource pool.

– User

The computer that uses the resources on the network.

Features of Grid Computing

- Large scale: a grid must be able to deal with a number of resources ranging from just a few to millions.
- Geographical distribution: grid's resources may be located at distant places.
- Heterogeneity: a grid hosts both software and hardware resources that can be very varied.
- Resource sharing and coordination: resources in a grid must be coordinated in order to provide aggregated computing capabilities.
- Transparent access: a grid should be seen as a single virtual computer.
- Dependable access: a grid must assure the delivery of services under established Quality of Service (QoS) requirements.

Advantages of Grid Computing

- Makes better use of existing resources.
- Grid environments don't have single points of failure. If one of the nodes within the grid fail there are plenty of other resources able to pick the load.
- Jobs can be executed in parallel speeding performance.
- Can solve larger, more complex problems in a shorter time.

Cluster Computing

- A cluster is a group of computers connected by a network.

- The group works together to accomplish the same task.
- Clusters are used mainly for load balancing.
- They shift processes between nodes to keep an even load on the group of connected computers.
- All the nodes connected in the cluster must have the same configuration.

Cloud vs Grid vs Cluster

	Cloud	Grid	Cluster
Connectivity	WAN	WAN	LAN
Dependency	Loosely coupled	Loosely coupled	Tightly coupled

Configuration of nodes	Heterogeneous	Heterogeneous	Homogeneous
Resource Management	Decentralized	Distributed	Centralized
Autonomy	Every node is autonomous	Every node is autonomous	Whole cluster functions as a single system.
Security	Lower than Grid and Cluster	Lower than Cluster	High

Volunteer Computing

- Volunteer computing is a distributed computing paradigm which uses computing resources of the volunteers.
- Volunteers are organizations or members who own personal computers.
- Volunteer computing is mainly used to help achieve scientific and medical research at minimal cost.
- Some issues with volunteer computing are
 - Volunteered computers heterogeneity.
 - Their sporadic availability.
 - Incorrect results at volunteers are unaccountable as they are essentially from anonymous volunteers.

Data Architecture Design

"Big Data architecture is the logical and/ or physical layout/structure of how Big Data will be stored, accessed and managed within a Big Data or IT environment. Architecture logically defines how Big Data solution will work, the core components (hardware, database,

software, storage) used, flow of information, security and more."

- Big Data architecture design uses a logical layers approach.
- There are 5 layers in a Big Data architecture design which satisfy the fundamental Big Data characteristics.

DESIGNING DATA ARCHITECTURE



The layers of Big data architecture are:

- Identification of data sources.
- Acquisition, ingestion, extraction, pre-processing, transformation of data.
- Data storage at files, servers, cluster or cloud.
- Data-processing.

(v) Data consumption in the number of programs and tools.

- Data ingestion, pre-processing, storage and analytics require special tools and technologies.

- Data is consumed for applications like data mining, AI, ML, text analytics, descriptive and

predictive analytics etc.

- Logical layer 1 (L1) is for identifying data sources, which are external, internal or both.

- Layer 2 (L2) is for data-ingestion which is a process of absorbing information.

- Ingestion is the process of obtaining and importing data for immediate use or transfer.

- Ingestion may be in batches or in real time.

- Layer L3 is for storage of data from the L2 layer.

- Layer L4 is for data processing using software, such as MapReduce, Hive, Pig or Spark.

- The top Layer L5 is for data consumption.

- Data is consumed for analytics, visualizations, reporting, export to cloud or web servers.

- L1 considers the following aspects in a design:

- Amount of data needed at ingestion layer (L2).

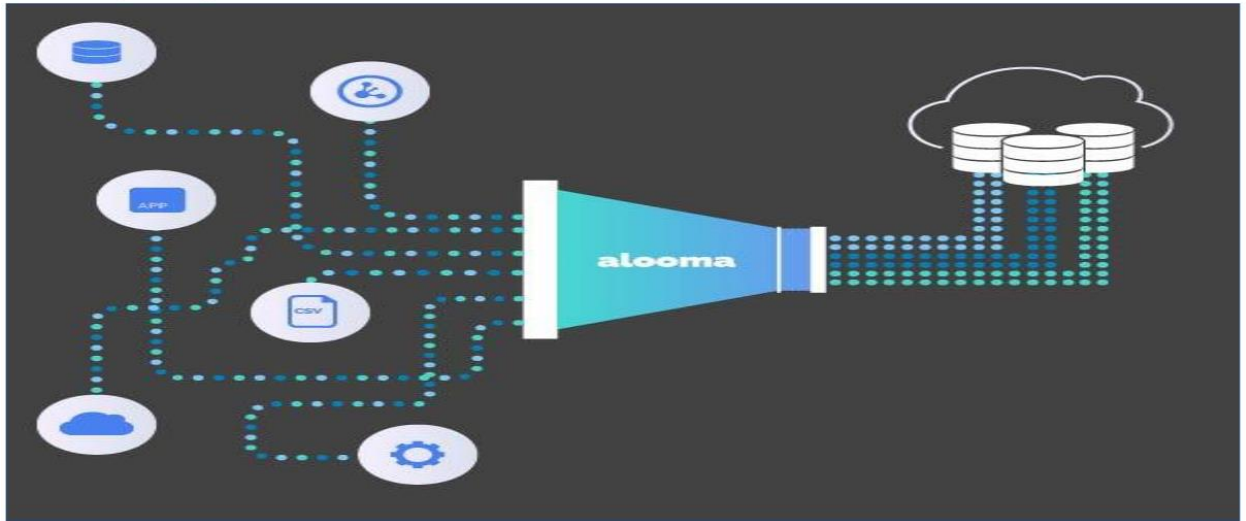
- Push from L1 or pull by L2 as per the mechanism for the usages.

- Source data-types: Database, files, web.

- Source formats: Semi-structured, unstructured or structured.

- L2 which is the Ingestion Layer is the first step for the data coming from variable sources to

start its journey



Data ingestion is a process by which data is moved from one or more sources to a destination

where it can be stored and further analyzed.

- The data might be in different formats and come from various sources, including RDBMS,

other types of databases, S3 buckets, CSVs, or from streams.

- Since the data comes from different places, it needs to be cleansed and transformed in a way

that allows you to analyse it together with data from other sources.

- Otherwise, your data is like a bunch of puzzle pieces that don't fit together.
- We can ingest data in real time or in batches.
- When ingested in batches, data is imported at regularly scheduled intervals.
- This can be very useful when we have processes that run on a schedule, such as reports that

run daily at a specific time.

- Real-time ingestion is useful when the information required is very time-sensitive.
- Data from a power grid that must be monitored moment to moment is an example.
- L3 is the Data Storage layer where the ingested data is stored for further processing.
- This is where Big Data lives, once it is gathered from various sources.
- As the volume of data generated and stored by companies start to explode, measures need to

be taken to manage this huge volume of data.

- Tools like Apache Hadoop DFS (Distributed File System) or Google File System are some

tools available for this.

- A computer with a big hard disk might be all that is needed for smaller data sets.
- But when we start to deal with storing and analyzing truly big data, a more sophisticated,

distributed system is called for.

- Such a system must be able to
 - Store data that the computer system will understand – File system.
 - Organize and categorize data in a way that people will understand – Database.
- Examples of such storage systems are



L4 is the Data Processing layer where the data stored in the repository is subjected to analytics for the first time.

- When we want to use the data, we have stored to find out something useful, we will need to

process and analyses it.

- Essentially, processing involves selecting the elements of the data that we want to analyze and

putting it into a format from which insights can be extracted.

- Automated pattern recognition tools and manual analysis is used to determine trends in the

data and draw conclusions.

- Examples of popular processing and analysis tools are

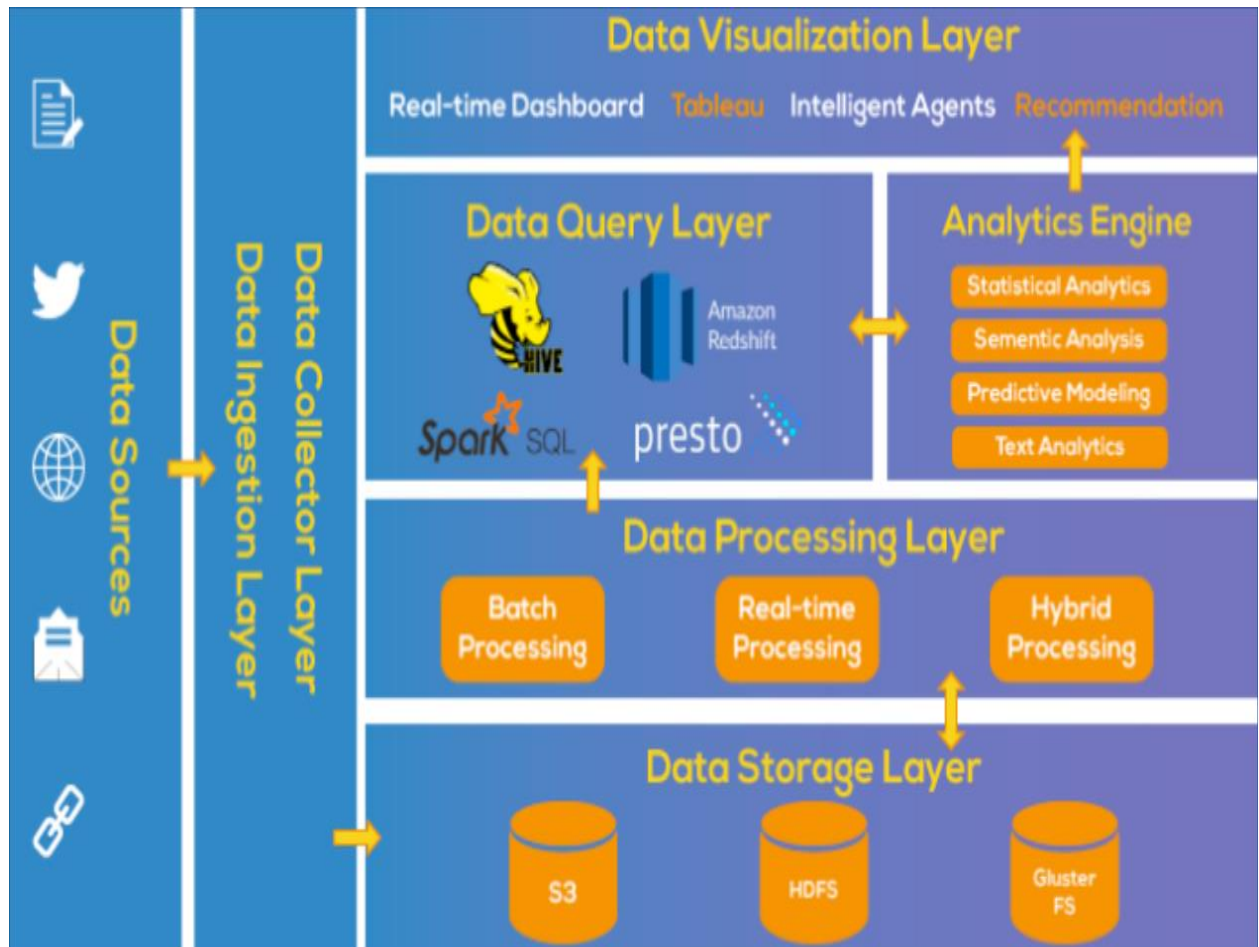


L5 is the Data Consumption or Data Output layer.

This is how the insights from the analysis is passed on to the end users who can take action to

benefit from them.

This output can take the form of reports, charts, figures and key recommendations.



Managing Data for Analysis

- Data managing means enabling, controlling, protecting, delivering and enhancing the value

of data and information asset.

Data management functions include:

- Data assets creation, maintenance and protection.
- Data governance, which includes establishing the processes for ensuring the availability, usability, integrity, security and high-quality of data.
- Data architecture creation, modeling and analysis.

- Database maintenance, administration and management system.

For example, RDBMS, NoSQL.

- Managing data security, data access control, deletion, privacy and security.
- Managing the data quality.
- Data collection using the ETL(Extract, Transform, Load) process.
- Managing documents, records and contents.
- Creation of reference and master data and data control and supervision.
- Data and application integration.
- Integrated data management, enterprise-ready data creation, fast access and analysis, automation and simplification of operations on the data.
- Data warehouse management.
- Maintenance of business intelligence.
- Data mining and analytics algorithms.

DATA SOURCES

- There are two types of big data sources
 - Internal
 - External
- Data is internal if a company generates, owns and controls it.
- Corporate ERP modules, Internal documents, website logs are some examples of internal data.
- External data is public data or the data generated outside the company. The company neither owns nor controls it.
- Surveys, questionnaires, research and customer feedback are some examples of external data.
- Data sources can be structured, semi-structured, multi-structured or unstructured.

Structured Data Sources

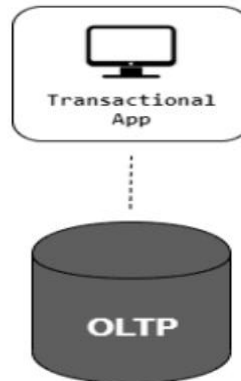
- Structured data source for ingestion, storage and processing can be a file or a database.
- The data source may be on the same computer running a program or a networked computer.

- Examples of structured data sources are SQL Server, MySQL, Microsoft Access database,

Oracle DBMS, IBM DB2 etc.

- In addition to databases, other sources of structured data include,

The image shows a screenshot of a financial projection spreadsheet. The title is 'Financial Projections'. It displays a table with columns for months (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec) and rows for various financial metrics. The metrics include Sales, Production Costs, Total Costs, and Total Profit. The data is organized into sections for Sales, Production, and Total Costs, with sub-sections for each. The spreadsheet is titled 'Year 1 Financial Projections'.



The image shows a screenshot of a 'Room Reservation' form. The form has a progress bar at the top with three steps, the first of which is completed. The 'Contact details' section includes fields for 'NAME' (First: Richard, Last: Johnson), 'PHONE' (98654237892), and 'EMAIL' (richardjohnson@zyker.com). The 'CHECK-IN DATE' is 07-Jan-2020 01:51 PM and the 'CHECK-OUT DATE' is 11-Jan-2020 4:51 PM. There is a 'Next' button at the bottom right.



```
214.1.211.251 - - [15/Apr/2011:09:40:17 -0700] "GET /global.asa HTTP/1.0" 404 315 "-"
214.1.211.251 - - [15/Apr/2011:09:40:17 -0700] "GET /-root HTTP/1.0" 404 310 "-"
214.1.211.251 - - [15/Apr/2011:09:40:18 -0700] "GET /-apache HTTP/1.0" 404 312 "-"
219.167.17.173 - - [17/Apr/2011:17:55:40 -0700] "POST /sony/mmr HTTP/1.1" 200 130 "-"
218.41.54.67 - - [17/Apr/2011:18:20:18 -0700] "POST /sony/mmr HTTP/1.1" 200 130 "-"
10.132.93.114 - - [18/Apr/2011:11:05:39 -0700] "POST /sony/mmr HTTP/1.1" 200 61 "-"
10.132.93.114 - - [18/Apr/2011:11:07:07 -0700] "POST /sony/mmr HTTP/1.1" 200 61 "-"
10.132.93.114 - - [18/Apr/2011:11:13:52 -0700] "POST /sony/mmr HTTP/1.1" 200 61 "-"
218.41.54.67 - - [20/Apr/2011:17:42:37 -0700] "POST /sony/mmr HTTP/1.1" 200 100 "-"
60.34.131.229 - - [20/Apr/2011:18:22:32 -0700] "POST /sony/mmr HTTP/1.1" 200 100 "-"
202.213.251.245 - - [21/Apr/2011:21:16:45 -0700] "POST /sony/mmr HTTP/1.1" 200 100 "-"
202.213.251.245 - - [21/Apr/2011:21:24:43 -0700] "POST /sony/mmr HTTP/1.1" 200 100 "-"
178.202.110.92 - - [22/Apr/2011:18:59:05 -0700] "GET / HTTP/1.1" 200 315 "-"
178.202.110.92 - - [22/Apr/2011:18:59:05 -0700] "GET /favicon.ico HTTP/1.1" 404 333 "-"
178.202.110.92 - - [22/Apr/2011:18:59:05 -0700] "GET /access-navigator-media HTTP/1.1" 20
178.202.110.92 - - [22/Apr/2011:19:05:00 -0700] "GET /admin/cdr/counter.txt HTTP/1.1" 404
178.202.110.92 - - [22/Apr/2011:19:05:41 -0700] "GET /help/readme.nsf?OpenAbout HTTP/1.1
178.202.110.92 - - [22/Apr/2011:19:05:54 -0700] "GET /catinfoA HTTP/1.1" 404 329 "-"
178.202.110.92 - - [22/Apr/2011:19:06:08 -0700] "GET /errors-navigator-media HTTP/1.1" 20
178.202.110.92 - - [22/Apr/2011:19:27:04 -0700] "GET / HTTP/1.1" 200 315 "-"
```

Naming a structured data source is also very important. The name needs to be meaningful.

- A data source name implies a defined name, which a process uses to identify the source.
- For example, a source that holds data about student grades could be named as StudentName_Grades.

- Data Dictionary is another way by which data can be easily accessed and managed.
- A data dictionary is a centralized repository of metadata

The dictionary consists of a set of master lookup tables and resides at a central location.

- The central location enables easier access as well as administration of changes in sources.

Field Name	Data type	Field Size	Data Format	Description	Example
StudentID	Text	10	NLLNNLLNNN	Unique ID of a student	1BO10IS001
First Name	Text	20		First name of Student	Anand
Last Name	Text	20		Last name of Student	Ravindran
DOB	Date	10	NN/NN/NNNN	Date of birth of student	06/07/1994
Address	Text	100		Address of student	#32, 2 nd cross, Kattegenahalli, Yelahanka, Bangalore
PostCode	Number	6	NNNNNN	Postal code of student	560063
Mobile	Number	10	NNNNNNNNNN	Mobile number of student	9632587412

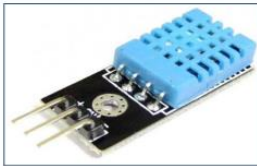
Unstructured Data Sources

- Unstructured data is the data which does not conform to a data model and has no easily identifiable structure.
- Unstructured data is not organized in a pre-defined manner or does not have a pre-defined data model.
- Some characteristics of unstructured data are:
 - Data can not be stored in the form of rows and columns as in Databases.
 - Data does not follows any rules.
 - Data lacks any particular format or sequence.
 - Due to lack of identifiable structure, it can not used by computer programs easily.
- Sources of Unstructured Data include
 - Text files
 - Web pages
 - Images (JPEG, GIF, PNG, etc.)
 - Videos
 - Word documents and PowerPoint presentations

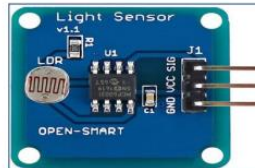
Data Sources - Sensors, Signals and GPS

- Sensors are electronic devices that sense the physical environment.
- Sensors are devices which are used for measuring temperature, pressure, humidity, light intensity, acceleration, locations, object(s) proximity etc.

- Sensors play an active role in the automotive industry.
- RFIDs and their sensors play an active role in RFID based supply chain management and tracking parcels, goods and delivery.
- Sensors embedded in processors, which include machine-learning instructions and wireless communication capabilities sources in IoT applications.



11/4/2021



Padmavathi H G – 18CS72



84

Data quality

- Data quality is the measure of how well suited a data set is to serve its specific purpose.
- Data quality is said to be high if it enables all the required operations, analysis, decisions,

planning and knowledge discovery correctly.

- Data quality is determined by 5 important characteristics:

- Relevancy
- Recency
- Range
- Robustness
- Reliability

Data Integrity

- Data integrity refers to the maintenance of consistency and accuracy in data over its usable

life.

- Software which stores, processes or retrieves the data, should maintain the integrity of data.

- Data should be incorruptible.
- For example, the grades of students should remain unaffected upon processing.
- To summarize, data integrity is the overall accuracy, completeness and consistency of data.

Data Noise, Outliers, Missing and Duplicate Values

Noise

- Noise in data refers to data giving additional meaningless information besides true (actual/required) information.
- Noise refers to difference in the value measured from true value due to additional influences.
- Noise is random in character, which means frequency with which it occurs is variable over time.
- Result of data analysis is adversely affected due to noisy data.

Outliers

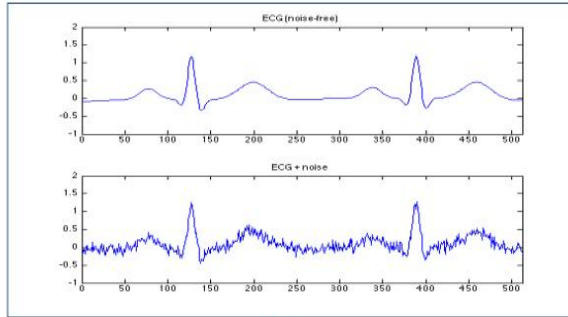
- An Outlier refers to data, which appears to not belong to the dataset.
- It is an observation that lies an abnormal distance from other values in a random sample from a population.
- Outliers are extremely high or extremely low values in a data set that can throw off your stats.
- Actual outliers need to be removed from the dataset, else the result will be affected by a small or large amount.

Missing Values

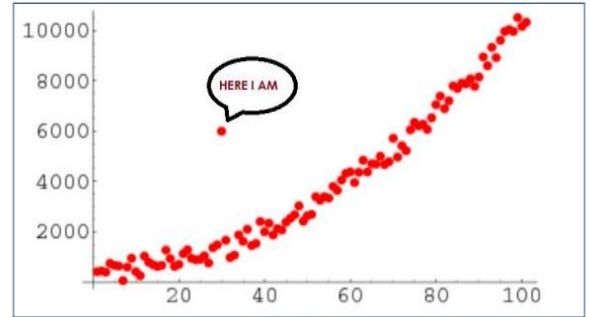
- Missing value implies data not appearing in the data set.
- Missing data is a problem because it adds ambiguity to the analysis.

Duplicate Values

- Duplicate value implies the same data appearing two or more times in a dataset.
- Presence of duplicate values or records will not result in accurate analysis.



Noise



Outliers

ID	Color	Weight	Broken	Class
1	Black	80	Yes	1
2	Yellow	100	No	2
3	Yellow	120	Yes	2
4	Blue	90	No	2
5	Blue	85	No	2
6	?	60	No	1
7	Yellow	100	?	2
8	?	40	?	1

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10248	90	5	1996-07-04	3
10249	81	6	1996-07-05	1
10250	34	4	1996-07-08	2
10251	84	3	1996-07-08	1
10251	84	3	1996-07-08	1
10252	76	4	1996-07-09	2

Data Pre-processing

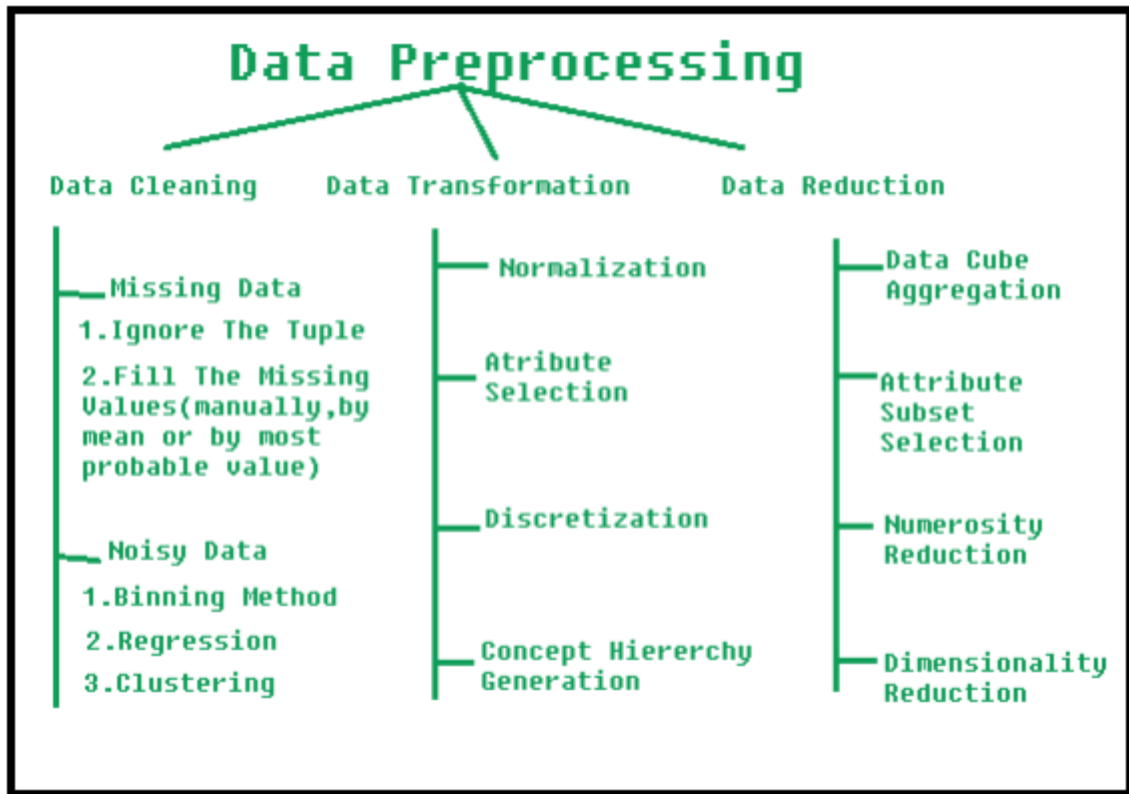
Data pre-processing is an important step at the ingestion layer.

Data when being exported to a cloud service or data store needs pre• processing.

Data preprocessing is a technique which is used to transform the raw data in a useful and efficient

Pre-processing needs are:

- (i) Dropping out of range, inconsistent and outlier values.
- (ii) Filtering unreliable, irrelevant and redundant information
- (iii) Data cleaning, editing, reduction and/ or wrangling
- (iv) Data validation, transformation or transcoding
- (v) ETL processing.



Data Cleaning

Data cleaning refers to the process of removing or correcting incomplete, incorrect, inaccurate or irrelevant parts of the data after detecting them.

For example,

correcting the grade outliers or mistakenly entered values means cleaning and correcting the

data.

Data Cleaning

Data Cleaning Tools

Data cleaning is done before mining of data.

Incomplete or irrelevant data may result into misleading decisions.

Data can generate in a system in many formats when it is obtained from the web.

Data cleaning tools help in refining and structuring data into usable data. Examples of such

tools are OpenRefine and DataCleaner.

Data Enrichment

Techopedia definition is as follows: "Data enrichment refers to operations or processes which refine, enhance or improve the raw data."

Data Editing

Data editing refers to the process of reviewing and adjusting the acquired datasets.

The editing controls the data quality.

Editing methods are

- (i) interactive,
- (ii) selective,
- (iii) automatic,
- (iv) aggregating and
- (v) distribution.

Data Reduction

Data reduction enables the transformation of acquired information into an ordered, correct and simplified form.

The reductions enable ingestion of meaningful data in the datasets.

The basic concept is the reduction of multitudinous amount of data, and use of the meaningful parts.

The reduction uses editing, scaling, coding, sorting, collating, smoothening, interpolating and preparing tabular summaries.

.

Data Wrangling

Data wrangling refers to the process of transforming and mapping the data.

Results from analytics are then appropriate and valuable.

For example, mapping enables data into another format, which makes it valuable for analytics and data visualizations.

Data Format used during Pre-Processing

Examples of formats for data transfer from

- (a) data storage,
- (b) analytics application,
- (c) service or

(d) cloud

can be:

(i) Comma-separated values CSV

(ii) Java Script Object Notation (JSON) as batches of object arrays or resource arrays

(iii) Tag Length Value (TLV)

(iv) Key-value pairs

(v) Hash-key-value pairs

CSV Format

An example is a table or Microsoft Excel file which needs conversion to CSV format.

A student_record.xlsx converts to student_record.csv file.

Comma-separated values (CSV) file refers to a plain text file which stores the table data of

numbers and text.

When processing for data visualization of Excel format file, the data conversion will be done

from csv to xlsx format.

CSV Format

Each CSV file line is a data record.

Each record consists of one or more fields, separated from each other by commas.

RFC 4180 standard specifies the various specifications.

A CSV file may also use space, tab or delimiter tab-separated formats for the values in the

fields.

Data Format Conversions

Transferring the data may need pre-processing for data-format conversions.

Data sources store need portability and usability.

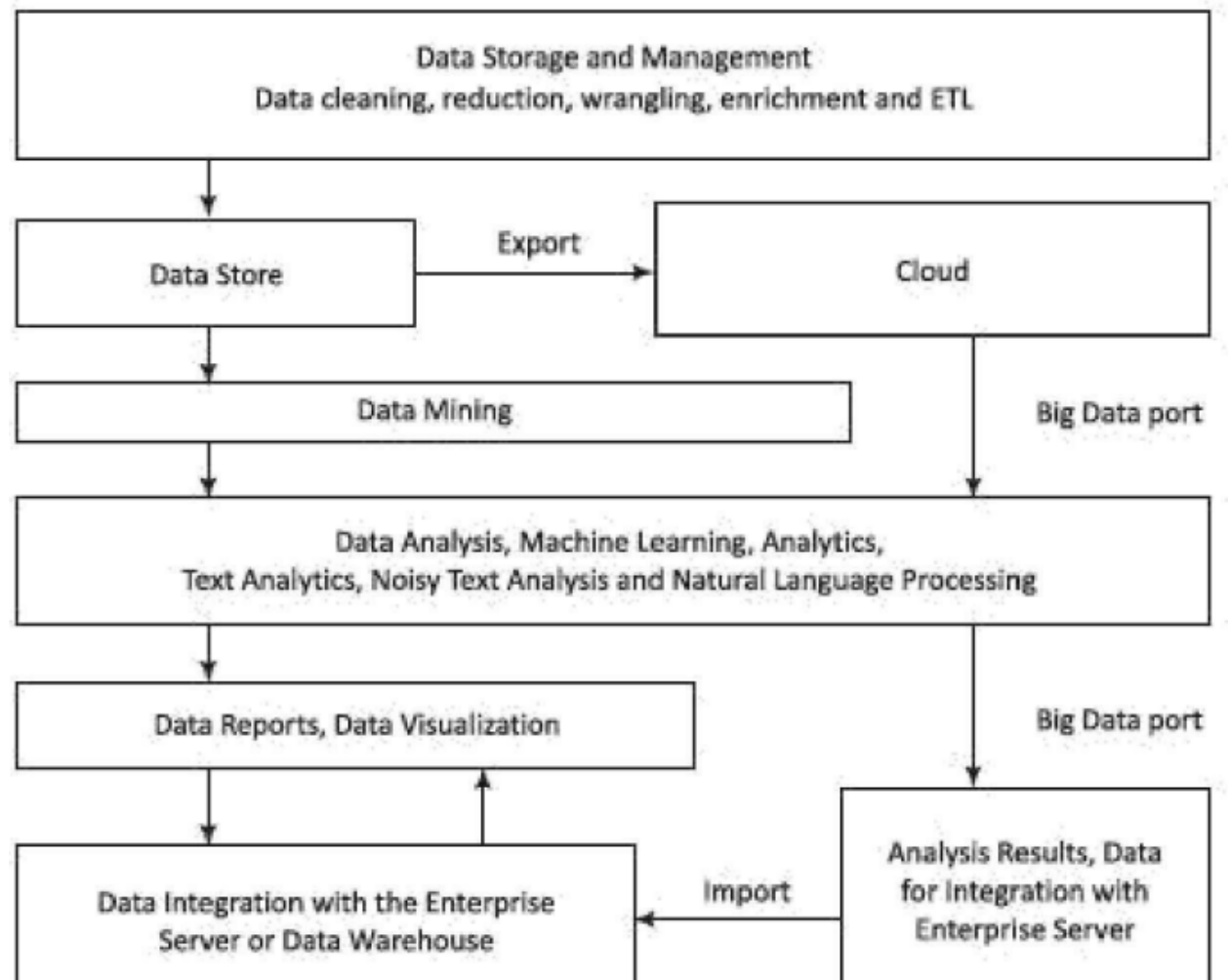
A number of different applications, services and tools need a specific format of data only.

Pre-processing before their usages or storage on cloud services is a must

Data Store Export to Cloud

Figure 1.3 shows resulting data pre-processing, data mining, analysis, visualization and data store. The data exports to cloud services.

The results integrate at the enterprise server or data warehouse.



Cloud Services

Cloud offers various services.

These services can be accessed through a cloud client (client application), such as a web browser, SQL or other client.

Figure 1.4 shows data-store export from machines, files, computers, web servers and web services.

The data exports to clouds, such as IBM, Microsoft, Oracle, Amazon, Rackspace, TCS, Tata Communications or Hadoop cloud services.

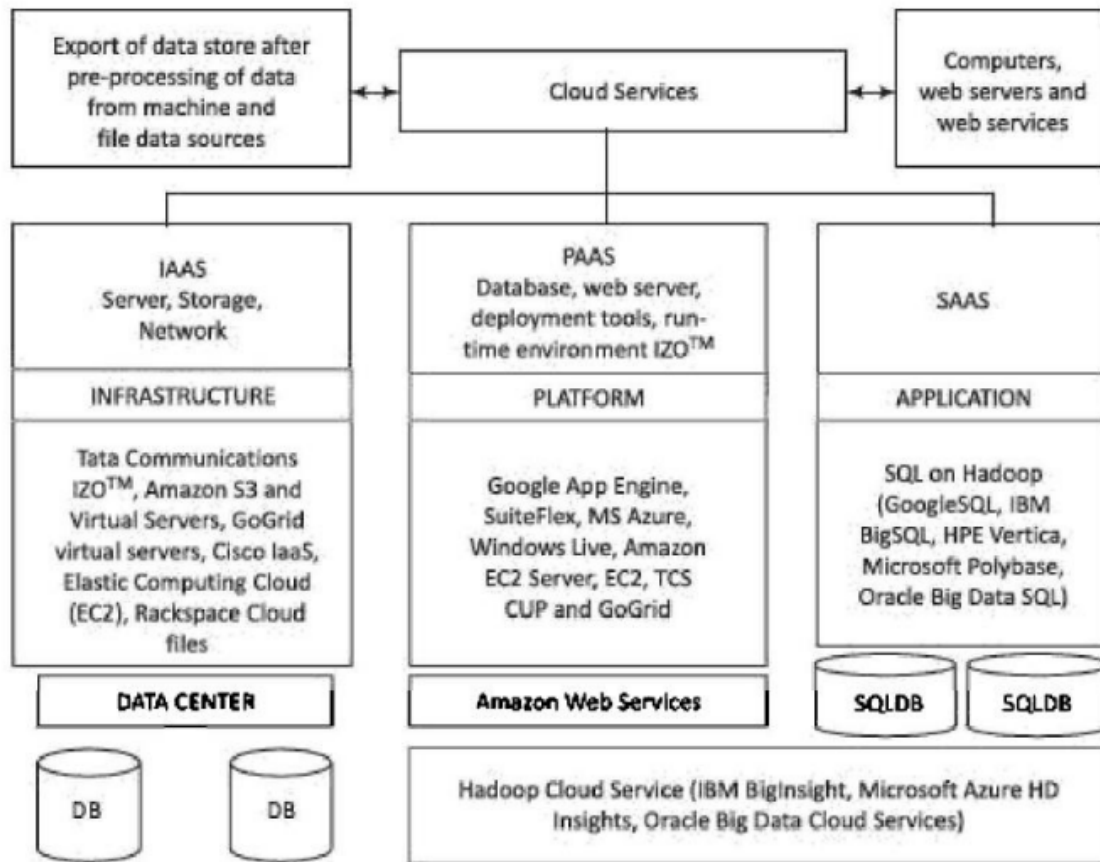


figure 1.4 Data store export from machines, files, computers, web servers and web services

Export of Data to AWS and Rackspace Clouds

The following example explains the export processes to Amazon and Rackspace clouds.

- How do the rows in MySQL database table export to Amazon AWS?
- How do the rows in MySQL database table export to Rackspace?

Data Store Export to Cloud

Export of Data to AWS and Rackspace Clouds

SOLUTION

- Following are the steps to export to an EC2 instance:
 - A process pre-processes the data-rows of table in MySQL database and creates a CSV file.
 - EC2 instance provides AWS data pipeline.

(iii) The CSV file exports to Amazon S3 using pipeline. The CSV file then copies into an S3

bucket.

(iv) AWS notification service (SNS) sends notification on completion.

(b) Following are the steps to export to Rackspace9:

(i) One or more databases create a database instance.

An instance name has maximum 255 characters.

The process of creation can be configured to create an instance now or later.

Each database can have a number of users.

(ii) Default port number for binding of MySQL is port 3306.

(ii) Command

```
mysqldump - u root - p database_name > database_name.sql
```

exports to Rackspace cloud

(iii) When a database is at a remote host then a command

```
mysqldump- h host_name - u user_name -p database_name > database_name.sql
```

Google cloud platform provides a cloud service called BigQuery.

Figure 1.5 shows BigQuery cloud service at Google cloud platform.

After pre-processing the data exports from a table or partition schema, JSON, CSV or AVRO files from data sources.

Data Store first pre-processes from machine and file data sources.

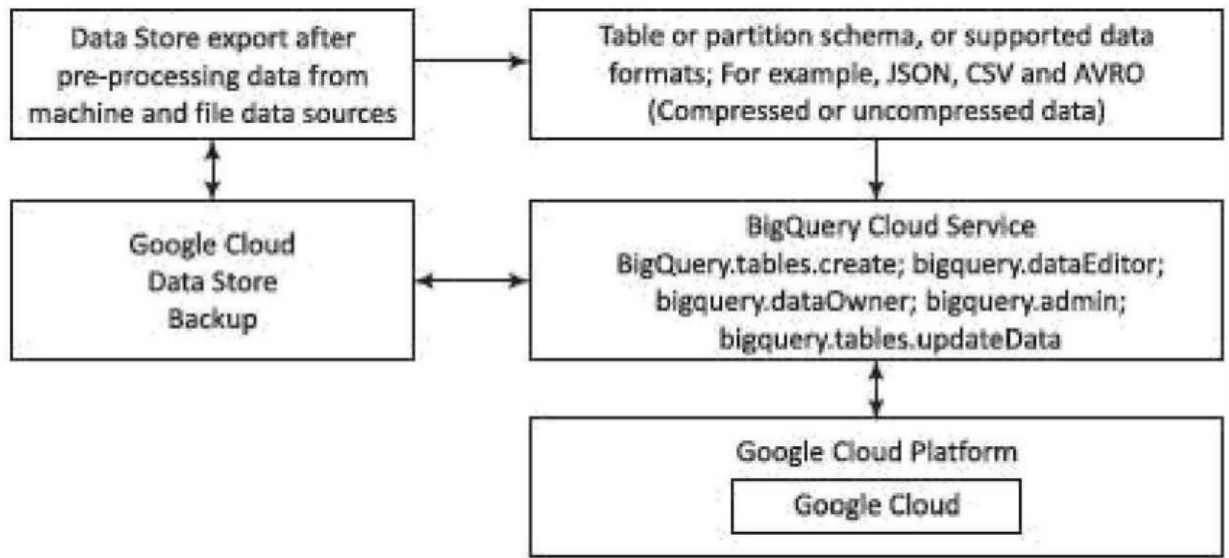
Pre-processing transforms the data in table or partition schema or supported data formats.

For example, JSON, CSV and AVRO.

Google cloud platform provides a cloud service called Big Query.

Figure 1.5 shows Big Query cloud service at Google cloud platform.

Figure 1.5 BigQuery cloud service at Googlecloud platform



Data Storage and Management: Traditional Systems

Data Store with Structured or Semi-Structured Data

Traditional systems use structured or semi-structured data.

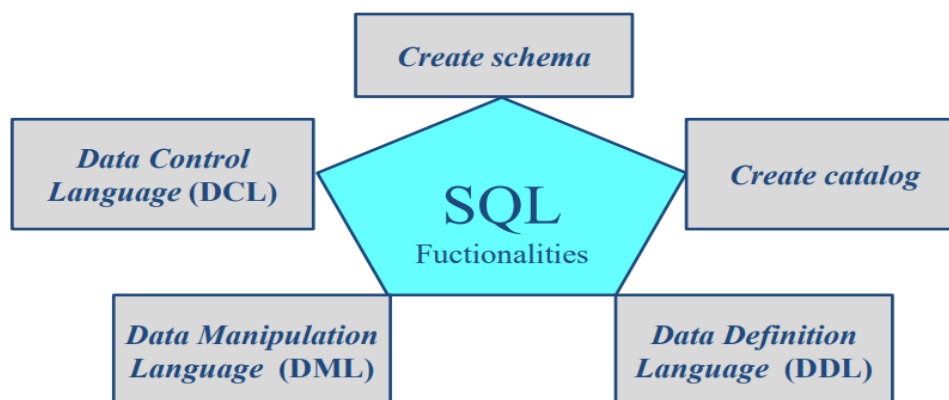
SQL An RDBMS uses SQL (Structured Query Language).

SQL is a language for viewing or changing (update, insert or append or delete) databases.

It is a language for data access control, schema creation and

data

modifications.



Data Storage and Management: Traditional Systems

Data Store with Structured or Semi-Structured Data SQL

Functionalities of SQL

1. Create schema, which is a structure which contains description of objects (base tables, views, constraints) created by a user.

The user can describe the data and define the data in the database.

2. Create catalog, which consists of a set of schemas which describe the database.

3. Data Definition Language (DDL) for the commands which depicts a database, that include creating, altering and dropping of tables and establishing the constraints.

A user can create and drop databases and tables, establish foreign keys, create view, stored procedure, functions in the database.

Data Store with Structured or Semi-Structured Data

SQL

4. Data Manipulation Language (DML) for commands that maintain and query the database.

A user can manipulate (INSERT/UPDATE) and access (SELECT) the data.

5. Data Control Language (DCL) for commands that control a database, and include administering of privileges and committing.

A user can set (grant, add or revoke) permissions on tables, procedures and views.

Data Store with Structured or Semi-Structured Data :

Large Data Storage using RDBMS

RDBMS tables store data in a structured form.

The tables have rows and columns.

Data management of Data Store includes

- privacy and security,
- data integration,
- compaction and
- fusion.

The systems use machine-generated data, human-sourced data, and data from business processes (BP) and business intelligence (BI).

Data Store with Structured or Semi-Structured Data:

Large Data Storage using RDBMS

A set of keys and relational keys access the fields at tables, and retrieve data using queries (insert, modify, append, join or delete).

RDBMSs use software for data administration also.

Data Store with Structured or Semi-Structured Data:

Distributed Database Management System

A distributed DBMS (DDBMS) is a collection of logically interrelated databases at multiple system over a computer network.

Distributed Database Management System

1. A collection of logically related databases.
2. Cooperation between databases in a transparent manner. Transparent means that each user within the system may access all of the data within all of the databases as if they were a single database.
3. Should be 'location independent' which means the user is unaware of where the data is located, and it is possible to move the data from one physical location to another without affecting the user.

In-Memory Column Formats Data

The In-Memory Column Store (IM column store) stores tables and partitions in memory using a columnar format

A columnar format in-memory allows faster data retrieval when only a few columns in a table

need to be selected during query processing or aggregation.

A single memory access loads many values at the column.

Parquet, and ORC file are examples of columnar file formats.

Data Store with Structured or Semi-Structured Data:

In-Memory Column Formats Data

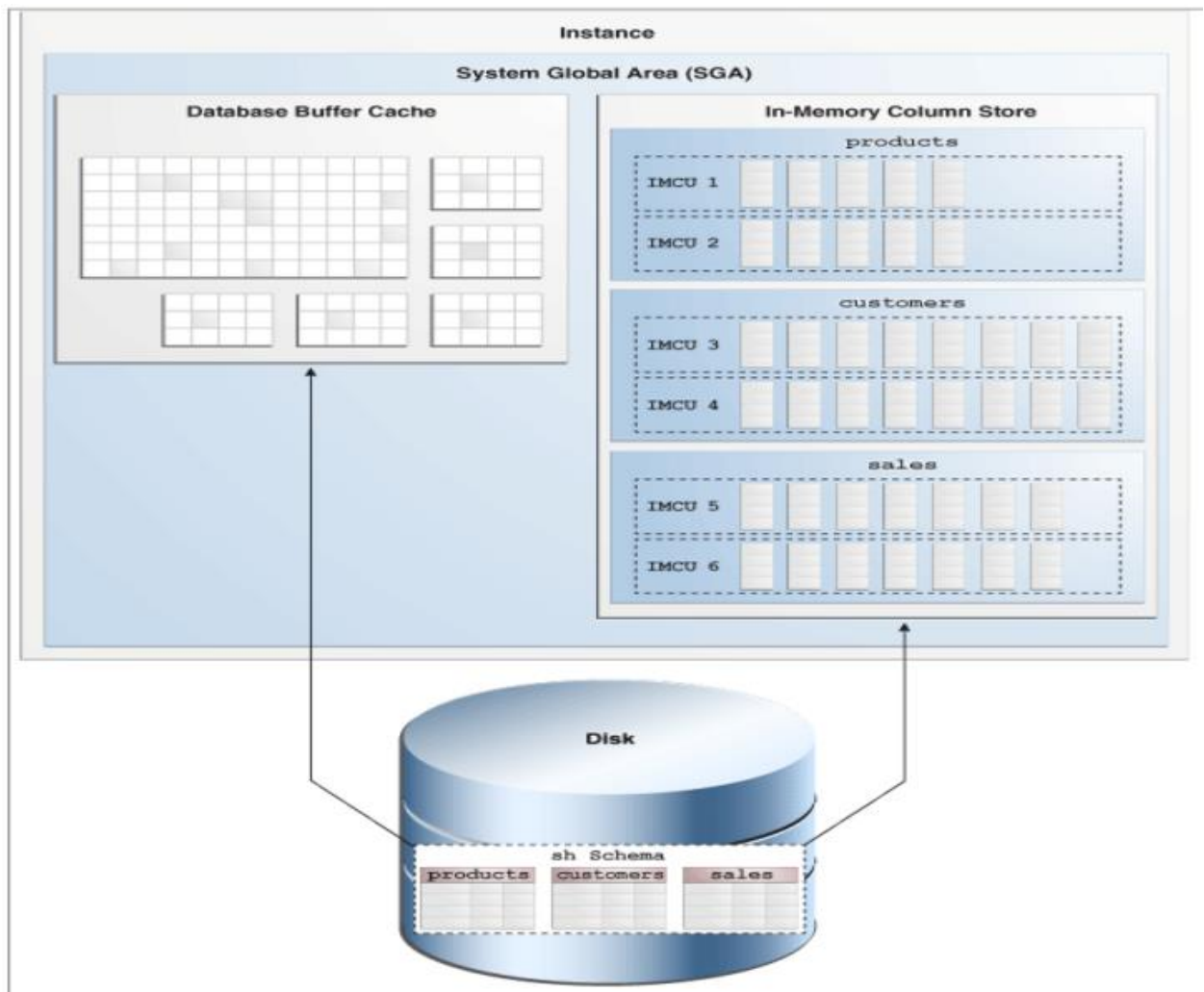
Columns Stored Contiguously



Data Store with Structured

or Semi-Structured Data :

In-Memory Column Formats Data



Data Storage and Management: Traditional Systems

Data Store with Structured or Semi-Structured Data:

In-Memory Column Formats Data

Online Analytical Processing (OLAP) in real-time transaction processing is fast when using in-memory column format tables.

OLAP enables real-time analytics.

The CPU accesses all columns in a single instance of access to the memory in columnar format in memory data-storage.

In-Memory Column Formats Data

Online Analytical Processing (OLAP) enables online viewing of analyzed data and visualization up to the desired granularity by rolling up or drilling down.

OLAP enables obtaining online summarized information and automated reports for a large

database.

Metadata describes the data.

Pre-storing of calculated values provide consistently fast response.

Result formats from the queries are based on Metadata.

In-Memory Row Format Databases

A row format in-memory allows much faster data processing during OLTP (online transaction processing).

Each row record has corresponding values in multiple columns and the on-line values store at the consecutive memory addresses in row format.

A specific day's sale of five different chocolate flavours is stored in consecutive columns c to c+5 at memory.

A single instance of memory accesses loads values of all five flavours at successive columns during online processing.

Data Storage and Management: Traditional Systems

Data Store with Structured or Semi-Structured Data :

In-Memory Row Format Databases

For example, the total number of chocolates sold computes online.

Data is in-memory row-formats in stream and event analytics.

The stream analytics method does continuous computation that happens as data is flowing through the system.

Event analytics does computation on event and use event data for tracking and reporting events.

Data Storage and Management: Traditional Systems

Data Store with Structured or Semi-Structured Data:



Data Storage and Management: Traditional Systems

Data Store with Structured or Semi-Structured Data:

Enterprise Data-Store Server and Data Warehouse

Enterprise data, after data cleaning process, integrate with the server data at warehouse.

Enterprise data server use data from several distributed sources which store data using various technologies.

All data merge using an integration tool.

Integration enables collective viewing of the datasets at the data warehouse (Figure 1.3).

DATA STORAGE AND ANALYSIS

11/4/2021 Padmavathi H G – 18CS72 130

Data Storage and Management: Traditional Systems

Data Store with Structured or Semi-Structured Data :

Enterprise Data-Store Server and Data Warehouse

Enterprise data integration may also include integration with application(s), such as analytics, visualization, reporting, business intelligence and knowledge discovery.

Heterogeneous systems execute complex integration processes when integrating at an enterprise server or data

Complex application-integration integrates heterogeneous application architectures and processes with the databases at the enterprise.

DATA STORAGE AND ANALYSIS

11/4/2021 Padmavathi H G – 18CS72 131

Data Storage and Management: Traditional Systems

Data Store with Structured or Semi-Structured Data:

Enterprise Data-Store Server and Data Warehouse

Enterprise data integration may also include integration with application(s), such as analytics, visualization, reporting, business intelligence and knowledge discovery.

Following are some standardized business processes, as defined in the Oracle application-integration architecture:

1. Integrating and enhancing the existing systems and processes
2. Business intelligence
3. Data security and integrity
4. New business services/products (Web services)
5. Collaboration/knowledge management

DATA STORAGE AND ANALYSIS

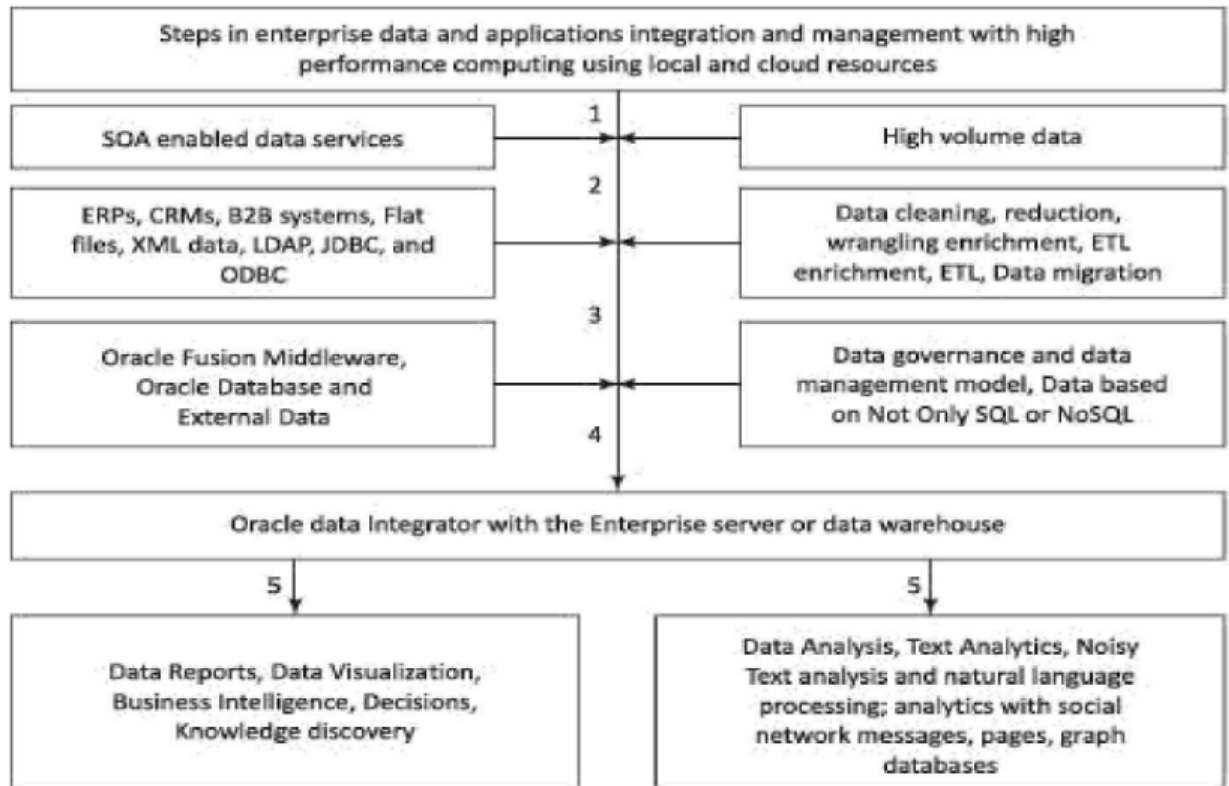
11/4/2021 Padmavathi H G – 18CS72 132

Data Storage and Management: Traditional Systems

Data Store with Structured or Semi-Structured Data:

Enterprise Data-Store Server and Data Warehouse

6. Enterprise architecture/SOA
7. e-commerce
8. External customer services
9. Supply chain automation/visualization
10. Data center optimization



Big Data Storage

Big Data NoSQL or Not Only SQL NoSQL databases are considered as semi-structured data.

Big Data Store uses NoSQL.

NOSQL stands for No SQL or Not Only SQL.

The stores do not integrate with applications using SQL.

NoSQL is also used in cloud data store.

Features of NoSQL are as follows:

1. It is a class of non-relational data storage systems, and the flexible data models and multiple schema:

- (i) Class consisting of uninterrupted key/value or big hash table
- (ii) Class consisting of unordered keys and using JSON (PNUTS)
- (iii) Class consisting of ordered keys and semi-structured data storage systems
[Big Table, Cassandra (used in Facebook/ Apache) and HBase]
- (iv) Class consisting of JSON (Mongo DB)

Features of NoSQL are as follows:

- (v) Class consisting of name/value in the text (Couch DB)
- (vi) May not use fixed table schema.
- (vii) Do not use the JOINS
- (viii) Data written at one node can replicate at multiple nodes, therefore Data storage is fault-tolerant,
- (ix) May relax the ACID rules during the Data Store transactions.
- (x) Data Store can be partitioned and follows CAP theorem (out of three properties, consistency, availability and partitions, at least two must be there during the transactions)

Consistency means all copies have the same value like in traditional DBs.

Availability means at least one copy is available in case a partition becomes inactive or fails.

Partition means parts which are active but may not cooperate as in the distributed DBs.

Coexistence of Big Data, NoSQL and Traditional Data Stores

Figure 1.7 shows coexistence of data at server,

SQL, RDBMS with NoSQL and Big Data at Hadoop, Spark, Meses, 53 or compatible Clusters.

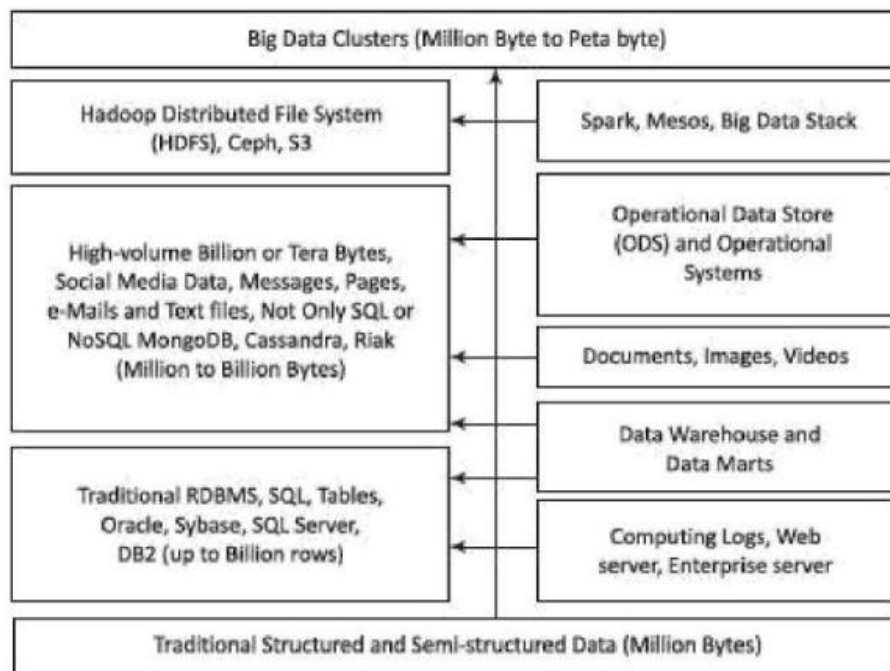


Figure 1.7 Coexistence of RDBMS for traditional server data, NoSQL and Hadoop, Spark and compatible Big Data Clusters

Big Data Storage

Coexistence of Big Data, NoSQL and Traditional Data Stores

Table 1.4 gives various data sources for Big Data along with its examples of usages and the tools used.

Table 1.4 Various data sources and examples of usages and tools

Data Source	Examples of Usages	Example of Tools
Relational databases	Managing business applications involving structured data	Microsoft Access, Oracle, IBM DB2, SQL Server, MySQL, PostgreSQL Composite, SQL on Hadoop [HPE (Hewlett Packard Enterprise) Vertica, IBM BigSQL, Microsoft Polybase, Oracle Big Data SQL]
Analysis databases (MPP, columnar, In-memory)	High performance queries and analytics	Sybase IQ, Kognitio, Terradata, Netezza, Vertica, ParAccel, ParStream, Infobright, Vectorwise,
NoSQL databases (Key-value pairs, Columnar format, documents, objects, graph)	Key-value pairs, fast read/write using collections of name-value pairs for storing any type of data; Columnar format, documents,	Key-value pair databases: Riak DS (Data Store), OrientDB, Column format databases (HBase, Cassandra), Document oriented databases: CouchDB, MongoDB; Graph databases (Neo4j, Tetan)
Hadoop clusters	Ability to process large data sets across a distributed computing environment	Cloudera, Apache HDFS
Web applications	Access to data generated from web applications	Google Analytics, Twitter
Cloud data	Elastic scalable outsourced databases, and data administration services	Amazon Web Services, Rackspace, GoogleSQL
Individual data	Individual productivity	MS Excel, CSV, TLV, JSON, MIME type
Multidimensional	Well-defined bounded exploration especially popular for financial applications	Microsoft SQL Server Analysis Services
Social media data	Text data, images, videos	Twitter, LinkedIn

Big Data Platform

A Big Data platform supports large datasets and volume of data.

The data generate at a higher velocity, in more varieties or in higher veracity.

Managing Big Data requires large resources of MPPs, cloud, parallel processing and specialized tools.

Big data platform tools and services:

1. storage, processing and analytics,
2. developing, deploying, operating and managing Big Data environment.
3. reducing the complexity of multiple data sources and integration of applications into one cohesive solution,
4. custom development, querying and integration with other systems, and
5. the traditional as well as Big Data techniques.

Data management, storage and analytics of Big data captured at the companies and services require the following:

1. New innovative non-traditional methods of storage, processing and analytics
2. Distributed Data Stores
3. Creating scalable as well as elastic virtualized platform (cloud computing)
4. Huge volume of Data Stores
5. Massive parallelism
6. High speed networks
7. High performance processing, optimization and tuning
8. Data management model based on Not Only SQL or NoSQL
9. In-memory data column-formats transactions processing or dual in-memory data columns as well as row formats for OLAP and OLTP
10. Data retrieval, mining, reporting, visualization and analytics
11. Graph databases to enable analytics with social network messages, pages and data analytics.
12. Machine learning or other approaches.
13. Big data sources: Data storages, data warehouse, Oracle Big Data, Mongo DB NoSQL, Cassandra NoSQL.
14. Data sources: Sensors, Audit trail of Financial transactions data, external data such as **Web, Social Media, weather data, health records data.**

Big Data Platform

Hadoop

Big Data platform consists of Big Data storage(s), server(s) and data management and business intelligence software.

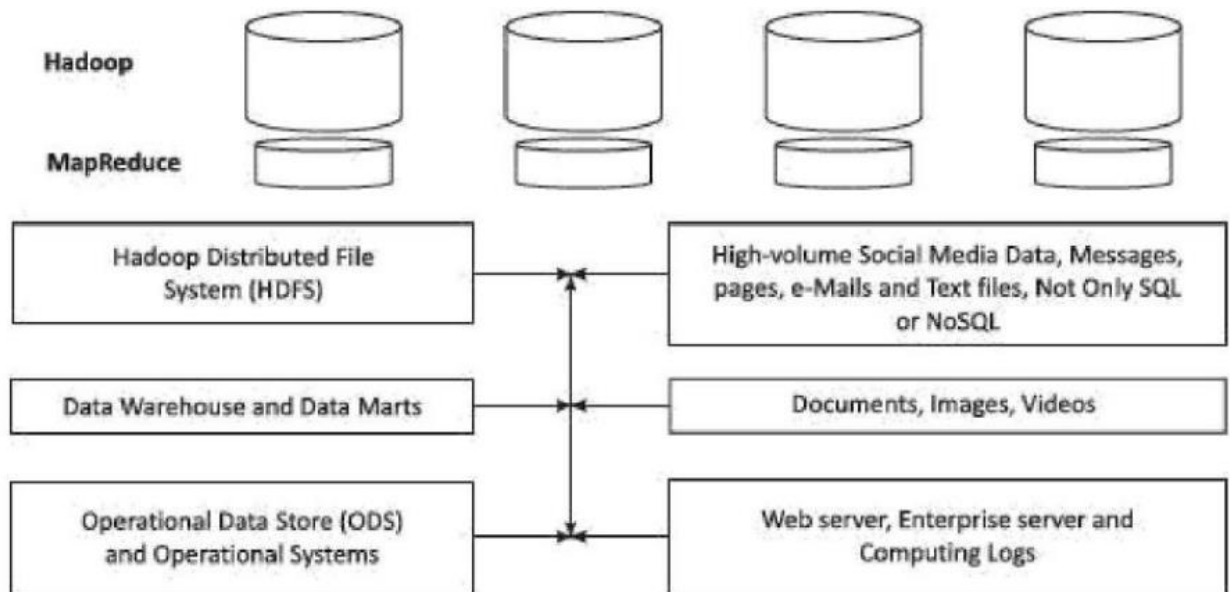
Storage can deploy Hadoop Distributed File System (HDFS), NoSQL data stores, such as HBase, MongoDB, Cassandra.

HDFS system is an open source storage system. HDFS is a scaling, self-managing and self-healing file system.

Hadoop is a scalable and reliable parallel computing platform.

Hadoop manages Big Data distributed databases.

Figure 1.8 shows Hadoop based Big Data environment. Small height cylinders represent MapReduce and big ones represent the Hadoop.



Big Data Platform Mesos

Mesos v0.9 is a resources management platform which enables sharing of

cluster of nodes by multiple frameworks and which has compatibility with an open analytics stack [data processing (Hive, Hadoop, HBase, Storm), data

management (HDFS)].

Big Data Stack

A stack consists of a set of software components and data store units.

Applications, machine-learning algorithms, analytics and visualization tools use Big Data Stack (BDS) at a cloud service, such as Amazon EC2, Azure or private cloud.

The stack uses cluster of high-performance machines.

Table 1.5 gives Big Data management, storage and processing tools.

Big Data Stack

Table 1.5 gives Big Data management, storage and processing tools.

Table 1.5 Tools for Big Data environment

Types	Examples
MapReduce	Hadoop, Apache Hive, Apache Pig, Cascading, Cascalog, mrjob (Python MapReduce library), Apache S4, MapR, Apple Acunu, Apache Flume, Apache Kafka
NoSQL Databases	MongoDB, Apache CouchDB, Apache Cassandra, Aerospike, Apache HBase, Hypertable
Processing	Spark, IBM BigSheets, PySpark, R, Yahoo! Pipes, Amazon Mechanical Turk, Datameer, Apache Solr/Lucene, ElasticSearch
Servers	Amazon EC2, S3, GoogleQuery, Google App Engine, AWS Elastic Beanstalk, Salesforce Heroku
Storage	Hadoop Distributed File System, Amazon S3, Mesos

Data analysis need pre-processing of raw data and gives information

useful for decision making. Analysis brings order, structure and meaning to the collection of data.

Data is collected and analyzed to answer questions, test the hypotheses or disprove theories.

Data Analytics can be formally defined as the statistical and mathematical data analysis that clusters, segments, ranks and predicts future possibilities.

An important feature of data analytics is its predictive, forecasting and prescriptive capability.

Analytics uses historical data and forecasts new values or results.

Analytics suggests techniques which will provide the most efficient and beneficial results for an enterprise.

Data analysis helps in finding business intelligence and helps in decision making.

Data analysis can be defined as (Wikipedia),

"Analysis of data is a process of inspecting, cleaning, transforming and modeling data with the goal of discovering useful information, suggesting conclusions and supporting decision making."

Phases in Analytics

Analytics has the following phases before deriving the new facts, providing business intelligence and generating new knowledge.

1. Descriptive analytics enables deriving the additional value from visualizations and reports.
2. Predictive analytics is advanced analytics which enables extraction of new facts and knowledge, and then predicts/forecasts.
3. Prescriptive analytics enable derivation of the additional value and undertake better decisions for new option(s) to maximize the profits.
4. Cognitive analytics enables derivation of the additional value and undertake better decisions.

Analytics integrates with the enterprise server or data warehouse.

Figure 1.9 shows an overview of a reference model for analytics architecture.

The figure also shows on the right-hand side the Big Data file systems, machine learning algorithms and query languages and usage of the Hadoop ecosystem.

Examples are: Determine root causes of defects, faults and failures in minimum time.

Deliver advertisements on mobiles or web, based on customer's location and buying habits.

Detect offender before that affects the organization or society.

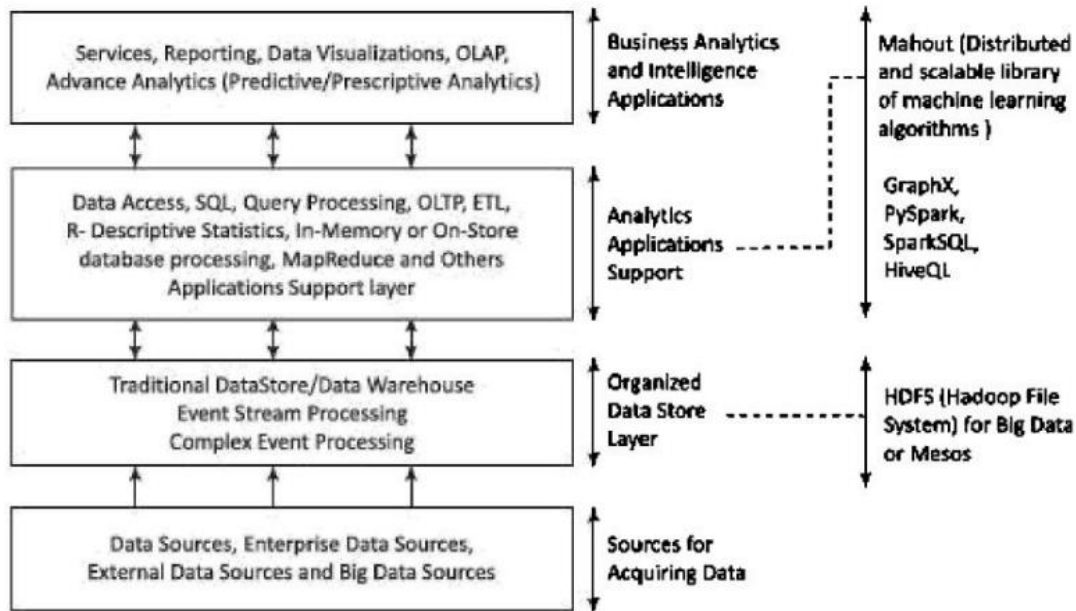


Figure 1.9 Traditional and Big Data analytics architecture reference model

Berkeley Data Analytics Stack (BDAS)

The importance of Big Data lies in the fact that what one does with it rather than how big or large it is.

Identify whether the gathered data is able to help in obtaining the following findings:

- 1) cost reduction,
- 2) time reduction,
- 3) new product planning and development,
- 4) smart decision making using predictive analytics and
- 5) knowledge discovery.

Big Data analytics need innovative as well as cost effective techniques.

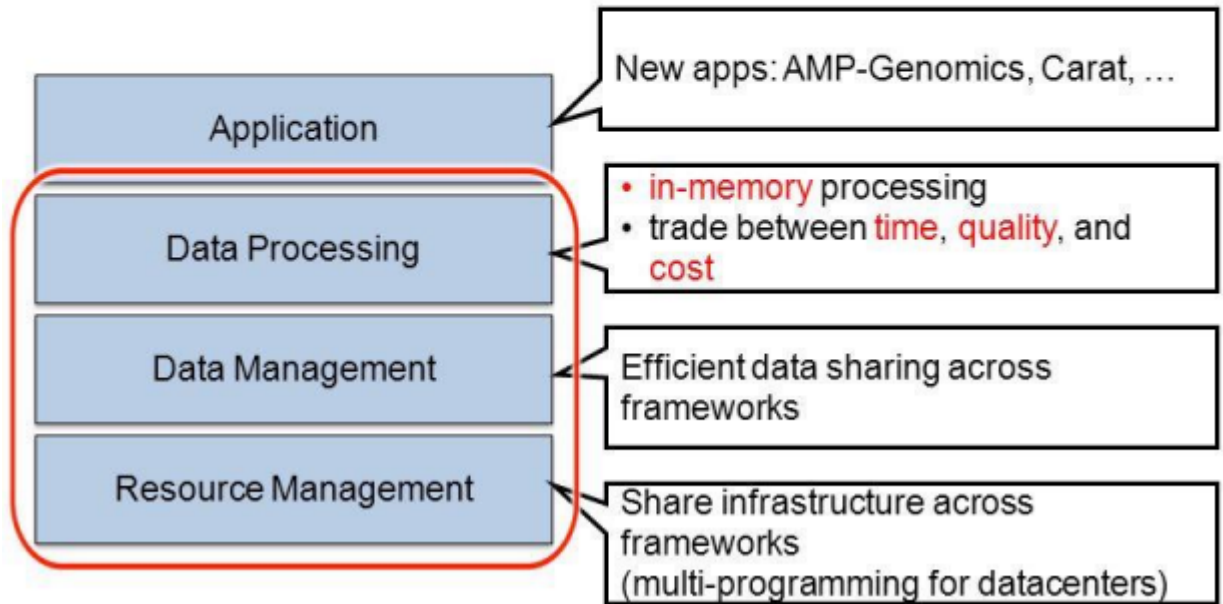
BOAS is an open-source data analytics stack for complex computations on Big Data.

It supports efficient, large-scale in-memory data processing, and thus enables user applications achieving three fundamental processing requirements; accuracy, time and cost.

Berkeley Data Analytics Stack (BDAS) consists of data processing, data

management and resource management layers.

Berkeley Data Analytics Stack (BDAS)



Big Data Analytics

Berkeley Data Analytics Stack (BDAS)

1. Applications, AMP-Genomics and Carat run at the BOAS.

Data processing software component provides in-memory processing which processes the data efficiently across the frameworks?

AMP stands for Berkeley's Algorithms, Machines and Peoples Laboratory.

2. Data processing combines batch, streaming and interactive computations.

3. Resource management software component provides for sharing the infrastructure across various frameworks.

Figure 1.10 shows a four layers architecture for Big Data Stack that consists of Hadoop, MapReduce, Spark core and parkSQL, Streaming,R, Graphx, MLib, Mahout,Arrow and Kafka.

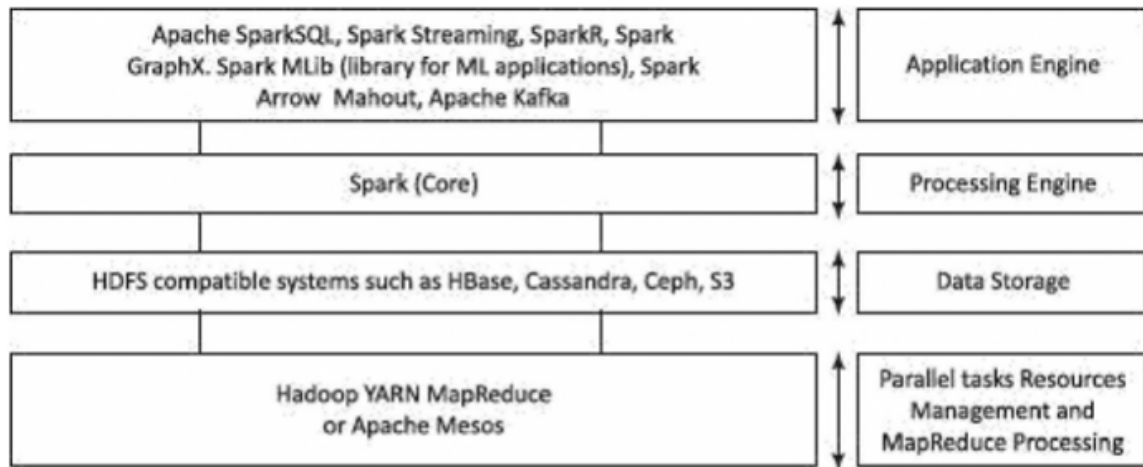


Figure 1.10 Four layers architecture for BigData Stack consisting of Hadoop, MapReduce, Spark core and SparkSQL, Streaming, R, GraphX, MLib, Mahout, Arrow and Kafka

Many applications such as social network and social media, cloud applications, public and commercial web sites, scientific experiments,

simulators and e-government services generate Big Data.

Big Data analytics find applications in many areas. Some of the popular ones are.

marketing, sales, health care, medicines, advertising etc.

Data are important for most aspect of marketing, sales and advertising.

Customer Value (CV) depends on three factors –

- **quality,**
- **service and**
- **price.**

The facts enable marketing companies to decide what products to sell.

A definition of marketing is the

"Creation, communication and delivery of value to customers".

Customer (desired) value means what a customer desires from a product.

Customer (perceived) value means what the customer believes to have received from a product after purchase of the product.

Customer value analytics (CVA) means analyzing what a customer really needs.

CVA makes it possible for leading marketers, such as Amazon to deliver the

consistent customer experiences.

Following are the five application areas in order of the popularity of Big Data use cases:

1. CVA using the inputs of evaluated purchase patterns, preferences, quality, price and post sales servicing requirements.
2. Operational analytics for optimizing company operations.
3. Detection of frauds and compliances.
4. New products and innovations in service.
5. Enterprise data warehouse optimization.

Big data is providing marketing insights into

- (i) most effective content at each stage of a sales cycle,
- (ii) investment in improving the customer relationship management (CRM),
- (iii) addition to strategies for increasing customer lifetime value (CLTV),
- (iv) lowering of customer acquisition cost (CAC).

Cloud services use Big Data analytics for CAC, CLTV and other metrics, the essentials in any cloud-based business.

Contextual marketing means using an online marketing model in which a marketer sends to potential customers the targeted advertisements, which are based on the search terms during latest browsing patterns usage by customers.

For example, if a customer is searching an airline for flights on a specific date from Delhi to Bangalore, then a smart travel agency targeting that customer through advertisements will show him/her, at specific intervals, better options for another

airline or different but cheap dates for travel or options in which price reduction occurs gradually. Big data algorithms and advanced analytics techniques enable price optimization for a given product or service, and pricing decisions, especially in the commodity driven industries where products are inelastic.

Inelastic product means a situation in which the service, required quantity or supply of a product remains unaffected by the price changes.

Big Data Analytics in Detection of Marketing Frauds

Fraud detection is vital to prevent financial losses to users.

Fraud means someone deceiving deliberately.

For example, mortgaging the same assets to multiple financial institutions, Compromising customer data and transferring customer information to third party, falsifying company information to financial institutions,

- ☐ marketing product with compromising quality,
- ☐ marketing product with service level different from the promised,
- ☐ stealing intellectual property, and much more.

Big Data usages features-for enabling detection and prevention of frauds:

1. Fusing of existing data at an enterprise data warehouse with data from sources such as social media, websites, blogs, e-mails, and thus enriching existing data.
2. Using multiple sources of data and connecting with many applications.
3. Providing greater insights using querying of the multiple source data.
4. Analyzing data which enable structured reports and visualization.

Big Data usages features-for enabling detection and prevention of frauds:

5. Providing high volume data mining, new innovative applications and thus leading to new business intelligence and knowledge discovery
6. Making it less difficult and faster detection of threats, and predict likely frauds by using various data and information publicly available.

Big Data Risks

Large volume and velocity of Big Data provide greater insights but also associate risks with the data used.

Data included may be erroneous, less accurate or far from reality.

Analytics introduces new errors due to such data.

Big Data can cause potential harm to individuals.

For example, when someone puts false or distorted data about an individual in a blog, Facebook post, WhatsApp groups or tweets, the individual may suffer loss of educational opportunity, job or credit for his/her urgent needs. A company may suffer financial losses.

Five data risks, described by Bernard Marr are

- data security,
- data privacy breach,
- costs affecting profits,
- bad analytics and bad data.

Companies need to take risks of using Big Data and design appropriate risk management procedures.

They have to implement robust risk management processes and ensure reliable predictions.

Corporate, society and individuals must act with responsibility.

Big Data Credit Risk Management

Financial institutions, such as banks, extend loans to industrial and household sectors.

These institutions in many countries face credit risks, mainly risks of

(i) loan defaults,

(ii) timely return of interests and principal amount.

Financing institutions are keen to get insights into the following:

1. Identifying high credit rating business groups and individuals,
2. Identifying risk involved before lending money
3. Identifying industrial sectors with greater risks
4. Identifying types of employees (such as daily wage earners in construction sites) and businesses (such as oil exploration) with greater risks.
5. Anticipating liquidity issues (availability of money for further issue of credit and rescheduling credit installments) over the years.

The insight using Big Data decreases the default rates in returning of loan, greater accuracy in issuing credit and faster identification of the non-payment or fraud issues of the loan receiving entities.

One innovative way to manage credit risks and liquidity risks is use of available data and Big Data.

High volume of data analysis gives greater insight into the default patterns, emerging patterns and thus credit risks.

Big Data analytics monitors social media, interactions data, contact addresses, mobile numbers, website, financial status, activities or job changes to find the emerging credit risk that may affect a customer loan returning capacity.

Digital footprints across social media provide a valuable alternative data source for credit risk analysis.

The data companies assist in rating the customer in application processing and also during the period of repayment of a loan.

Friends on Facebook and their credit rating, comments and assets posted also help in determining the risks.

The data insights from the analytics lead to credit and liquidity risk management and faster reactions.

Three benefits are

- (i) minimize the non-payments and frauds,
- (ii) identifying new credit opportunities, new customers and revenue streams, thereby broadening the company high credit rating customers base and
- (iii) marketing to low-risk businesses and households.

BIG DATA ANALYTICS APPLICATIONS AND CASE STUDIES

Big Data and Algorithmic Trading Wikipedia gives a definition of algorithm trading as follows:

"Algorithmic trading is a method of executing a large order (too large to fill all at once) using automated pre-programmed trading instructions accounting for variables such as time, price and volume."

Complex mathematics computations enable algorithmic trading and business investment decisions to buy and sell.

The input data are insights gathered from the risk analysis of market data.

Big data bigger volume, velocity and variety in the trading provide an edge over other trading entities

Big Data and Healthcare

Big Data analytics in health care use the following data sources:

- (i) clinical records,
- (ii) pharmacy records,
- (iii) electronic medical records
- (iv) diagnosis logs and notes and
- (v) additional data, such as deviations from person usual activities, medical leaves from job, social interactions.

Healthcare analytics using Big Data facilitate the following:

1. Provisioning value-based and customer-centric healthcare,
2. Utilizing the 'Internet of Things' for health care.
3. Preventing fraud, waste, abuse in the healthcare industry and reduce healthcare costs.
4. Improving outcomes
5. Monitoring patients in real time.

Value-based and customer-centric healthcare means cost effective patient care by improving healthcare quality using latest knowledge, usages of electronic health and medical records and improving coordination among the healthcare providing agencies, which reduce avoidable overuse and healthcare costs.

Healthcare Internet of Things create unstructured data. The data enables the monitoring of the devices data for patient parameters, such as glucose, BP, ECGs and necessities of visiting physicians.

Prevention of fraud, waste, and abuse uses Big Data predictive analytics and help resolve excessive or duplicate claims in a systematic manner.

The analytics of patient records and billing help in detecting, anomalies such as overutilization of services in short intervals, different hospitals in different locations simultaneously, or identical prescriptions for the same patient filed from multiple locations.

Improving outcomes is possible by accurately diagnosing patient conditions, early diagnosis, predicting problems such as congestive heart failure, anticipating and avoiding complications, matching treatments with outcomes and predicting patients at risk for disease or readmission.

Patient real-time monitoring uses machine learning algorithms which process real-time events. They provide physicians the insights to help them make life-saving decisions and allow for effective interventions. The process automation sends the alerts to care providers and informs them instantly about changes in the condition of a patient.

Big Data in Medicine

Big Data analytics deploys large volume of data to identify and derive intelligence using predictive models about individuals.

Big Data driven approaches help in research in medicine which can help the patients.

Big Data offers potential to transform medicine and the healthcare.

Big Data analytics deploys large volume of data to identify and derive intelligence using predictive models about individuals.

Big Data driven approaches help in research in medicine which can help the patients.

Big Data offers potential to transform medicine and the healthcare.

Following are some findings:

building the health profiles of individual patients and predicting models for diagnosing better and offer better treatment,

1. Aggregating large volume and variety of information around from multiple sources the DNAs, proteins, and metabolites to cells, tissues, organs, organisms, and ecosystems, that can enhance the understanding of biology of diseases. Big data creates patterns and models by data mining and help in better understanding and research,

2. Deploying wearable devices data, the devices data records during active as well as inactive periods, provide better understanding of patient health, and better risk profiling the user for certain diseases,

The impact of Big Data is tremendous on the digital advertising industry.

The digital advertising industry sends advertisements using SMS, e-mails, WhatsApp, LinkedIn, Facebook, Twitter and other mediums.

Big Data technology and analytics provide insights, patterns and models, which relate the media exposure of all consumers to the purchase activity of all consumers using multiple digital channels.

Big Data help in identity management and can provide an advertising mix for building better branding exercises.

Big Data captures data of multiple sources in large volume, velocity and variety of data unstructured and enriches the structured data at the enterprise data warehouse.

Big data real time analytics provide emerging trends and patterns, and gain actionable insights for facing competitions from similar products.

The data helps digital advertisers to discover new relationships, lesser competitive regions and areas.

Success from advertisements depend on collection, analyzing and mining.

The new insights enable the personalization and targeting the online, social media and mobile for advertisements called hyper-localized advertising.

Advertising nowadays limits no longer to TV, radio and print.

Advertising on digital medium needs optimization.

Too much usage can also effect negatively.

Phone calls, SMSs, e-mail-based advertisements can be nuisance if sent without appropriate researching on the potential targets.

The usage of Big Data after appropriate filtering and elimination is crucial enabler of Big Data Analytics with appropriate data, data forms and data handling in the right manner.

Module 2

Introduction to Hadoop

OBJECTIVES

Introduction to Hadoop

- Introduction
- Hadoop and its Ecosystem
- Hadoop Distributed
- File System
- MapReduce Framework and Programming Model
- Hadoop Yarn
- Hadoop Ecosystem Tools.

Hadoop Distributed File System Basics

- HDFS Design Features
- Components
- HDFS User Commands.

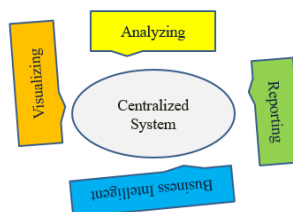
What is Hadoop?

The Technology that empowers Yahoo, Facebook, Twitter, Walmart and others



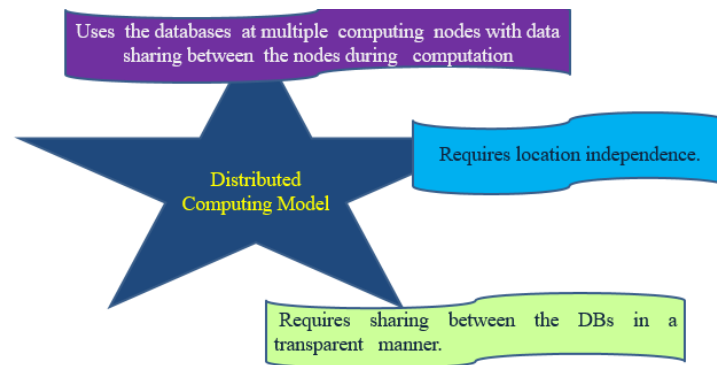
An **open source** framework that allows **distributed processing** of large data-sets across the Cluster of commodity hardware

Centralized Computing Model



All These tasks are Computed Centrally

Big Data Distributed Computing Model



Transparency between data nodes at computing nodes do not fulfill for Big Data when distributed computing takes place using data sharing between local and remote.

reasons for this are:

- ❖ Distributed data storage systems do not use the concept of joins.
- ❖ Data need to be fault-tolerant and data stores should take into account the possibilities of network failure.
- ❖ Follows CAP theorem-- out of three properties (consistency, availability and partitions), two must at least be present for applications, services and processes.

The solution is the Hadoop which provides the model for this.

Distributed computing model which requires no sharing between data nodes.

Multiple tasks of an application are also distributed, run using machines associated with multiple data nodes and execute at the same time in parallel.

Application is divided in number of tasks and sub-tasks. The sub-tasks get inputs from data nodes at the same cluster. The results of sub-tasks aggregate and communicate to the application. The aggregate results from each cluster collect using APIs at the application.

Big Data Storage Model

Data Store model of files in data nodes in racks in the clusters, Hadoop system uses the data store model.

In which storage is at clusters, racks, data nodes and data blocks.

Data blocks replicate at the DataNodes such that a failure of link leads to access of the data block from the other nodes replicated at the same or other racks

Big data programming model

Big Data programming model is that application in which application **jobs** and tasks (or sub-tasks) is **scheduled** on the same servers which store the data for processing.

Hadoop system uses the programming model, where jobs or tasks are assigned and scheduled on the same servers which hold the data

Job means running an assignment of a set of instructions for processing.

Example

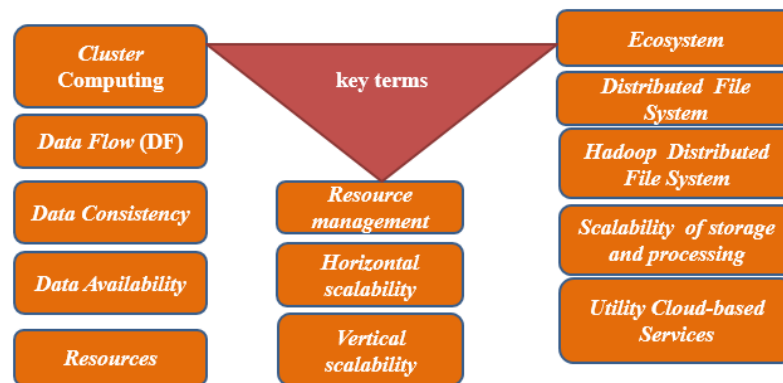
- 1) Processing the queries in an application and sending the result back to the application.
- 2) Instructions for sorting the examination performance data.

Job scheduling means assigning a job for processing following a schedule.

Example

Scheduling after a processing unit finishes the previously assigned job, scheduling as per specific sequence or after a specific period.

Big Data Storage Model



Important key terms and their meaning

Cluster Computing: refers to computing, storing and analyzing huge amounts of unstructured or structured data in a distributed computing environment.

Clusters improve

- ❖ the performance,
- ❖ provide cost-effective and
- ❖ improved node accessibility.

Data Flow (DF) refers to flow of data from one node to another. For example,

Transfer of output data after processing to input of application.

Important key terms and their meaning

Resources means computing system resources, i.e., the physical or virtual components or devices, made available for specified or scheduled periods within the system.

Examples

- ❖ Files,
- ❖ Network connections and
- ❖ Memory blocks.

Important key terms and their meaning

Resource management refers to managing resources such as their creation, deletion and controlled usages.

The manager functions includes managing the

- (i) availability for specified or scheduled periods,
- (ii) prevention of resource unavailability after a task finishes and (iii) resources allocation when multiple tasks attempt to use the same set of resources

Important key terms and their meaning

Horizontal scalability means increasing the number of systems working in coherence. Processing different datasets of a large data store *running similar application* deploys the horizontal scalability.

Vertical scalability means scaling up using the given system resources and increasing the number of tasks in the system.

Processing different datasets of a large data store *running multiple application* tasks deploys vertical scalability

Important key terms and their meaning

Vertical scalability

Example

Extending analytics processing by including the

- reporting,
- business processing (BP),
- business intelligence (BI),
- Data visualization,
- knowledge discovery and
- machine learning (ML) capabilities

All these require additional ways to solve problems.

Important key terms and their meaning

Ecosystem refers to a system made up of multiple computing components, which work together.

Distributed File System means a system of storing files

Files can be for

- The set of data records,
- Key-value pairs,
- Hash key-value pairs,
- Relational database or NoSQL database
- **Important key terms and their meaning**
- **Hadoop Distributed File System** means a system of storing files (set of data records, key-value pairs, hash key-value pairs or applications data) at distributed computing nodes according to Hadoop architecture and accessibility of data blocks after finding reference to their racks and cluster.

Important key terms and their meaning

Scalability of storage and processing means the execution using varying number of servers according to the requirements,

i.e., bigger data store on greater number of servers when required and on smaller data when smaller data used on limited number of servers.

Big Data Analytics require deploying the clusters *using the servers or cloud* for computing as per the requirements

Important key terms and their meaning

Utility Cloud-based Services mean infrastructure, software and computing platform services similar to utility services, such as electricity, gas, water etc. Infrastructure refers to units for data-store, processing and network. services at the cloud are

- IaaS,
- SaaS and
- PaaS

HADOOP AND ITS ECOSYSTEM

Hadoop has mainly two Components

- ❖ data store
- ❖ computations

Data is stored in blocks in the clusters.

Computations are done at each individual cluster in parallel with another.

Hadoop components are written in Java with part of native code in C.

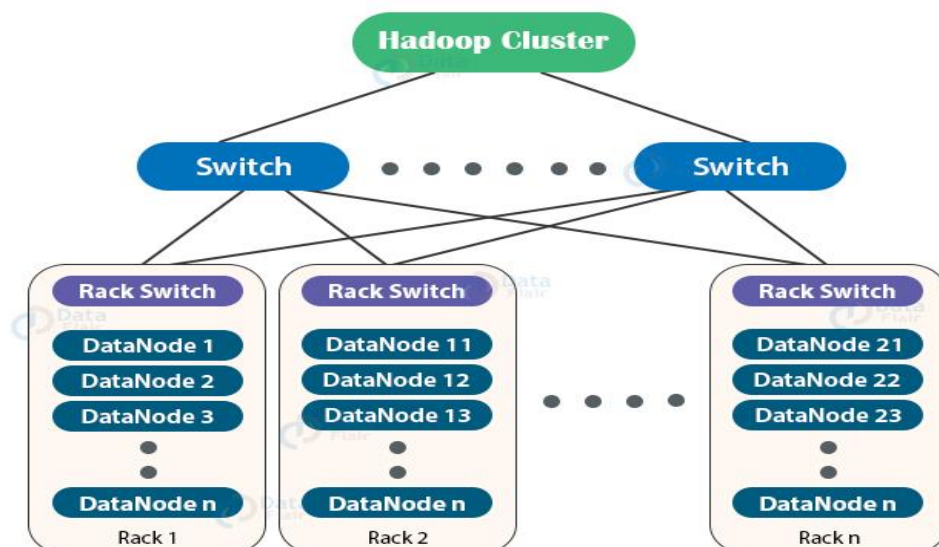
The command line utilities are written in shell scripts.

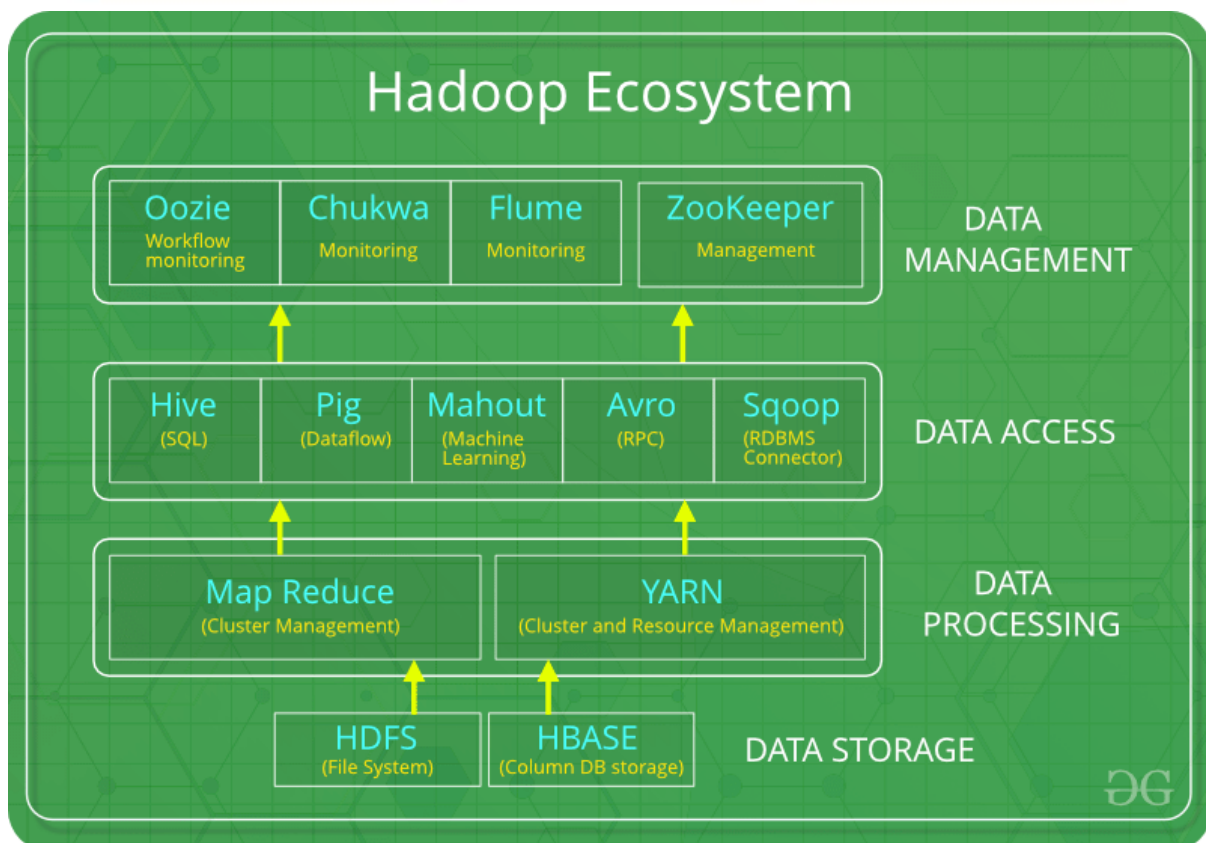
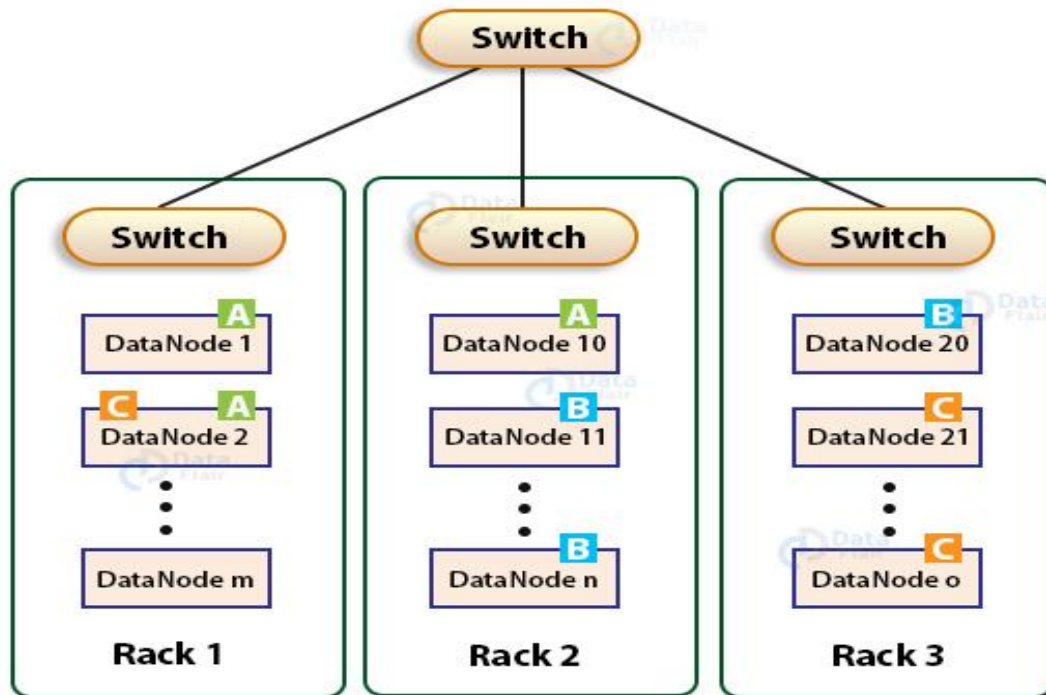
The **Rack** is the collection of around 40-50 DataNodes connected using the same network switch.

A **data node** is an appliance that can add to your event and flow processors to increase storage capacity and improve search performance.

Each data node can be connected to only one processor, but a processor can support multiple data nodes.

Block is **the physical representation of data**. It contains a minimum amount of data that can be read or written. HDFS stores each file as blocks.





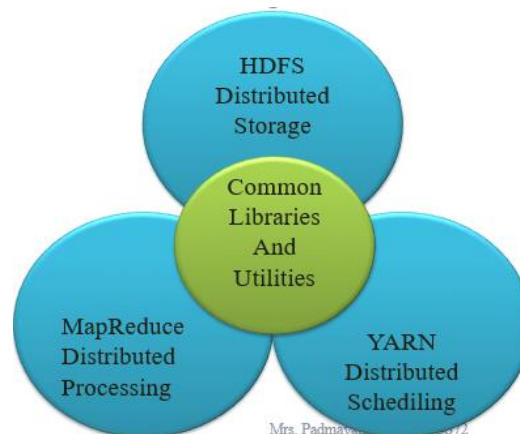
Hadoop enables Big Data storage and cluster computing.

The Hadoop system manages both, large-sized structured and unstructured data in different formats, such as XML, JSON and text with efficiency and effectiveness.

The Hadoop system performs better with clusters of many servers when the focus is on horizontal scalability.

The system provides faster results from Big Data and from unstructured data as well.

Hadoop Core Components: Figure shows the Core Components of Apache Hadoop Framework




The Hadoop core components of the framework are

Hadoop Common Module	The common module contains the libraries and utilities that are required by the other modules of Hadoop For example: various components and interfaces for distributed file system and general like Serialization, Java RPC (Remote Procedure Call) and file-based data structures.
Hadoop Distributed File System (HDFS)	A Java-based distributed file system which can store all kinds of data on the disks at the clusters.
MapReduce v1	Software programming model in Hadoop1 using Mapper and Reducer. The v1 processes large sets of data in parallel and in batches.
YARN	Software for managing resources for computing. The user application tasks or sub-tasks run in parallel at the Hadoop, uses scheduling and handles the requests for the resources in distributed running of the tasks.
MapReduce v2	Hadoop 2 YARN-based system for parallel processing of large datasets and distributed processing of the application tasks.

Spark

Spark is an open-source cluster-computing framework of Apache Software Foundation.

- Spark deploys data in-memory analytics.
- Enables OLAP and real-time processing.
- Spark does faster processing of Big Data.
- Spark has been adopted by large organizations, such as Amazon, eBay and Yahoo.
- Spark is now increasingly becoming popular.



Spark is an open-source cluster-computing framework of Apache Software Foundation.

- Spark deploys data in-memory analytics.
- Enables OLAP and real-time processing.
- Spark does faster processing of Big Data.
- Spark has been adopted by large organizations, such as Amazon, eBay and Yahoo.
- Spark is now increasingly becoming popular.

May 2022 Mrs. Padmavathi H G – 18CS72 5

Features of Hadoop: Hadoop features are

Fault-efficient scalable, flexible and modular design

Robust design of HDFS

Store and process Big Data

Distributed clusters computing model with data locality

Hardware fault-tolerant

Open-source framework

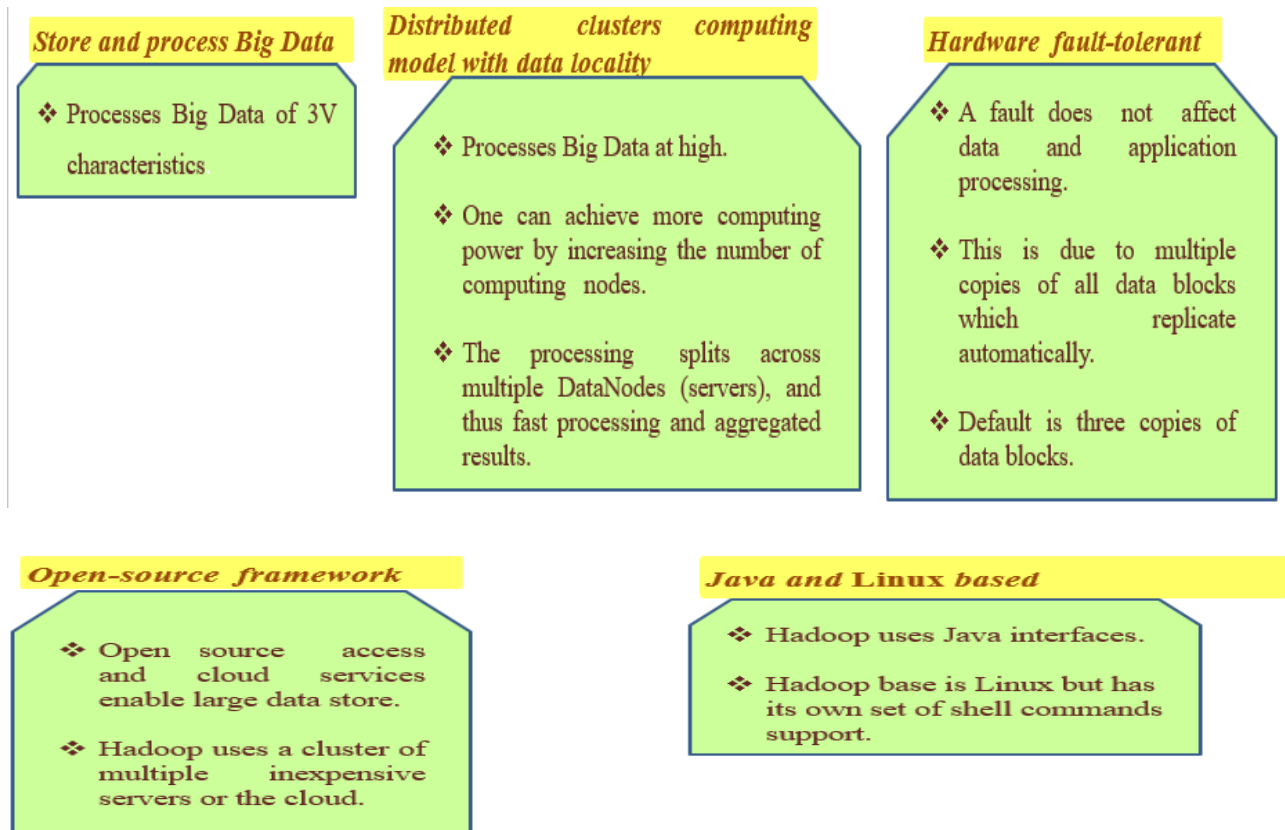
Java and Linux based

Fault-efficient scalable, flexible and modular design

- ❖ Uses simple and modular programming model.
- ❖ Provides servers at high scalability.
- ❖ The system is scalable by adding new nodes to handle larger data.
- ❖ Very helpful in storing, managing, processing and analyzing Big Data.
- ❖ Modular functions make the system flexible.
- ❖ One can add or replace components at ease. Modularity allows replacing its components for a different software tool.

Robust design of HDFS

- ❖ Execution of Big Data applications continue even when an individual server or cluster fails.
- ❖ This is because of backup and a data recovery mechanism.
- ❖ HDFS thus has high reliability.



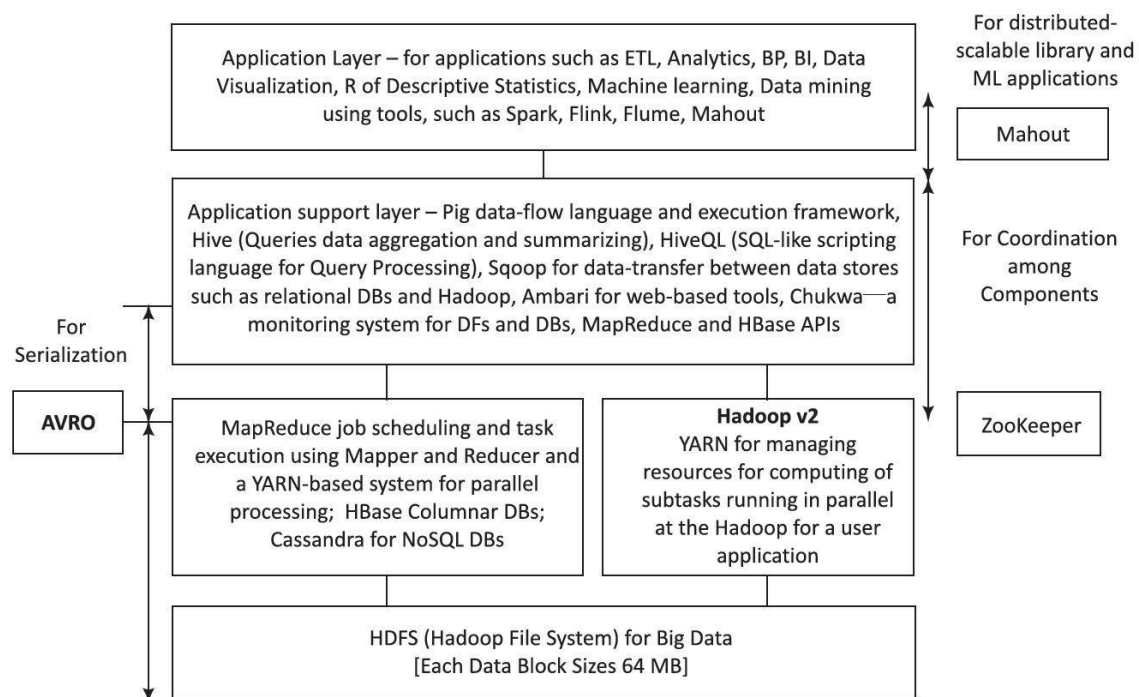
Hadoop Ecosystem Components

Hadoop ecosystem refers to a combination of technologies.

Hadoop ecosystem consists of own family of applications which support the storage, processing, access, analysis, governance, security and operations for Big Data

The system includes the application support layer and application layer components- AVRO, ZooKeeper, Pig, Hive, Sqoop, Ambari, Chukwa, Mahout, Spark, Flink and Flume.

The figure also shows the components and their usages



The four layers in Figure 2.2 are

- (i) **Distributed storage layer**
- (ii) **Resource-manager layer** for job or application sub-tasks scheduling and execution
- (iii) **Processing-framework layer**, consisting of Mapper and Reducer for the MapReduce process-flow.
- (iv) **APIs at application support layer** (applications such as Hive and Pig).

The codes communicate and run using MapReduce or YARN at processing framework layer. Reducer output communicate to Apis
AVRO enables data serialization between the layers. *Zookeeper* enables coordination among layer components

Hadoop Streaming

HDFS with MapReduce and YARN-based system enables parallel processing of large datasets. The two stream processing technologies are

- Spark
- Flink

The two lead stream processing systems and are more useful for processing a large volume of data.

Spark	Flink
<ul style="list-style-type: none">❖ Spark provides in-memory processing of data, thus improving the processing speed.❖ Spark includes security features.	<ul style="list-style-type: none">❖ Flink improves the overall performance as it provides single run-time for streaming as well as batch processing.❖ Simple and flexible architecture of Flink is suitable for streaming data.

Hadoop Pipes

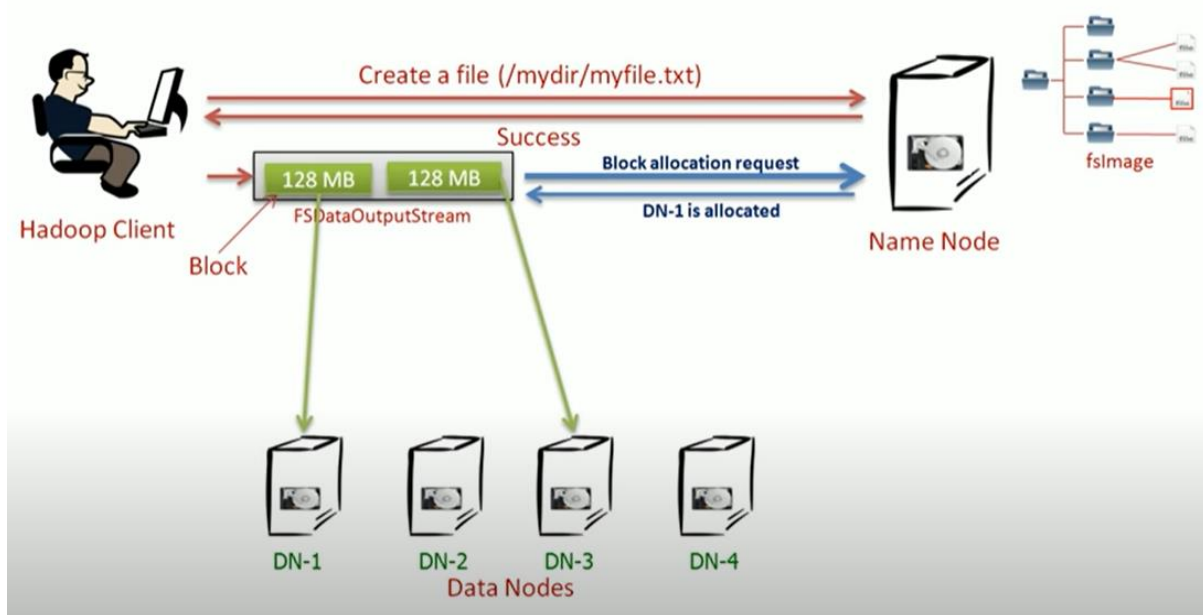
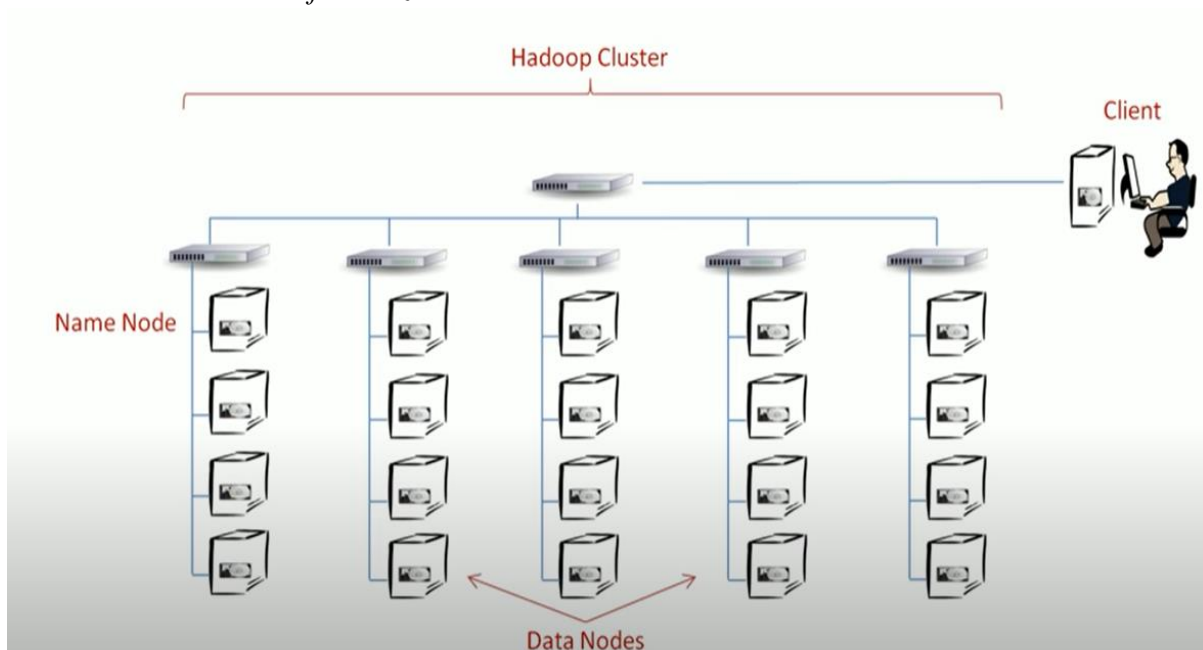
- ❖ Hadoop Pipes are the C++ Pipes which interface with MapReduce.
- ❖ A pipe means data streaming into the system at Mapper input and aggregated results flowing out at outputs.
- ❖ Apache Hadoop provides an adapter layer, which processes in pipes.
- ❖ The adapter layer enables running of application tasks in C++ coded MapReduce programs.
- ❖ Pipes do not use the standard I/O when communicating with Mapper and
- ❖ Reducer codes. Cloudera distribution including Hadoop (CDH) version CDH 5.0.2 runs the pipes.
- ❖ Applications which require faster numerical computations can achieve higher throughput using C++ when used through the pipes

HADOOP DISTRIBUTED FILE SYSTEM (HDFS)

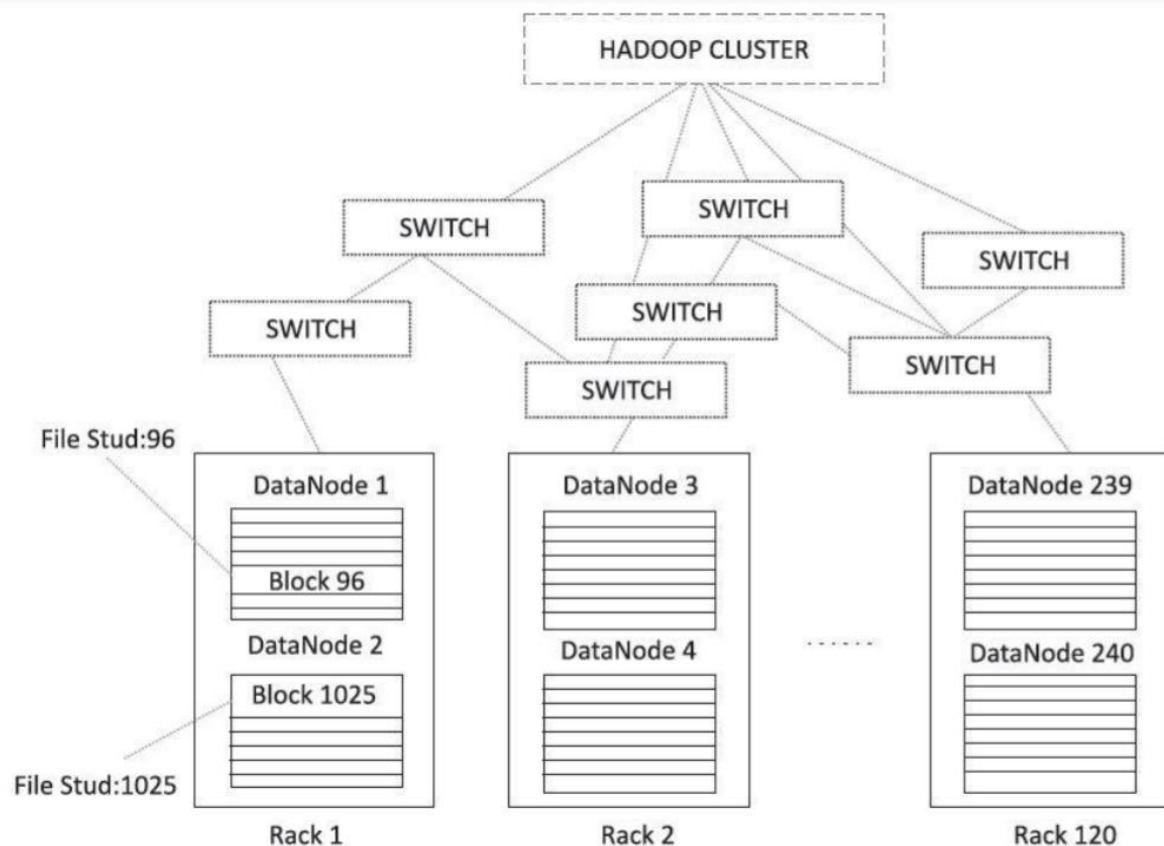
- ❖ HDFS is a core component of Hadoop.
- ❖ HDFS is designed to run on a cluster of computers and servers at cloud-based utility services.
- ❖ HDFS stores Big Data which may range from GBs to PBs
- ❖ HDFS stores the data in a distributed manner in order to compute fast.
- ❖ The distributed data store in HDFS stores data in any format regardless of schema.
- ❖ HDFS provides high throughput access to data-centric applications that require large-scale data processing workloads.

HDFS Data Storage

- ❖ Hadoop data store concept implies storing the data at a number of *clusters*.
- ❖ Each cluster has a number *racks*.
- ❖ Each rack stores a number of DataNodes.
- ❖ Each DataNode has a large number of *data blocks*.
- ❖ The racks distribute across a cluster.
- ❖ The nodes have processing and storage capabilities.
- ❖ The nodes have the data in data blocks to run the application tasks.
- ❖ The data blocks *replicate by default at least on three DataNodes* in same or remote nodes.
- ❖ Data at the stores enable running the distributed applications including analytics, data mining, OLAP using the clusters.
- ❖ A file, containing the data divides into data blocks.
- ❖ A data block default size is 64 MBs



A Hadoop cluster example, and the replication of data blocks in racks for two students of IDs 96 and 1025



Hadoop HDFS features are

- (i) **Create, append, delete, rename and attribute modification functions**
- (ii) **Content of individual file cannot be modified or replaced but appended with new data at the end of the file.**
- (iii) **Write once but read many times during usages and processing.**
- (iv) **Average file size can be more than 500 MB.**

Hadoop Physical Organization

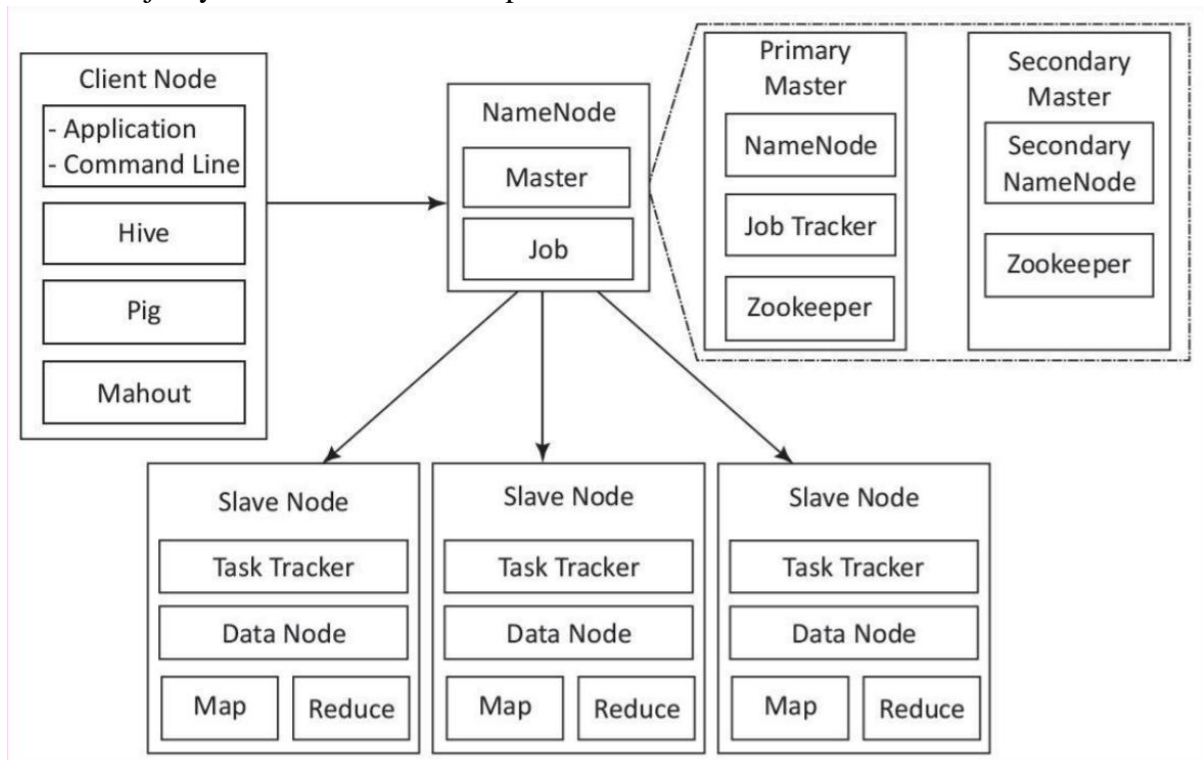
Conventional file system

- ❖ Uses directories
- ❖ Each directory consists of folders
- ❖ Each folder consists of files
- ❖ When data processes, the data sources identify by pointers for the resources
- ❖ Resource Pointers at data-dictionary.
- ❖ Master tables at the dictionary store at a central location
- ❖ The centrally stored tables enable administration easier when the data sources change during processing

Hadoop Physical Organization

- ❖ Similarly, the identification of datablocks, DataNodes and Racks using MasterNodes (NameNodes) is needed for processing the data at slave nodes.
- ❖ . HDFS use the NameNodes and DataNodes
- ❖ A NameNode stores the file's meta data.

- ❖ Meta data gives information about the file of user application, but does not participate in the computations.
- ❖ The DataNode stores the actual data files in the data blocks.
- ❖ Few nodes in a Hadoop cluster act as NameNodes, termed as MasterNodes
- ❖ Different configuration supporting high DRAM and processing power
- ❖ Masters use much less local storage
- ❖ Majority of the nodes in Hadoop cluster act as DataNodes and TaskTrackers.



Clients as the users run the application with the help of Hadoop ecosystem projects. For example, Hive, Mahout and Pig are the ecosystem's projects.

A single MasterNode provides HDFS, MapReduce and Hbase using threads in small to medium sized clusters. When the cluster size is large, multiple servers are used, such as to balance the load.

HDFS Data Storage: Hadoop Physical Organization

- ❖ The MasterNode fundamentally plays the role of a coordinator.
- ❖ The MasterNode receives client connections, maintains the description of the global file system namespace, and the allocation of file blocks.
- ❖ It also monitors the state of the system in order to detect any failure.
- ❖ The Masters consists of three components NameNode, Secondary NameNode and JobTracker.

NameNode

NameNode stores all the file system related information such as:

1. The file section is stored in which part of the cluster
2. Last access time for the files.
3. User permissions like which user has access to the file.

Secondary NameNode

- ❖ Secondary NameNode is an alternate for NameNode.
- ❖ Secondary node keeps a copy of NameNode meta data.
- ❖ Thus, stored meta data can be rebuilt easily, in case of NameNode failure.
- ❖ The secondary NameNode provides NameNode management services and Zookeeper is used by HBase for metadata storage.

JobTracker

- ❖ The JobTracker coordinates the parallel processing of data.

Drawback of Hadoop1

- ❖ Single NameNode failure in Hadoop1 is an operational limitation.
- ❖ Scaling up was also restricted to scale beyond a few thousands of DataNodes and few number of clusters.
- ❖ However Hadoop2 provides the multiple NameNodes.
- ❖ This enables higher resource availability.

Each MN has the following components:

- ❖ An associated NameNode
- ❖ Zookeeper coordination client (an associated NameNode), functions as a centralized repository for distributed applications.

Zookeeper uses synchronization, serialization and coordination activities.

It enables functioning of a distributed system as a single function.

- ❖ Associated JournalNode (JN).

The JN keeps the records of the state, resources assigned, and intermediate results or execution of application tasks.

Distributed applications can write and read data from a JN.

The system takes care of

One set of resources is in active state.

The other one remains in standby state.

Two masters, one MN1 is in active state and other MN2 is in secondary state. That ensures the availability in case of network fault of an active NameNode NM1.

The Hadoop system then activates the secondary NameNode NM2 and creates a secondary in another MasterNode MN3 unused earlier.

The entries copy from JN1 in MN1 into the JN2, which is at newly active MasterNode MN2.

Therefore, the application runs uninterrupted and resources are available uninterrupted.

HDFS Data Storage: HDFS Commands

Commands for interacting with the files in HDFS require `/bin/hdfs dfs`

`<args>`, where `args` stands for the command arguments.

-copyToLocal is the command for copying a file at HDFS to the local.

-cat is command for copying to standard output (stdout).

All Hadoop commands are invoked by the ***bin/Hadoop*** script.

% Hadoop fsck I -files -blocks

HDFS shell command	Example of usage
-mkdir	Assume stu_filesdir is a directory of student files in Example 2.2. Then command for creating the directory is <code>\$Hadoop hdfs-mkdir/user/stu_filesdir</code> creates the directory named stu_files_dir
-put	Assume file stuData_id96 to be copied at stu_filesdir directory in Example 2.2. Then <code>\$Hadoop hdfs-put stuData_id96 /user/ stu_filesdir</code> copies file for student of id96 into stu_filesdir directory
-ls	Assume all files to be listed. Then <code>\$hdfs hdfs dfs-ls</code> command does provide the listing.
-cp	Assume stuData_id96 to be copied from stu_filesdir to new students' directory newstu_filesDir. Then <code>\$Hadoop hdfs-cp stuData_id96 /user/stu_filesdir newstu_filesDir</code> copies file for student of ID 96 into stu_filesdir directory

MAPREDUCE FRAMEWORK AND PROGRAMMING MODEL

MapReduce is a programming model for distributed computing.

Mapper means software for doing the assigned task after organizing the data blocks imported using the key.

Reducer means software for reducing the mapped data by using the aggregation, query or user-specified function.

Aggregation function means the function that groups the values of multiple rows together to result a single value of more significant meaning or measurement.

For example, function such as count, sum, maximum, minimum, deviation and standard deviation.

Querying function means a function that finds the desired values.

For example, function for finding a best student of a class who has shown the best performance in examination.

MapReduce allows writing applications to process reliably the huge amounts of data, in parallel, on large clusters of servers.

Features of MapReduce framework are

1. Provides automatic parallelization and distribution of computation based on several processors
2. Processes data stored on distributed clusters of DataNodes and racks
3. Allows processing large amount of data in parallel.

MAPREDUCE FRAMEWORK AND PROGRAMMING MODEL: Hadoop MapReduce Framework

MapReduce provides two important functions.

- ❖ The distribution of job based on client application task or users query to various nodes within a cluster.

- ❖ The second function is organizing and reducing the results from each node into a cohesive response to the application or answer to the query.

Daemon refers to a highly dedicated program that runs in the background in a system.

MapReduce runs as per assigned Job by JobTracker, which keeps track of the job submitted for execution and runs TaskTracker for tracking the tasks.

MapReduce programming enables job scheduling and task execution as follows

- A client node submits a request of an application to the JobTracker.
- A JobTracker is a Hadoop daemon (background program).

The following are the steps on the request to MapReduce:

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

The job execution is controlled by two types of processes in MapReduce:

A single master process called **JobTracker** is one.

This process coordinates all jobs running on the cluster and assigns map and reduce tasks to run on the TaskTrackers.

The second is a number of subordinate processes called **TaskTrackers**.

These processes run assigned tasks and periodically report the progress to the JobTracker.

Figure 2.4 showed the job execution model of MapReduce.

Here the JobTracker schedules jobs submitted by clients, keeps track of TaskTrackers and maintains the available Map and Reduce slots.

The JobTracker also monitors the execution of jobs and tasks on the cluster.

The TaskTracker executes the Map and Reduce tasks, and reports to the JobTracker

MAPREDUCE FRAMEWORK AND PROGRAMMING MODEL: Hadoop MapReduce Framework

MapReduce program can be written in any language including JAVA, C++ PIPES or Python.

Map function of MapReduce program do mapping to compute the data and convert the data into other data sets (distributed in HDFS).

After the Mapper computations finish, the Reducer function collects the result of map and generates the final output result.

MapReduce program can be applied to any type of data, i.e., structured or unstructured stored in HDFS.

The input data is in the form of file or directory and is stored in the HDFS.

The MapReduce program performs two jobs on this input data, the Map job/Phase and the Reduce job/Phase.

The map job takes a set of data and converts it into another set of data.

The individual elements are broken down into tuples (key/value pairs) in the resultant set of data.

The reduce job takes the output from a map as input and combines the data tuples into a smaller set of tuples.

Map and reduce jobs run in isolation from one another.

The reduce job is always performed after the map job

MapReduce and YARN
based resource scheduling
splits the jobs into subtasks
which run across number of
servers in parallel.

MAPREDUCE FRAMEWORK AND PROGRAMMING MODEL: HADOOP YARN

YARN is a resource management platform.

It manages computer resources.

The platform is responsible for providing the computational resources, such as CPUs, memory, network I/O which are needed when an application executes.

An application task has a number of sub-tasks.

YARN manages the schedules for running of the sub-tasks.

Each sub-task uses the resources in allotted time intervals.

YARN separates the resource management and processing components.

YARN stands for Yet Another Resource Negotiator.

An application consists of a number of tasks.

Each task can consist of a number of sub-tasks (threads), which run in parallel at the nodes in the cluster.

YARN enables running of multi-threaded applications.

YARN manages and allocates the resources for the application sub-tasks and submits the resources for them at the Hadoop system.

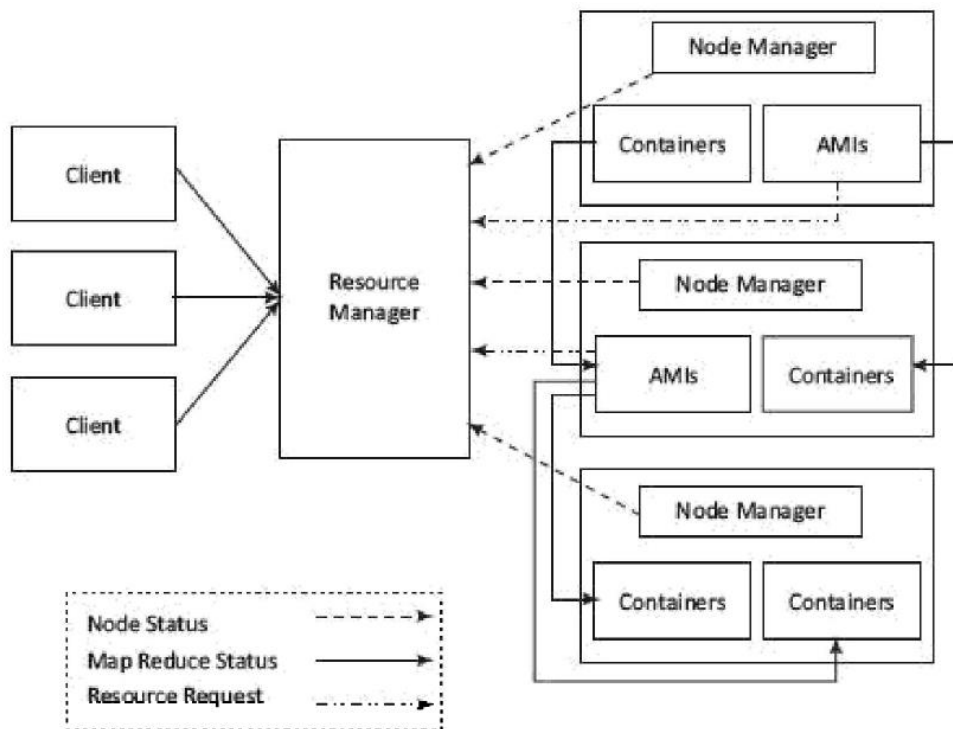
MAPREDUCE FRAMEWORK AND PROGRAMMING MODEL: Hadoop 2

Execution Model

Figure 2.5 shows the YARN-based execution model.

The figure shows the YARN components-Client, Resource Manager (RM), Node Manager (NM), Application Master (AM) and Containers.

Figure 2.5 also illustrates YARN components namely, Client, Resource Manager (RM), Node Manager (RM), Application Master (AM) and Containers.



List of actions of YARN resource allocation and scheduling functions are:

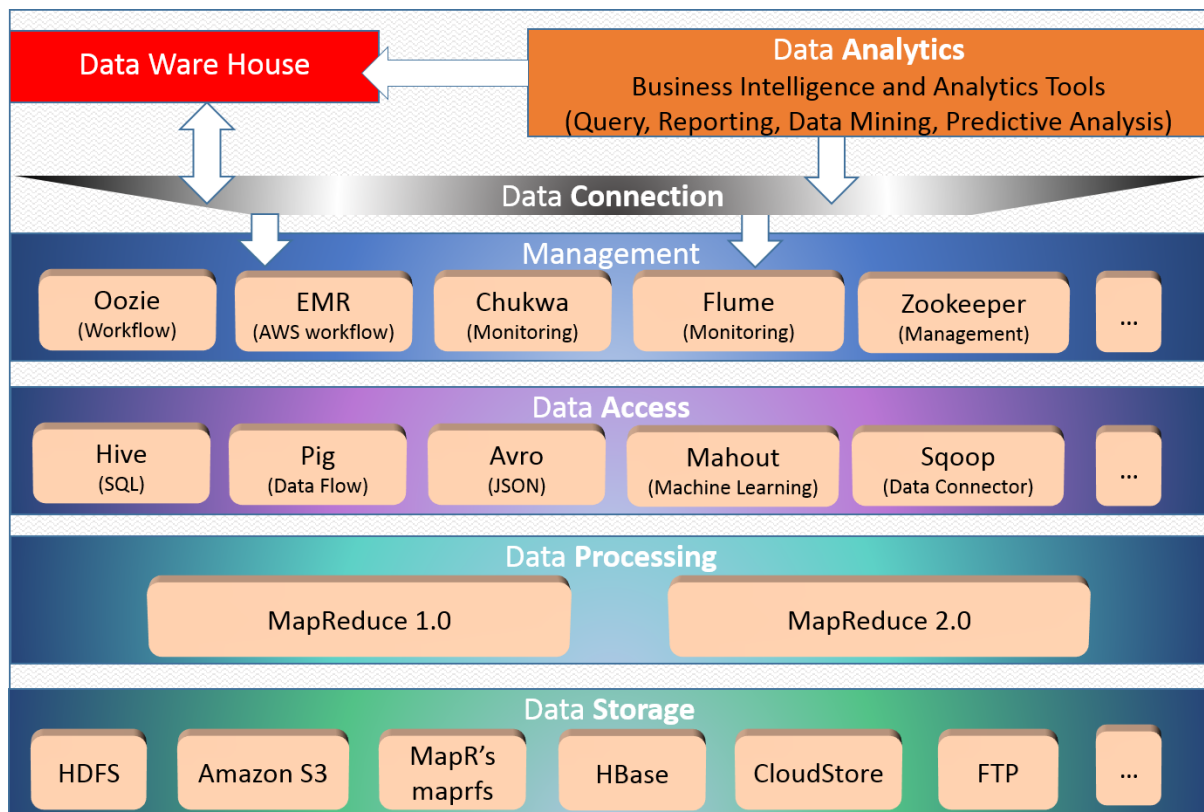
- ❖ A MasterNode has two components: (i) Job History Server and (ii) Resource Manager(RM).
- ❖ A Client Node submits the request of an application to the RM. The RM is the master. One RM exists per cluster. The RM keeps information of all the slave NMs. Information is about the location (Rack Awareness) and the number of resources (data blocks and servers) they have. The RM also renders the Resource Scheduler service that decides how to assign the resources. It, therefore, performs resource management as well as scheduling.
- ❖ Multiple NMs are at a cluster. An NM creates an AM instance (AMI) and starts up. The AMI initializes itself and registers with the RM. Multiple AMIs can be created in an AM.

List of actions of YARN resource allocation and scheduling functions are:

- ❖ The AMI performs role of an Application Manager (ApplM), that estimates the resources requirement for running an application program or sub-task. The ApplMs send their requests for the necessary resources to the RM. Each NM includes several containers for uses by the subtasks of the application.
- ❖ NM is a slave of the infrastructure. It signals whenever it initializes. All active NMs send the controlling signal periodically to the RM signaling their presence.

HADOOP DISTRIBUTED FILE SYSTEM (HDFS)

The functionalities of the ecosystem tools and components.



HADOOP ECOSYSTEM TOOLS

Ecosystem Tool	Functionalities
ZooKeeper - Coordination service	Provisions high-performance coordination service for distributed running of applications (Section 2.6.1.1)
Avro-Data serialization and transfer utility	Provisions data serialization during data transfer between application and Hadoop (Section 2.4.1)
Oozie	Provides a way to package and bundles multiple coordinator and workflow jobs (Section 2.6.1.2)
Sqoop (SQL-to-Hadoop)-A data-transfer software	Provisions for data-transfer between data stores such as relational DBs and Hadoop (Section 2.6.1.3)
Flume - Large data transfer utility	Provisions for reliable data transfer and provides for recovery in case of failure of applications, such as related to social-media messages (Section 2.6.1.4)
Ambari-A web-based tool	Provisions, monitors, manages, and viewing of functioning of the cluster, MapReduce jobs (Section 2.6.2)
Ecosystem Tool	Functionalities

ZooKeeper - Coordination service	Provisions high-performance coordination service for distributed running of (Section 2.6.1.1)
Avro-Data serialization and transfer utility	Provisions data serialization during data transfer between application and process (Section 2.4.1)
Oozie	Provides a way to package and bundles multiple coordinator and workflow jobs (Section 2.6.1.2)
Sqoop (SQL-to-Hadoop)-A data-transfer software	Provisions for data-transfer between data stores such as relational DBs and Hadoop (Section 2.6.1.3)
Flume - Large data transfer utility	Provisions for reliable data transfer and provides for recovery in case of failure for applications, such as related to social-media messages (Section 2.6.1.4)
Ambari-A web-based tool	Provisions, monitors, manages, and viewing of functioning of the cluster, MapReduce jobs (Section 2.6.2)

HADOOP ECOSYSTEM TOOLS: *Zookeeper*

Zookeeper in Hadoop behaves as a centralized repository where distributed applications can write data at a node called JournalNode and read the data out of it.

Zookeeper uses synchronization, serialization and coordination activities.

It enables functioning of a distributed system as a single function.

ZooKeeper's main coordination services are:

Name service -A name service maps a name to the information associated with that name. For example, DNS service is a name service that maps a domain name to an IP address.

Similarly, name keeps a track of servers or services those are up and running, and looks up their status by name in name service.

Concurrency control - Concurrent access to a shared resource may cause inconsistency of the resource.

A concurrency control algorithm accesses shared resource in the distributed system and controls concurrency.

ZooKeeper's main coordination services are:

Configuration management - A requirement of a distributed system is a central configuration manager.

A new joining node can pick up the up-to-date centralized configuration from the ZooKeeper coordination service as soon as the node joins the system.

Failure - Distributed systems are susceptible to the problem of node failures. This requires implementing an automatic recovering strategy by selecting some alternate node for processing (Using two MasterNodes with a NameNode each).

HADOOP ECOSYSTEM TOOLS: *Oozie*

- Apache Oozie is an open-source project of Apache that schedules Hadoop jobs.

-
- Analysis of Big Data requires creation of multiple jobs and sub-tasks in a process.
- Oozie design provisions the scalable processing of multiple jobs.
- Thus, Oozie provides a way to package and bundle multiple coordinator and workflow jobs, and manage the lifecycle of those jobs.

The two basic Oozie functions are:

1. Oozie workflow jobs are represented as Directed Acrylic Graphs (DAGs), specifying a sequence of actions to execute.
2. Oozie coordinator jobs are recurrent Oozie workflow jobs that are triggered by time and data availability.

Oozie provisions for the following:

1. Integrates multiple jobs in a sequential manner
2. Stores and supports Hadoop jobs for MapReduce, Hive, Pig, and Sqoop
3. Runs workflow jobs based on time and data triggers.

HADOOP ECOSYSTEM TOOLS: *Sqoop*

Apache Sqoop is a tool that is built for loading efficiently the voluminous amount of data between Hadoop and external data repositories that resides on enterprise application servers or relational databases.

Sqoop works with relational databases such as Oracle, MySQL, PostgreSQL and DB2.

Sqoop provides the mechanism to import data from external Data Stores into HDFS.

Sqoop relates to Hadoop eco-system components, such as Hive and HBase.

Sqoop can extract data from Hadoop or other ecosystem components.

Sqoop provides command line interface to its users.

Sqoop can also be accessed using Java APIs.

Sqoop exploits MapReduce framework to import and export the data, and transfers for parallel processing of sub-tasks.

Sqoop provisions for fault tolerance.

Parallel transfer of data results in parallel results and fast data transfer.

Sqoop initially parses the arguments passed in the command line and prepares the map task. The map task initializes multiple Mappers depending on the number supplied by the user in the command line.

Each map task will be assigned with part of data to be imported based on key defined in the command line.

Sqoop distributes the input data equally among the Mappers.

Then each Mapper creates a connection with the database using JDBC and fetches the part of data assigned by Sqoop and writes it into HDFS/Hive/HBase as per the choice provided in the command line.

HADOOP ECOSYSTEM TOOLS: *Flume*

Apache Flume provides a distributed, reliable and available service. Flume efficiently collects, aggregates and transfers a large amount of streaming data into HDFS.

Flume enables upload of large files into Hadoop clusters.

The features of flume include robustness and fault tolerance.

Flume provides data transfer which is reliable and provides for recovery in case of failure.

Flume is useful for transferring a large amount of data in applications related to logs of network traffic, sensor data, geo-location data, e-mails and social-media messages.

Apache Flume has the following four important components:

1. Sources which accept data from a server or an application.

2. Sinks which receive data and store it in HDFS repository or transmit the data to another source. Data units that are transferred over a **channel** from source to sink are called events.

HADOOP ECOSYSTEM TOOLS: Ambari

Apache Ambari is a management platform for Hadoop. It is open source.

Ambari enables an enterprise to plan, securely install, manage and maintain the clusters in the Hadoop.

Ambari provisions for advanced cluster security capabilities, such as Kerberos Ambari.

Features of Ambari and associated components are :

1. Simplification of installation, configuration and management
2. Enables easy, efficient, repeatable and automated creation of clusters
3. Manages and monitors scalable clustering
4. Provides an intuitive Web User Interface and REST API. The provision enables automation of cluster operations.
5. Visualizes the health of clusters and critical metrics for their operations
6. Enables detection of faulty node links
7. Provides extensibility and customizability.

Hadoop Administration

Hadoop large clusters pose a number of configuration and administration challenges.

Administrator procedures enable managing and administering Hadoop clusters, resources and associated Hadoop ecosystem components (Figure 2.2).

Administration includes installing and monitoring clusters.

Ambari also provides a centralized setup for security.

This simplifies the administering complexities and configures security of clusters across the entire platform.

Hadoop Administration

Ambari helps automation of the setup and configuration of Hadoop using Web User Interface and REST APIs.

The console is similar to web UI at Ambari.

The console enables visualization of the cluster health, HDFS directory structure, status of MapReduce tasks, review of log records and access application status.

Hadoop Administration

Single harmonized view on console makes administering the task easier.

Visualization can be up to individual components level on drilling down.

Nodes addition and deletion are easy using the console.

The console enables built-in tools for administering.

Web console provides a link to server tools and open-source components associated with those.

HADOOP ECOSYSTEM TOOLS: HBase

Similar to database, HBase is an Hadoop system database. HBase was created for large tables.

HBase is an open-source, distributed, versioned and non-relational (NoSQL) database.

Features of HBase features are:

1. Uses a partial columnar data schema on top of Hadoop and HDFS.
2. Supports a large table of billions of rows and millions of columns.
3. Provides small amounts of information, called sparse data taken from large data sets which are storing empty or presently not-required data.
4. Supports data compression algorithms.
5. Provisions in-memory column-based data transactions.
6. Accesses rows serially and does not provision for random accesses and write into the rows.
7. Provides random, real-time read/write access to Big Data.
8. Fault tolerant storage due to automatic failure support between DataNodes servers.
9. Similarity with Google BigTable.

HBase is written in Java. It stores data in a large structured table.

Hbase provides scalable distributed Big Data Store.

HBase data stores as key-value pairs.

HBASE applies a partial columnar scheme on top of the Hadoop and HDFS.

An HBase column represents an attribute of an object.

HADOOP ECOSYSTEM TOOLS: Hive

Apache Hive is an open-source data warehouse software. Hive facilitates reading, writing and managing large datasets which are at distributed Hadoop files. Hive uses SQL. Hive puts a partial SQL interface in front of Hadoop. Hive design provisions for batch processing of large sets of data. An application of Hive is for managing weblogs. Hive does not process real-time queries and does not update row-based data tables. Hive also enables data serialization/ deserialization and increases flexibility in schema design by including a system catalog called Hive Metastore. HQL also supports custom MapReduce scripts to be plugged into queries. Hive supports different storage types, such as text files, sequence files (consisting of binary key/value pairs) and RCFiles (Record Columnar Files), ORC (optimized row columnar) and HBase.

Three major functions of Hive are data summarization, query and analysis. Hive basically interacts with structured data stored in HDFS with a query language known as HQL (Hive Query Language) which is similar to SQL. HQL translates SQL-like queries into MapReduce jobs executed on Hadoop automatically.

HADOOP ECOSYSTEM TOOLS: Pig

Apache Pig is an open source, high-level language platform. Pig was developed for analyzing large-data sets. Pig executes queries on large datasets that are stored in HDFS using Apache Hadoop. The language used in Pig is known as Pig Latin. Pig Latin language is similar to SQL query language but applies on larger datasets.

Additional features of Pig are :

- (i) Loads the data after applying the required filters and dumps the data in the desired format.
- (ii) Requires Java runtime environment for executing Pig Latin programs.
- (iii) Converts all the operations into map and reduce tasks.
The tasks run on Hadoop.
- (iv) Allows concentrating upon the complete operation, irrespective of the individual Mapper and Reducer functions to produce the output results.

HADOOP ECOSYSTEM TOOLS: Mahout

Mahout is a project of Apache with library of scalable machine learning algorithms. Apache implemented Mahout on top of Hadoop. Apache used the MapReduce paradigm. Machine learning is mostly required to enhance the future performance of a system based on the previous outcomes.

Mahout provides the learning tools to automate the finding of meaningful patterns in the Big Data sets stored in the HDFS.

Mahout supports four main areas:

- Collaborative data-filtering that mines user behavior and makes product recommendations.
- Clustering that takes data items in a particular class, and organizes them into naturally occurring groups, such that items belonging to the same group are similar to each other.
- Classification that means learning from existing categorizations and then assigning the future items to the best category.
- Frequent item-set mining that analyzes items in a group and then identifies which items usually occur together.

HADOOP DISTRIBUTED FILE SYSTEM (HDFS) BASICS (T2)

HDFS Design Features

The Hadoop Distributed file system (HDFS) was designed for *Big Data processing*. Although capable of supporting many users simultaneously, HDFS is not designed as a true *parallel file system*.

Rather, the design assumes a *large file write-once/read-many* model.

The design of HDFS is based on the design of the Google File System (GFS).

HDFS is designed for data streaming where large amounts of data are read from disk in bulk. The HDFS block size is typically 64MB or 128MB.

HDFS FEATURES

- ❖ The write-once/read-many design is intended to facilitate streaming reads.
- ❖ Files may be appended, but random seeks are not permitted.
- ❖ There is no caching of data.
- ❖ Converged data storage and processing happen on the same server nodes.
- ❖ “Moving computation is cheaper than moving data.”

HDFS FEATURES

- ❖ The write-once/read-many design is intended to facilitate streaming reads.
- ❖ Files may be appended, but random seeks are not permitted.
- ❖ There is no caching of data.
- ❖ Converged data storage and processing happen on the same server nodes.
- ❖ “Moving computation is cheaper than moving data.”

HDFS FEATURES

- ❖ A reliable file system maintains multiple copies of data across the cluster.
- ❖ Consequently, failure of a single node (or even a rack in a large cluster) will not bring down the file system.
- ❖ A specialized file system is used, which is not designed for general use.

HDFS Components

There are two types of HDFS installation

Single Node Installation.

Multiple Node Installation.

The designed of HDFS is based on three types of nodes

Name Node : Only one node per cluster

Date Node : Multiple Nodes

Secondary Node : Checkpoint or Backup Node

client/NameNode/DataNode interaction is provided in Figure 3.1.

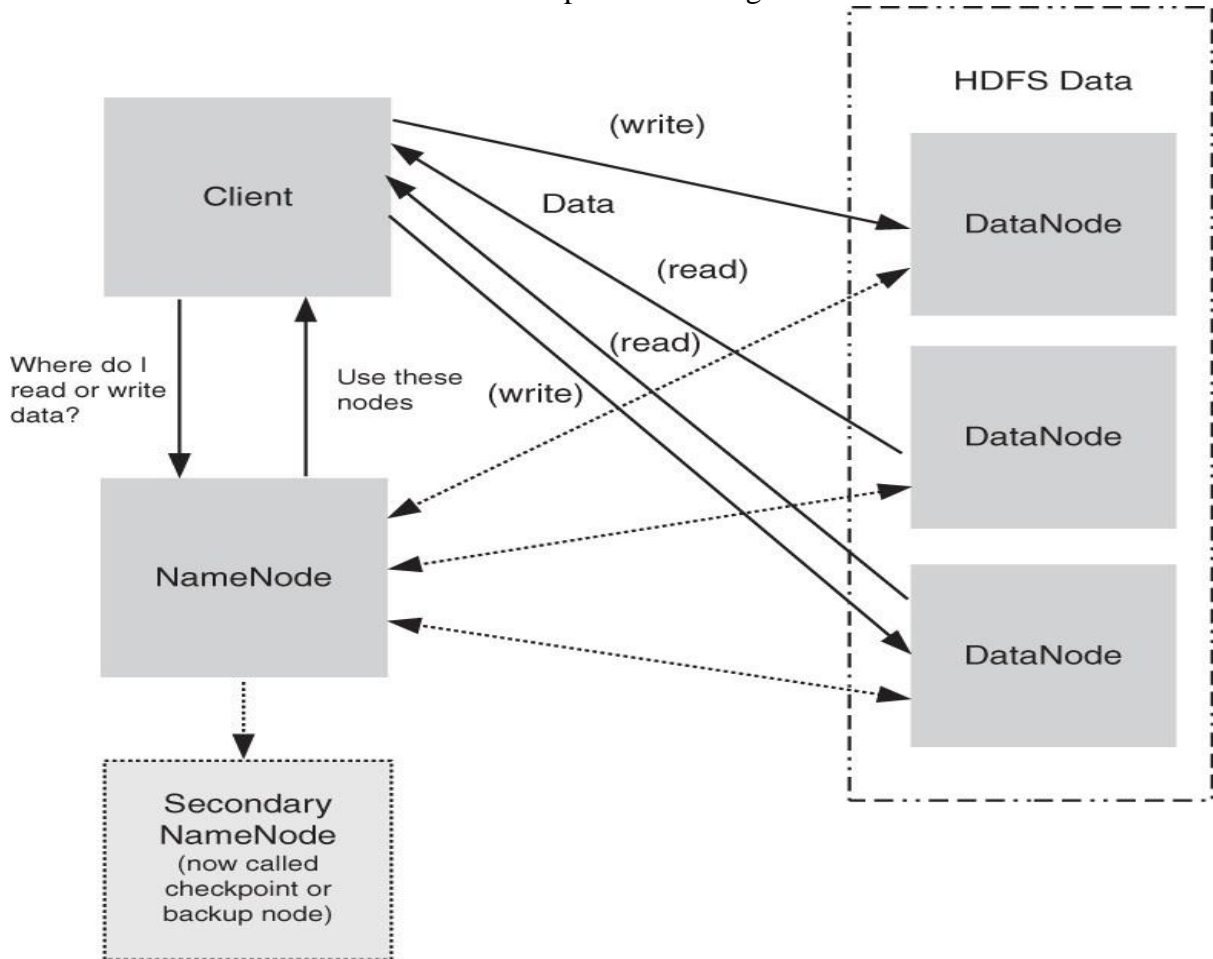


Figure 3.1 Various system roles in an HDFS deployment

NAME NODE (MASTER NODE)

Name Node Manages all the meta data needed to store and retrieve actual data from the Data Nodes.

Data is not stored in Namenode.

Master Node manages the file system namespace and regulates access to files by clients.

Name Node perform operation such as opening, closing, delete, renaming files and directories.

NAME NODE (MASTER NODE)

NameNode also handles mapping of blocks to DataNodes and Handle Data Node Failure.

Name Node Manages block creation, mapping of blocks, deletion and replication.

It maintains two file to track changes to the metadata: **FsImage (New File Entry)** and **EditLogs (Deletion and Modification)**

DATA NODE (SLAVE NODE)

DataNodes are the slave nodes in HDFS.

Unlike NameNode, DataNode is a commodity hardware, that is, a non-expensive system which is not of high quality or high-availability.

The DataNode is a block server that stores the data in the local file ext3 or ext4.

DATA NODE (SLAVE NODE)

The actual data is stored on DataNodes and process which runs on each slave machines.

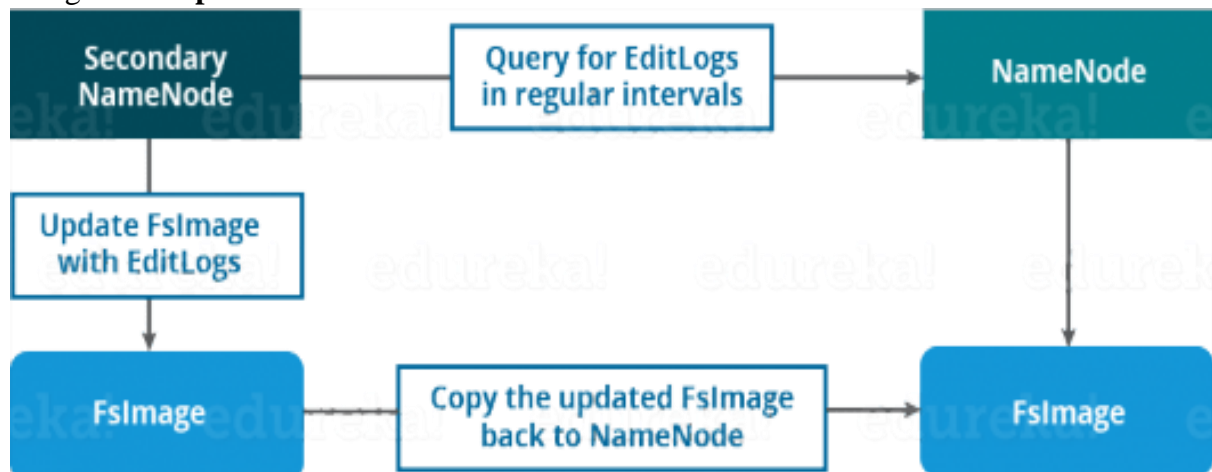
The DataNodes perform the low-level read and write requests from the file system's clients.

They send heartbeats to the NameNode periodically to report the overall health of HDFS, by default, this frequency is set to 3 seconds.

SECONDARY NAMENODE

The Secondary NameNode works concurrently with the primary NameNode as a **helper daemon**.

And don't be confused about the Secondary NameNode being a **backup NameNode because it is not**.



SECONDARY NAMENODE

The Secondary NameNode is one which constantly reads all the file systems and metadata from the RAM of the NameNode and writes it into the hard disk or the file system.

It is responsible for combining the EditLogs with FsImage from the NameNode.

It downloads the EditLogs from the NameNode at regular intervals and applies to FsImage.

The new FsImage is copied back to the NameNode, which is used whenever the NameNode is started the next time

The various roles in HDFS can be summarized as follows:

HDFS uses a master/slave model designed for large file reading/streaming.

The NameNode is a metadata server or “data traffic cop.”

HDFS provides a single namespace that is managed by the NameNode.

Data is redundantly stored on DataNodes; there is no data on the NameNode.

The SecondaryNameNode performs checkpoints of NameNode file system's state but is not a failover node.

- ❖ Block Replication
- ❖ Safe Mode
- ❖ Rack Awareness
- ❖ High Availability
- ❖ Back Up

Block Replication

When HDFS writes a file, it is replicated across the cluster.

The amount of replication is based on the value of *dfs.replication* in the *hdfs-site.xml* file.

This default value can be overruled with the *hdfs dfs-setrep* command.

For Hadoop clusters containing more than eight DataNodes, the replication value is usually set to 3.

In a Hadoop cluster of eight or fewer DataNodes but more than one DataNode, a replication factor of 2 is adequate.

For a single machine, the replication factor is set to 1.

Various HDFS user commands.

- *List Files in HDFS*
- *Make a Directory in HDFS*
- *Copy Files to HDFS*
- *Copy Files from HDFS*
- *Copy Files within HDFS*
- *Delete a File within HDFS*
- **List Files in HDFS**
- To list the files in the root HDFS directory, enter the following command:
 - Syntax: `$ hdfs dfs -ls /`
- Output:
- Found 2 items
- `drwxrwxrwx - yarn hadoop 0 2015-04-29 16:52 /app-logs`
- `drwxr-xr-x - hdfs hdfs 0 2015-04-21 14:28 /apps`

List Files in HDFS

- To list files in your home directory, enter the following command:
- Syntax: `$ hdfs dfs -ls`
- Output:
- Found 2 items
- `drwxr-xr-x - hdfs hdfs 0 2015-05-24 20:06 bin`
- `drwxr-xr-x - hdfs hdfs 0 2015-04-29 16:52 examples`

❖ Make a Directory in HDFS

To make a directory in HDFS, use the following command. As with the `-ls` command, when no path is supplied, the user's home directory is used

Syntax: `$ hdfs dfs -mkdir stuff`

`$ hdfs dfs -ls`

Copy Files to HDFS

To copy a file from your current local directory into HDFS, use the following command. If a full path is not supplied, your home directory is assumed. In this case, the file test is placed in the directory stuff that was created previously.

Syntax: `$ hdfs dfs -put test stuff`

The file transfer can be confirmed by using the `-ls` command:

Syntax: `$ hdfs dfs -ls stuff`

Output:

Found 1 items

`-rw-r--r-- 2 hdfs hdfs 12857 2015-05-29 13:12 stuff/test`

Copy Files from HDFS

Files can be copied back to your local file system using the following command.
In this case, the file we copied into HDFS, test, will be copied back to the current local directory with the name test-local.

Syntax: `$ hdfs dfs -get stuff/test test-local`

Copy Files within HDFS

The following command will copy a file in HDFS:

Syntax: `$ hdfs dfs -cp stuff/test test.hdfs`

Delete a File within HDFS

The following command will delete the HDFS file test.dhfs that was

Syntax: `$ hdfs dfs -rm test.hdfs`

Module 3

OBJECTIVES

- Introduction,
- NoSQL Data Store,
- NoSQL Data Architecture Patterns,
- NoSQL to Manage Big Data,
- Shared-Nothing Architecture for Big Data Tasks,
- MongoDB Databases,
- Cassandra Databases.

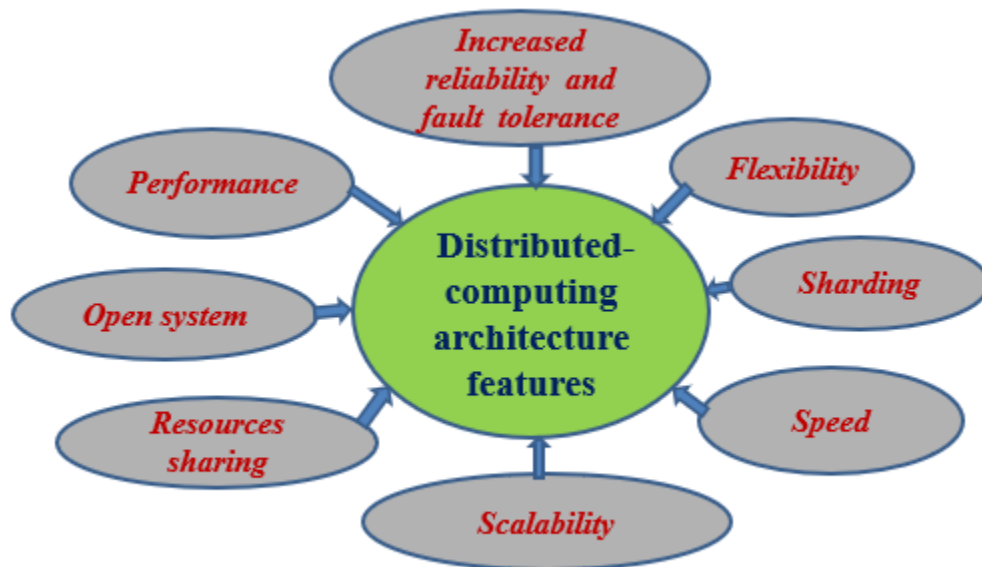
Introduction

Big Data uses distributed systems.

A distributed system consists of multiple data nodes at clusters of machines and distributed software components.

The tasks execute in parallel with data at nodes in clusters.

The computing nodes communicate with the applications through a network.



Increased reliability and fault tolerance

The important advantage of distributed computing system is reliability.

If a segment of machines in a cluster fails then the rest of the machines continue work

Flexibility

Flexibility makes it very easy to install, implement and debug new services in a distributed environment.

Sharding

Sharding is a **method of splitting and storing a single logical dataset in multiple databases**

Speed

Computing power increases in a distributed computing system as shards run parallelly on individual data nodes in clusters independently (no data sharing between shards).

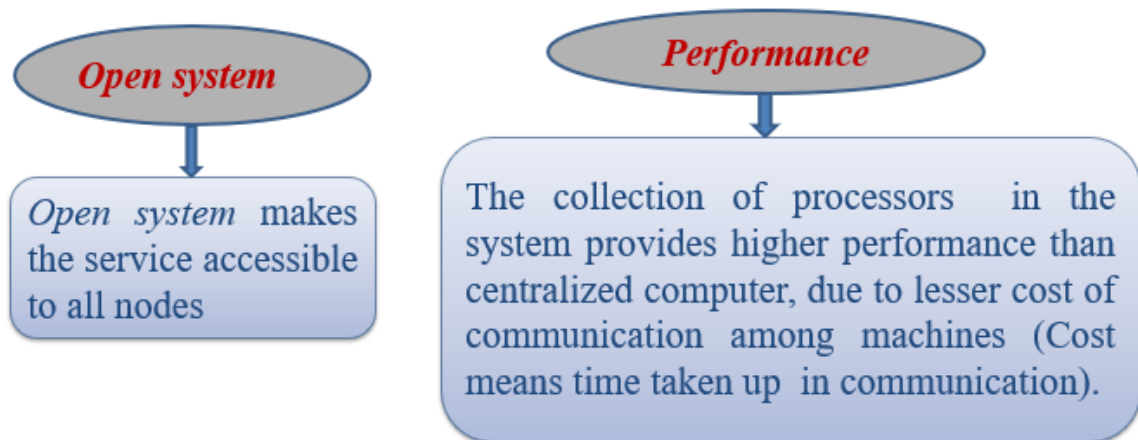
Scalability

When the database expands further, then adding more machines and increasing the number of shards provides horizontal scalability.

Increased computing power and running number of algorithms on the same machines provides vertical scalability.

Resources sharing

Shared resources of memory, machines and network architecture reduce the cost.



The demerits of distributed computing are

- (i) issues in troubleshooting in a larger networking infrastructure,
- (ii) additional software requirements and
- (iii) security risks for data and resources.

The Solution to these demerits is the Big Data solutions with a scalable distributed computing model with shared-nothing architecture.

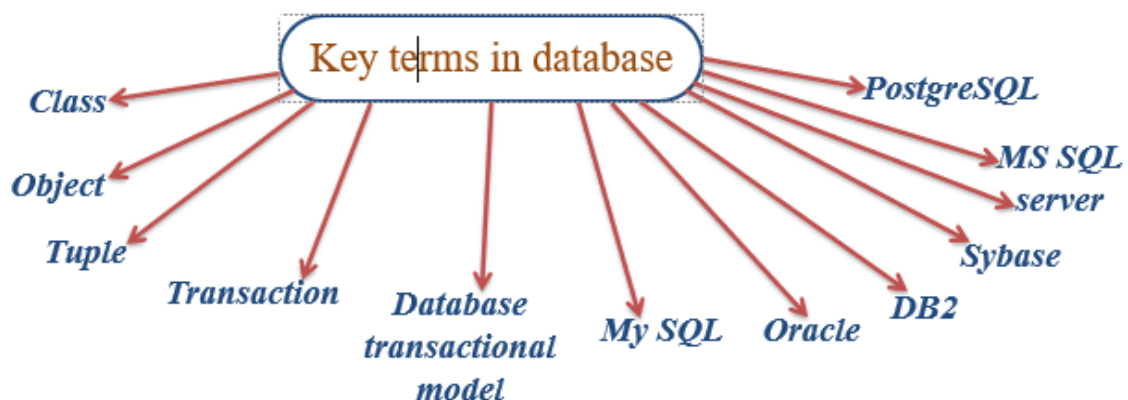
Examples

vMongoDB and

vCassandra.

v

MongoDB and Cassandra DBMS create HDFS compatible distributed data stores and include their specific query processing languages.



Class refers to a template of program codes that is extendable.

Class creates instances, called objects.

A class consists of states (variables), member functions and methods (behavior).

Object is an instance of a class in Java, C++, and other object-oriented languages. Object can be an instance of another object (for example, in JavaScript).

Tuple is an ordered set of data which constitutes a record. For example, one row record in a table. A row in a relational database has column fields or attributes.

Transaction means execution of instructions in two interrelated entities, such as a query and the database.

Database transactional model refers to a model for transactions, such as the one the ACID or BASE properties.

My SQL refers to a widely used open-source database, which excels as a content management server.

Oracle refers to a widely used object-relational DBMS, written in the C++ language that provides applications integration with service-oriented architectures and has high reliability. Oracle has also released the NoSQL database system.

DB2 refers to a family of database server products from IBM with built-in support to handle advanced Big Data analytics

Sybase refers to database server based on relational model for businesses, primarily on UNIX. Sybase was the first enterprise-level DBMS in Linux.

MS SQL server refers to a Microsoft-developed RDBMS for enterprise-level databases that supports both SQL and NoSQL architectures.

PostgreSQL refers to an enterprise-level, object-relational DBMS. PostgreSQL uses procedural languages like Perl and Python, in addition to SQL.

ACID Properties in SQL Transactions

Atomicity of transaction means all operations in the transaction must complete, and if interrupted, then must be undone (rolled back).

For example, if a customer withdraws an amount then the bank in first operation enters the withdrawn amount in the table and in the next operation modifies the balance with new amount available.

Atomicity means both should be completed, else undone if interrupted in between.

A new category of data stores is NoSQL (means Not Only SQL) data stores.

NoSQL is an altogether new approach of thinking about databases, such as schema flexibility, simple relationships, dynamic schemas, auto sharding, replication, integrated caching, horizontal scalability of shards, distributable tuples, semi-structures data and flexibility in approach.

Issues with NoSQL data stores are lack of standardization in approaches, processing difficulties for complex queries, dependence on eventually consistent results in place of consistency in all states.

NoSQL DB does not require specialized RDBMS like storage and hardware for processing. Storage can be on a cloud.

NoSQL records are in non-relational data store systems. They use flexible data models. The records use multiple schemas.

NoSQL data stores are considered as semi-structured data.

NoSQL data store characteristics are as follows:

1. NoSQL is a class of non-relational data storage system with flexible data model.

Examples of NoSQL data-architecture patterns of datasets are key-value pairs, name/value pairs, Column family Big-data store, Tabular data store, Cassandra (used in Facebook/ Apache), HBase, hash table unordered keys using JSON (CouchDB), JSON (PNUTS), JSON (MongoDB), Graph Store, Object Store, ordered keys and semi-structured data storage systems.

2. NoSQL not necessarily has a fixed schema, such as table; do not use the concept of Joins (in distributed data storage systems);

Data written at one node can be replicated to multiple nodes.

Data store is thus fault-tolerant.

The store can be partitioned into unshared shards.

Features in NoSQL Transactions

NoSQL transactions have following features:

(i) Relax one or more of the ACID properties.

(ii) Characterize by two out of three properties (consistency, availability and partitions) of CAP theorem, two are at least present for the application/ service/ process.

(iii) Can be characterized by BASE properties

Big Data NoSQL solutions use standalone-server, master-slave and peer-to-peer distribution models.

Big Data NoSQL Solutions

NoSQL DBs are needed for Big Data solutions.

They play an important role in handling Big Data challenges.

Table 3.1 gives the examples of widely used NoSQL data stores.

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

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

Consistency means all copies have the same value like in traditional DBs.

Availability means at least one copy is available in case a partition becomes inactive or fails.

For example, in web applications, the other copy in the other partition is available.

Partition means parts which are active but may not cooperate (share) as in distributed DBs.

1. Consistency in distributed databases means that all nodes observe the same data at the same time.

Therefore, the operations in one partition of the database should reflect in other related partitions in case of distributed database.

Operations, which change the sales data from a specific showroom in a table should also reflect in changes in related tables which are using that sales data.

2. Availability means that during the transactions, the field values must be available in other partitions of the database so that each request receives a response on success as well as failure. (Failure causes the response to request from the replicate of data). Distributed databases require transparency between one another. Network failure may lead to data unavailability in a certain partition in case of no replication. Replication ensures availability.

3. Partition means division of a large database into different databases without affecting the operations on them by adopting specified procedures.

Partition tolerance: Refers to continuation of operations as a whole even in case of message loss, node failure or node not reachable.

NoSQL: CAP Theorem

In case of any network failure, a choice can be:

- Database must answer, and that answer would be old or wrong data (AP).

- Database should not answer, unless it receives the latest copy of the data (CP).

The CAP theorem implies that for a network partition system, the choice of consistency and availability are mutually exclusive.

CA means consistency and availability, AP means availability and partition tolerance and CP means consistency and partition tolerance.

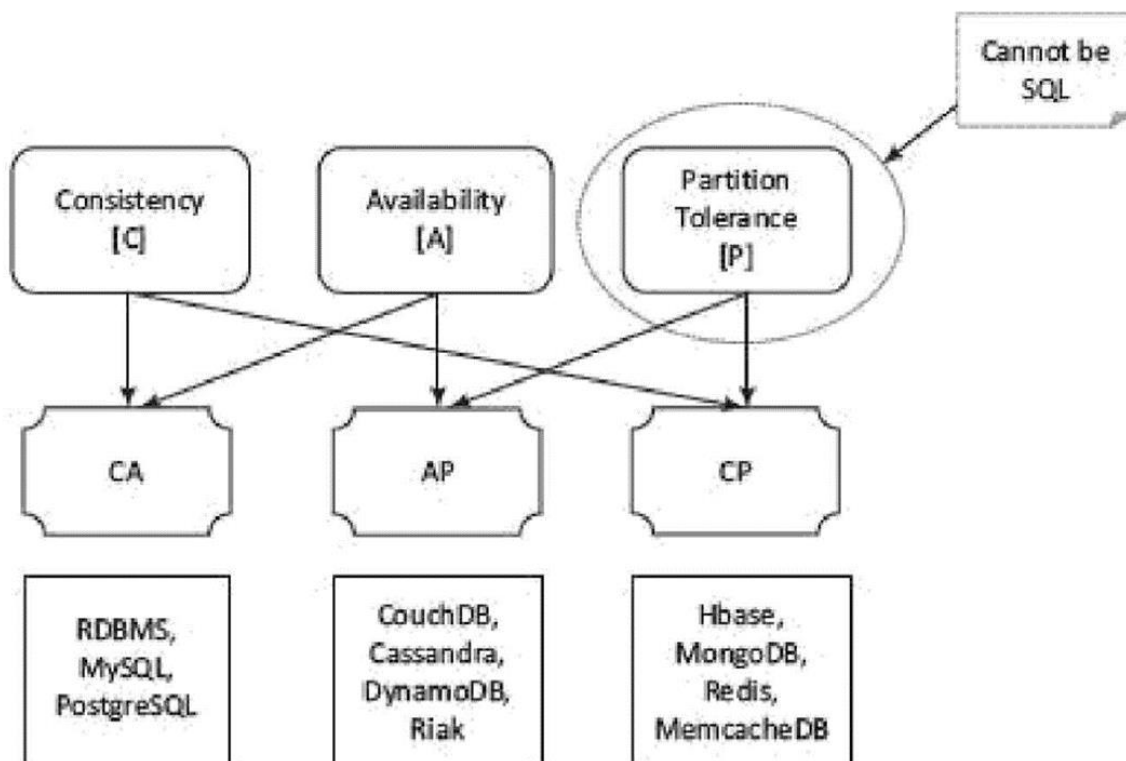


Figure 3.1 CAP theorem in Big Data solutions

NoSQL: Schema-less Models

Schema of a database system refers to designing of a structure for datasets and data structures for storing into the database.

NoSQL data not necessarily have a fixed table schema. The systems do not use the concept of Join (between distributed datasets). A cluster-based highly distributed node manages a single large data store with a NoSQL DB.

Data written at one node replicates to multiple nodes. Therefore, these are identical, fault-tolerant and partitioned into shards. Distributed databases can store and process a set of information on more than one computing nodes.

NoSQL data model offers relaxation in one or more of the ACID properties (Atomicity, consistence, isolation and durability) of the database. Distribution follows CAP theorem.

Figure 3.2 shows characteristics of Schema-less model for data stores. ER stands for entity-relation modelling.

Relations in a database build the connections between various tables of data. For example, a table of subjects offered in an academic programme can be connected to a table of programmes offered in the academic institution. NoSQL

data stores use non-mathematical relations but store this information as an aggregate called metadata.

Metadata refers to data describing and specifying an object or objects. Metadata is a record with all the information about a particular dataset and the inter-linkages.

Metadata helps in selecting an object, specifications of the data and, usages that design where and when.

Metadata specifies access permissions, attributes of the objects and enables additions of an attribute layer to the objects. Files, tables, documents and images are also the objects

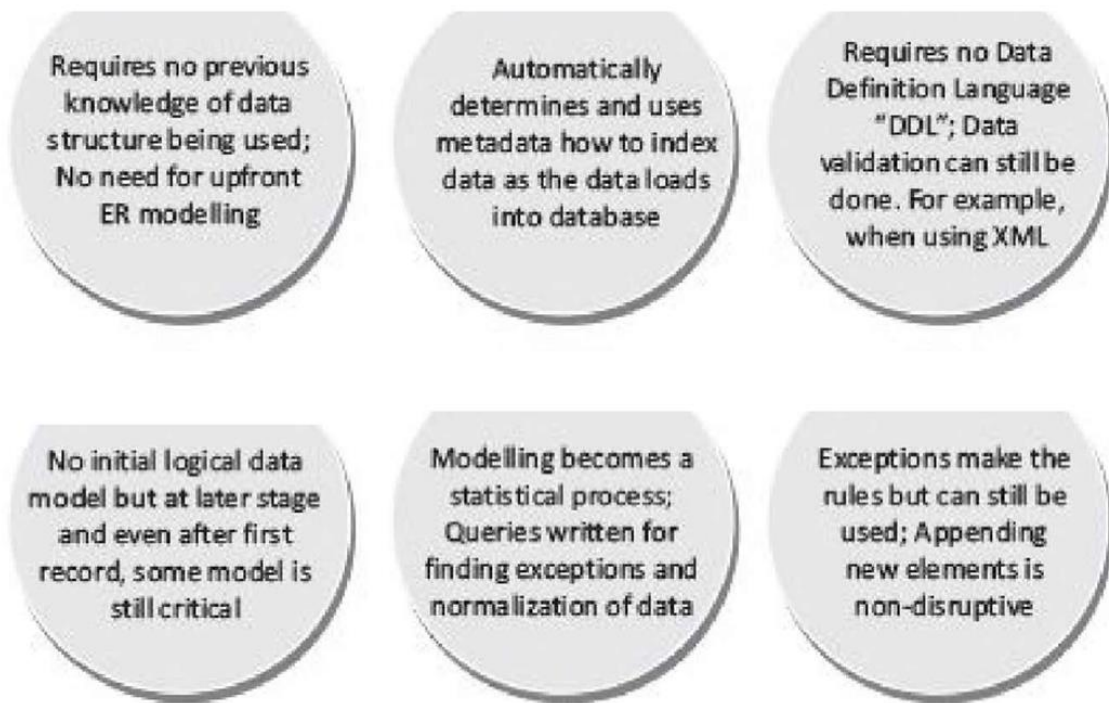


Figure 3.2 Characteristics of Schema-less model

NoSQL data store possess characteristic of increasing flexibility for data manipulation. The new attributes to database can be increasingly added. Late binding of them is also permitted.

BASE is a flexible model for NoSQL data stores. Provisions of BASE increase flexibility. BASE Properties BA stands for basic availability, S stands for soft state and E stands for eventual consistency.

1. Basic availability ensures by distribution of shards (many partitions of huge data store) across many data nodes with a high degree of replication.

Then, a segment failure does not necessarily mean a complete data store unavailability.

2. Soft state ensures processing even in the presence of inconsistencies but achieving consistency eventually. A program suitably takes into account the inconsistency found during processing. NoSQL database design does not consider the need of consistency all along the processing time.

3. Eventual consistency means consistency requirement in NoSQL databases meeting at some point of time in future.

Data converges eventually to a consistent state with no time-frame specification for achieving that.

ACID rules require consistency all along the processing on completion of each transaction.

BASE does not have that requirement and has the flexibility.

BASE model is not necessarily appropriate in all cases but it is flexible and is an alternative to SQL-like adherence to ACID properties.

Use examples of database for the students in various university courses to demonstrate the concept of increasing flexibility in NoSQL DBs.

SOLUTION

Figure 3.3 shows increasing flexibility concept using additional data models

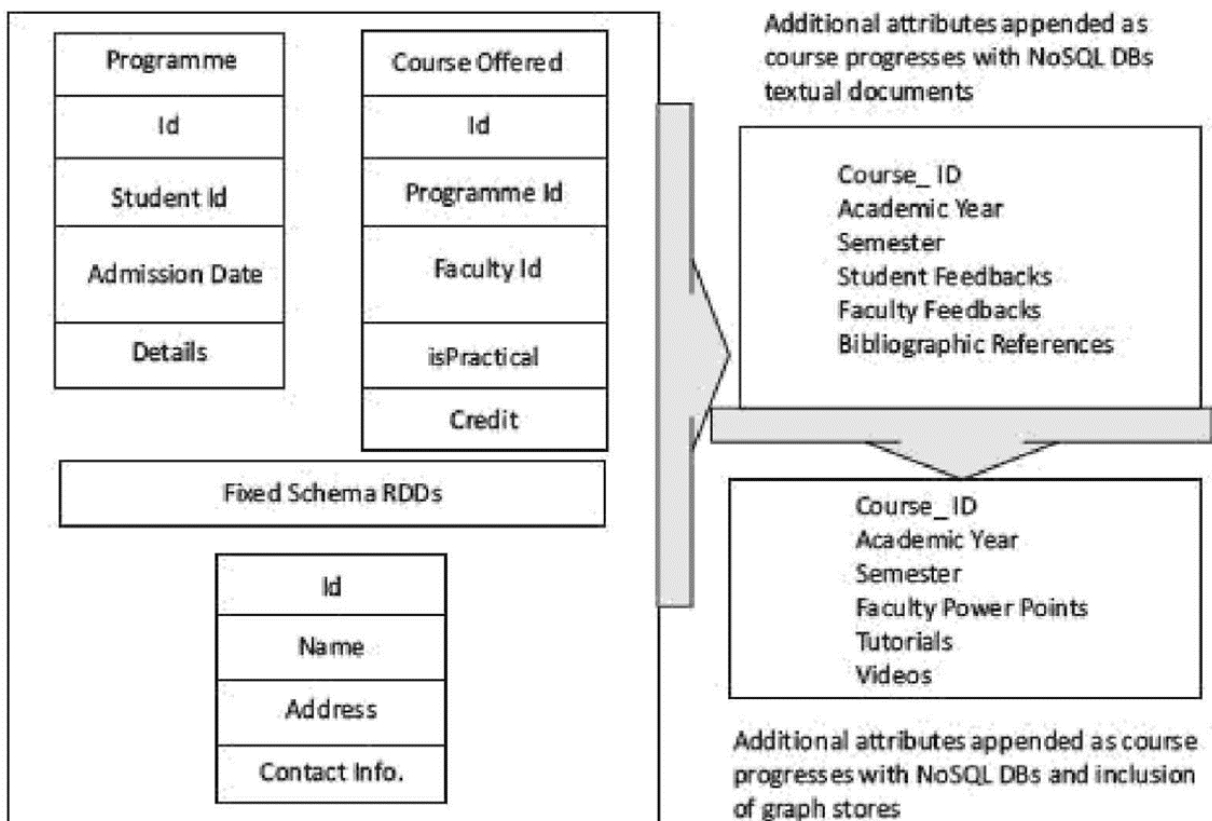


Figure 3.3 Increasing flexibility in NoSQL DB of students

Key-Value store

Document Store

TabularData

Object Data Store

Graph Database

Key-Value store

The simplest way to implement a schema-less data store is to use key-value pairs.

Data retrieval is fast in key-value pairs data store.

Key maps to a large data string or BLOB (Basic Large Object).

Key-value store accesses use a primary key for accessing the values. Therefore, the store can be easily scaled up for very large data.

The concept is similar to a hash table where a unique key points to a particular item(s) of data. Figure 3.4 shows key-value pairs architectural pattern and example of students' database as key-value pairs

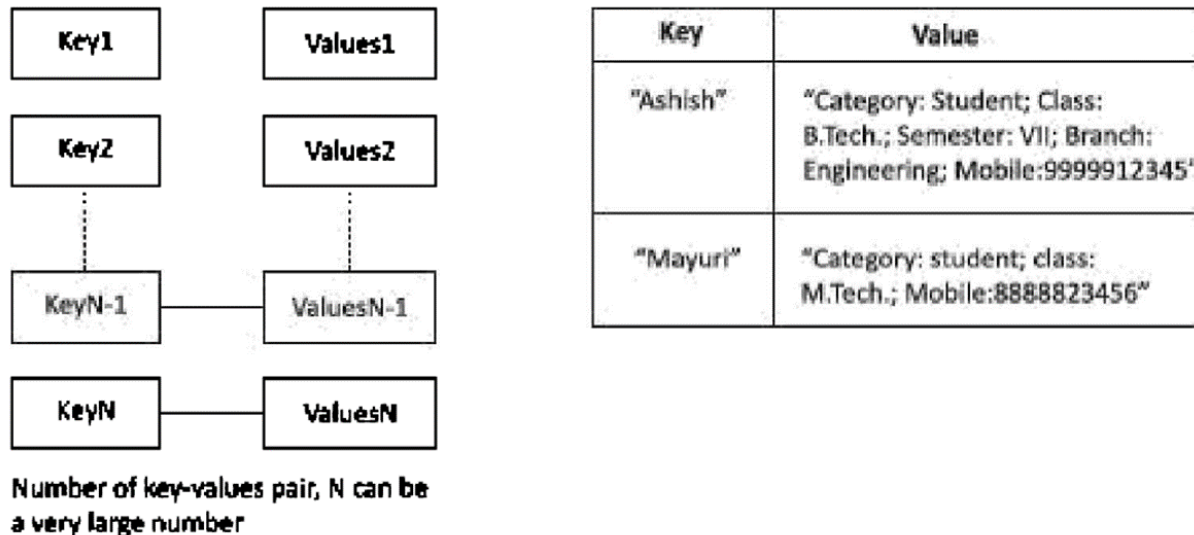


Figure 3.4 Example of key-value pairs in data architectural pattern

Advantages of a key-value store are :

1. The key-value system stores the information as a BLOB of data (such as text, hypertext, images, video and audio) and return the same BLOB when the data is retrieved, querying for key retrieves the values.
2. A query just requests the values and returns the values as a single item.
Values can be of any data type.
3. Key-value store is eventually consistent..
4. Key-value data store may be hierarchical or may be ordered key-value store.
5. Returned values on queries can be used to convert into lists, table• columns, data-frame fields and columns.

6. Have (i) scalability, (ii) reliability, (iii) portability and (iv) low operational cost.

7. The key can be synthetic or auto-generated.

The key is flexible and can be represented in many formats:

- (i) Artificially generated strings created from a hash of a value,
- (ii) Logical path names to images or files,
- (iii) REST web-service calls (request response cycles), and
- (iv) SQL queries.

The key-value store provides client to read and write values using a key

- (i) Get (key), returns the value associated with the key.
- (ii) Put (key, value), associates the value with the key and updates a value if this key is already present.
- (iii) Multi-get (key1, key2, .. 'keyN), returns the list of values associated with the list of keys.
- (iv) Delete (key), removes a key and its value from the data store.

Limitations of key-value store architectural pattern are:

- (i) No indexes are maintained on values, thus a subset of values is not searchable.

(ii) Key-value store does not provide traditional database capabilities, such as atomicity of transactions, or consistency when multiple transactions are executed simultaneously. The application needs to implement such capabilities.

Limitations of key-value store architectural pattern are:

- (i) No indexes are maintained on values, thus a subset of values is not searchable.

(ii) Key-value store does not provide traditional database capabilities, such as atomicity of transactions, or consistency when multiple transactions are executed simultaneously. The application needs to implement such capabilities.

Limitations of key-value store architectural pattern are:

- (iii) Maintaining unique values as keys may become more difficult when the volume of data increases. One cannot retrieve a single result when a key-value pair is not uniquely identified.

- (iv) Queries cannot be performed on individual values.

No clause like 'where' in a relational database usable that filters a result set.

Table 3.2 Traditional relational data model vs. the key-value store model

Traditional relational model	Key-value store model
Result set based on row values	Queries return a single item
Values of rows for large datasets are indexed	No indexes on values
Same data type values in columns	Any data type values

Typical uses of key-value store are:

- (i) Image store,
- (ii) Document or file store,
- (iii) Lookup table, and
- (iv) Query-cache.

key-value pairs used in

- Riak
- DynamoDB
- Redis
- Memcached and its flavours,
- Berkeley DB, upscaledb
- Project Voldemort and
- Couchbase.

Characteristics of Document Data Store are high performance and flexibility.

Scalability varies, depends on stored contents.

Complexity is low compared to tabular, object and graph data stores.

Following are the features in Document Store:

1. Document stores unstructured data.
2. Storage has similarity with object store.
3. Data stores in nested hierarchies. Hierarchical information stores in a single unit called document tree. Logical data stores together in a unit.
4. Querying is easy. For example, using section number, sub-section number and figure caption and table headings to retrieve document partitions.
5. No object relational mapping enables easy search by following paths from the root of document tree.
6. Transactions on the document store exhibit ACID properties.

Typical uses of a document store are:

- (i) office document
- (ii) inventory store,
- (iii) forms data,
- (iv) document exchange and

(v)document search.

The demerits in Document Store are incompatibility with SQL and complexity for implementation.

The database stores and retrieves documents, such as XML, JSON, BSON (Binary-encoded Script Object Notation (for objects)). The documents are self-describing, hierarchical tree-structured consisting of maps, collections and scalar values. The documents stored are similar to each other but do not have to be the same. Some of the popular document data stores are CouchDB, MongoDB, Terrastore, OrientDB and RavenDB.

CouchDB uses the JSON store data, HTTP AP is for connectivity, JavaScript for the query language and MapReduce for processing.

Document JSON Format CouchDB Database

Apache CouchDB is an open-source database.

Its features are:

1. CouchDB provides mapping functions during querying, combining and filtering of information.
2. CouchDB deploys JSON Data Store model for documents. Each document maintains separate data and metadata (schema).
3. CouchDB is a multi-master application. Write does not require field locking when controlling the concurrency during multi-master application.
4. CouchDB querying language is JavaScript. JavaScript is a language which documents use to transform.
5. CouchDB queries the indices using a web browser. CouchDB accesses the documents using HTTP APL HTTP methods are Get, Put and Delete
6. CouchDB data replication is the distribution model that results in fault tolerance and reliability.

Document JSON Format-MongoDB Database

MongoDB Document database provides a rich query language and constructs, such as database indexes allowing easier handling of Big Data.

```
{
  "id": "1001"
  "Student Name":
  {
    "First": "Ashish",
    "Middle": "Kumar",
    "Last": "Rai"
  }
  "Category": "Student",
  "Class": "B.Tech.",
  "Semester": "VII",
  "Branch": "Computer Engineering",
  "Mobile": "12345"
}
```

The document store allows querying the data based on the contents as well. For example, it is possible to search the document where student's first name is "Ashish", Document store can also provide the search value's exact location.

The search is by using the document path.

A type of key accesses the leaf values in the tree structure.

Since the document stores are schema-less, adding fields to documents (XML or JSON) becomes a simple task.

Document Architecture Pattern and Discovering Hierarchical Structure

Following is example of an XML document in which a hierarchical structure discovers later. Figure 3.5 shows an XML document architecture pattern in a document fragment and document tree structure.

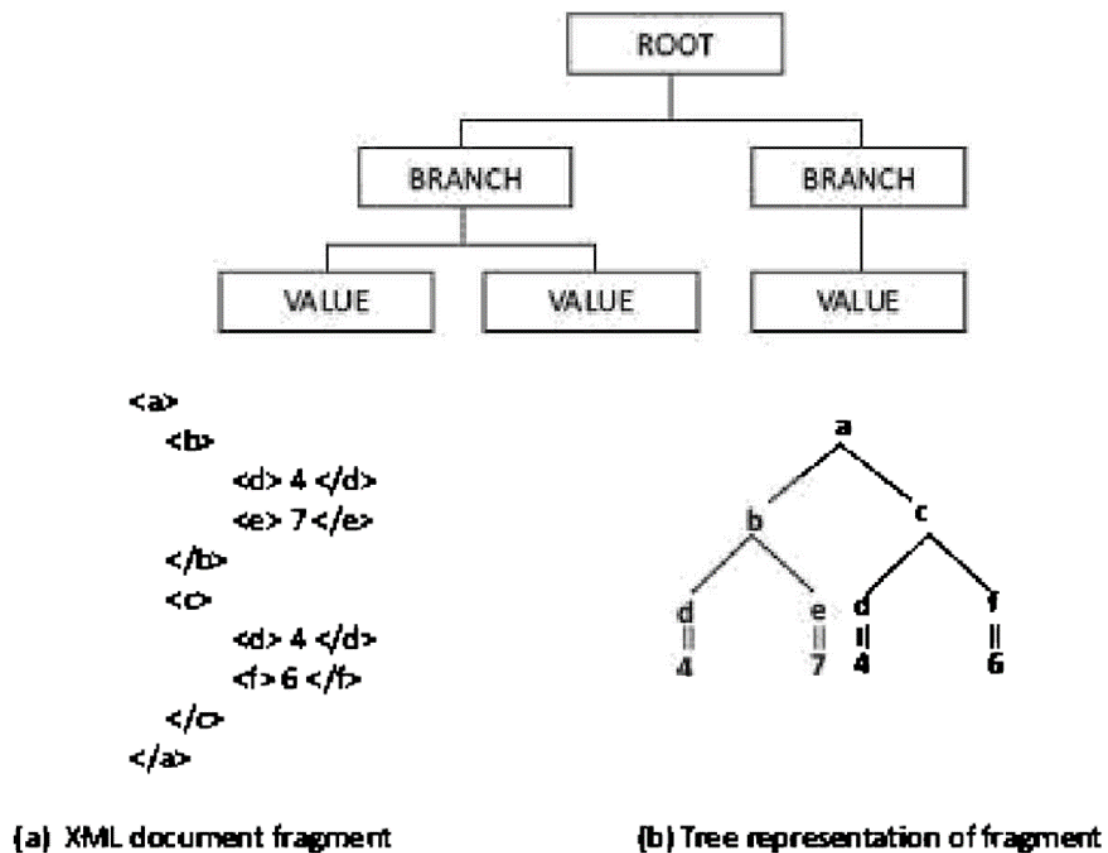


Figure 3.5 XML document architecture pattern

The document store follows a tree-like structure (similar to directory structure in file system). Beneath the root element there are multiple branches.

Each branch has a related path expression that provides a way to navigate from the root to any given branch, sub-branch or value.

The document store follows a tree-like structure (similar to directory structure in file system). Beneath the root element there are multiple branches.

Each branch has a related path expression that provides a way to navigate from the root to any given branch, sub-branch or value.

XQuery and XPath are query languages for finding and extracting elements and attributes from XML documents.

The query commands use sub-trees and attributes of documents.

The querying is similar as in SQL for databases.

XPath treats XML document as a tree of nodes.

XPath queries are expressed in the form of XPath expressions.

Give the structures of XML and JSON document fragments for a student record.

SOLUTION

Following are the structures:

<pre>{ "students": [{ "name": "Ashish Jain", "rollNo": "12345" }, { "name": "Sandeep Joshi", "rollNo": "12346" }] }</pre>	<pre><students> <student> <name>Ashish Jain</name> <rollNo>12345</rollNo> </student> <student> <name>Sandeep Joshi</name> <rollNo>12346</rollNo> </student> </students></pre>
---	---

(a) JSON

(b) XML equivalent

When compared with XML, JSON has the following advantages:

- XML is easier to understand but XML is more verbose than JSON.
- XML is used to describe structured data and does not include arrays, whereas JSON includes arrays.
- JSON has basically key-value pairs and is easier to parse from JavaScript.
- The concise syntax of JSON for defining lists of elements makes it preferable for serialization of text format objects.

Document Collection A collection can be used in many ways for managing a large document store. Three uses of a document collection are:

1. Group the documents together, similar to a directory structure in a file-system. (A directory consists of grouping of file folders.)
2. Enables navigating through document hierarchies, logically grouping similar documents and storing business rules such as permissions, indexes and triggers (special procedure on some actions in a database).
3. A collection can contain other collections as well.

Tabular data stores use rows and columns. Row-head field may be used as a key which access and retrieves multiple values from the successive columns in that row. The OLTP is fast on in-memory row-format data.

Oracle DBs provide both options: columnar and row format storages.

in-memory row-based data, in which a key in the first column of the row is at a memory address, and values in successive columns at successive memory addresses.

That makes OLTP easier.

All fields of a row are accessed at a time together during OLTP.

Different rows are stored in different addresses in the memory or disk.

In-memory row-based DB stores a row as a consecutive memory or disk entry. This strategy makes data searching and accessing faster during transactions processing.

In-memory column-based data has the keys (row-head keys) in the first column of each row at successive memory addresses. The next column of each row after the key has the values at successive memory addresses. The values in the third column of each row are at the next memory addresses in succession, and so on up to N columns. The N can be a very large number.

The column-based data makes the OLAP easier. All fields of a column access together. All fields of a set of columns may also be accessed together during OLAP. Different rows are stored in different addresses in the memory or disk, but each row values are now not at successive addresses. In-memory column-based DB store a column as a consecutive memory or disk entry. This strategy makes the analytics processing fast.

Columnar Data Store is A way to implement a schema. Storage of each column, successive values is at the successive memory addresses. Analytics processing (AP) In-memory uses columnar storage in memory. A pair of row-head and column-head is a key-pair. The pair accesses a field in the table.

All values in successive fields in a column consisting of multiple rows save at consecutive memory addresses. This enables fast accesses during in-memory analytics, which includes CPU accesses and analyses using memory addresses in which values are cached from the disk before processing. The OLAP (on-line AP) is also fast on in-memory column-format data. An application uses a combination of row head and a column head as a key for access to the value saved at the field.

Column-Family Data Store

Column-family databases store data in column families as rows that have many columns associated with a row key. Column families are groups of related data that is often accessed together.

Column-family data-store has a group of columns as a column family. A combination of row-head, column-family head and table-column head can also be a key to access a field in a column of the table during querying. A column-family head is also called a super-column head.

Column-Family Data Store

Column-family databases store data in column families as rows that have many columns associated with a row key. Column families are groups of related data that is often accessed together.

Column-family data-store has a group of columns as a column family. A combination of row-head, column-family head and table-column head can also be a key to access a field in a column of the table during querying. A column-family head is also called a super-column head.

Table 3.3 Each day's sales of chocolates on 999 ACVMs

	ACVM_ID	Nestle Chocolate Flavours Group				
		Popular Flavours Family		Costly Flavours Family		
		KitKat	Milk	Fruit and Nuts	Nougat	Oreo
Row-group_1 for IDs 1 to 100	1	360	150	500	101	222
	2	289	175	457	145	317

Row-group_m for IDs 901 to 999
	998	123	201	385	199	310
	999	75	215	560	108	250

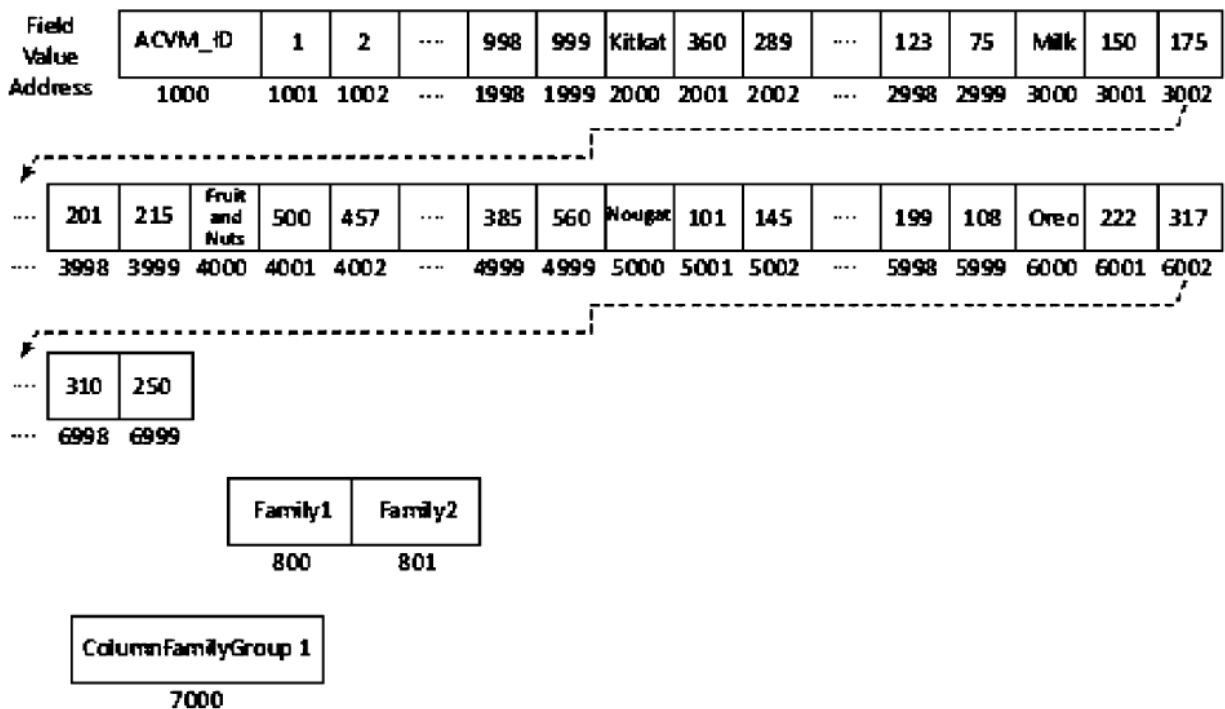


Figure 3.6 Fields in columnar storage and addresses in memory.

Columns store databases use a concept called a keyspace. A keyspace is kind of like a schema in the relational model. The keyspace contains all the column families (kind of like tables in the relational model), which contain rows, which contain columns.

A breakdown of each element in the row:

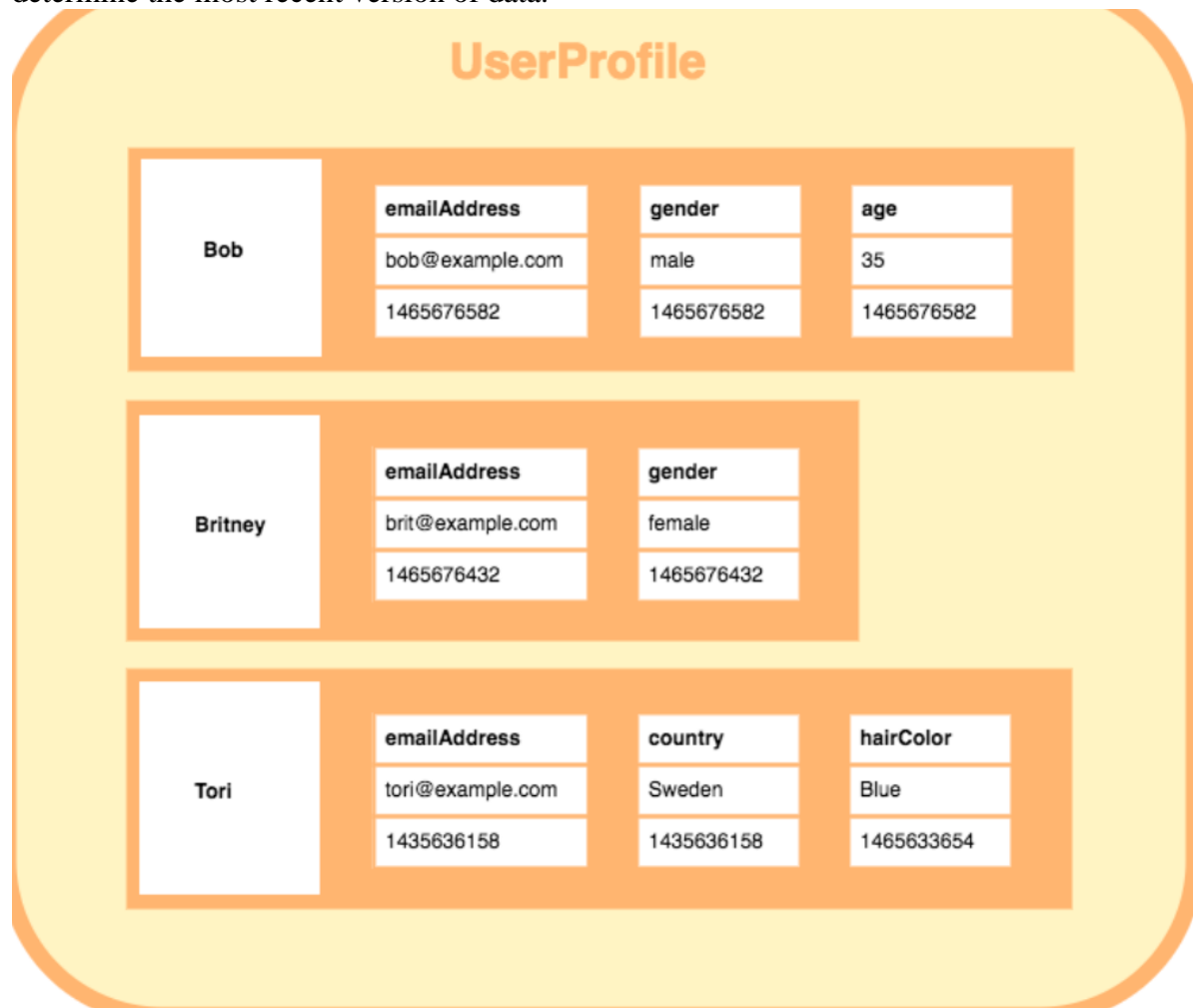
Row Key. Each row has a unique key, which is a unique identifier for that row.

Column. Each column contains a name, a value, and timestamp.

Name. This is the name of the name/value pair.

Value. This is the value of the name/value pair.

Timestamp. This provides the date and time that the data was inserted. This can be used to determine the most recent version of data.



Advantages of column stores are:

1. Scalability: The retrieval of data from the distributed node can be least complicated by an intelligent plan of row IDs and columns, thereby increasing performance. Scalability means addition of number of rows as the number of ACVMs increase (i). Number of processing instructions is proportional to the number of ACVMs due to scalable operations.
2. Partitionability: Large data of ACVMs can be partitioned into datasets of size, say 1MB in the number of row-groups. Values in columns of each row-group, process in-memory at a partition. Values in columns of each row-group independently parallelly process in-memory at the partitioned nodes.
3. Availability: The cost of replication is less. The lack of Join operations enables storing a part of a column-family matrix on remote computers. Thus, the data is always available in case of failure of any node.
4. Tree-like columnar structure consisting of column-family groups, column families and columns. The columns group into families. The column families group into column groups (super columns). A key for the column fields consists of three secondary keys: column-families group

ID, column-family ID and column-head name.

5. Adding new data at ease: Permits new column Insert operations. Trigger operation creates new columns on an Insert. The column-field values can add after the last address in memory if the column structure is known in advance. New row-head field, row-group ID field, column-family group, column family and column names can be created at any time to add new data.

6. Querying all the field values in a column in a family, all columns in the family or a group of column-families, is fast in in-memory column-family data store.

7. Replication of columns: HDFS-compatible column-family data stores replicate each data store with default replication factor= 3.

8. No optimization for Join: Column-family data stores are similar to sparse matrix data. The data do not optimize for Join operations.

Typical uses of column store are:

- (i) web crawling,
- (ii) large sparsely populated tables and
- (iii) system that has high variance.

Examples of widely used column-family data store are Google's BigTable, HBase and Cassandra.

Keys for row key, column key, timestamp and attribute uniquely identify the values in the fields

BigTable features :

1. Massively scalable NoSQL. BigTable scales up to 100s of petabytes.
2. Integrates easily with Hadoop and Hadoop compatible systems.
3. Compatibility with MapReduce, HBase APIs which are open-source Big Data platforms.
4. Key for a field uses not only row_ID and Column_ID but also timestamp and attributes. Values are ordered bytes. Therefore, multiple versions of values may be present in the BigTable.
5. Handles million of operations per second.
6. Handle large workloads with low latency and high throughput
7. Consistent low latency and high throughput
8. APIs include security and permissions
9. BigTable, being Google's cloud service, has global availability and its service is seamless.

EXAMPLE 3.7

Consider Example 3.6. Consider column fields which have keys to access a field not only by row ID and Column ID but also include the timestamp and attributes in a row. Show the column-keys for accessing column fields of a column.

SOLUTION

Table 3.4 gives keys for each day's sales of KitKat chocolates at ACVMs. First row-headings are the column-keys.

Table 3.4 Each day's sales of KitKat chocolates at ACVMs

Column-keys →	ACVM_ID	KitKatSalesDate	Timestamp	KitKatSalesNumber

Hive

Hive uses Record Columnar (RC) file-format records for querying. RC is the best choice for intermediate tables for fast column-family store in HDFS with Hive. Serializability of RC table column data is the advantage. RC file is DeSerializable into column data. A table such as that shown in Example 3.6 can be partitioned into row groups. Values at each column of a row group store as the RC record. The RC file records store data of a column in the row group (Serializability means query or transaction executable by series of instructions such that execution ensures correct results).

Explains the use of row groups in the RC file format for column of a row group:

Consider Example 3.6. Practically, row groups have millions of rows and in-memory between 10 MB and 1 GB. Assume two row groups of just two rows each. Consider the following values given in Table 3.3.

Row-group_1 for IDs 1 to 2					
1	360	150	500	101	222
2	289	175	457	145	317

Row-group_m for IDs 998 to 999					
998	123	201	385	199	310
999	75	215	560	108	250

Make a file in RC format.

SOLUTION

The values in each column are the records in file for each row group. Each row-group data is like a column of records which stores in the RC file.

Row group_1		Row group_m	
1, 2;		998, 999;	ACVM_ID
360, 289;		123, 75;	KitKat
...		...	Milk
...		...	Fruit and Nuts
		...	Nougat
222, 317;		310, 250;	Oreo

RC file for row group_1 will consists of records 1, 2; 360, 289; ..., 222, 317; on serialization of column records. RC file for row group_m will consists of

An ORC (Optimized Row Columnar) file consists of row-group data called stripes. ORC enables concurrent reads of the same file using separate RecordReaders. Metadata store uses Protocol Buffers for addition and removal of fields.

ORC is an intelligent Big Data file format for HDFS and Hive.2 An ORC file stores a collections of rows as a row-group. Each row-group data store in columnar format. This enables parallel processing of multiple row-groups in an HDFS cluster.

An ORC file consists of a stripe, size of the file is by default 256 MB. Stripe consists of indexing (mapping) data in 8 columns, row-group columns data (contents) and stripe footer (metadata). An ORC has two sets of columns data instead of one column data in RC. One column is for each map or list size and other values which enable a query to decide skipping or reading of the mapped columns. A mapped column has contents required by the query. The columnar layout in each ORC file thus, optimizes for compression and enables skipping of data in columns. This reduces read and decompression load.

Lightweight indexing is an ORC feature. Those blocks of rows which do not match a query skip as they do not map on using indices data at metadata. Each index includes the aggregated values of minimum, maximum, sum and count using aggregation functions on the content columns. Therefore, contents-column key for accessing the contents from a column consists of combination of row-group key, column mapping key, min, max, count (number) of column fields of the contents column. Table 3.5 gives the keys used to access or skip a contents column during querying. The keys are Stripe_ID, Index-column key, and contents-column name, min, max and count.

Stripe_ID	Index Column 1				Index Column 2
	Index column 1 key 1				Index column 2 key 1
	Contents-Column name	Contents Minimum value	Contents Maximum value	Count (number) of content-column fields	
	
	
	Index column 1 key 2				Index column 2 key 2
	Column-name	Minimum value	Maximum value	Count of number of column fields	
	
	
	

In the Example 3.6. ORC key to access during a query consist of not only column head 'KitKat' (Table 3.3) but also column minimum and maximum sales on an ACVM, count of number of fields in values 'KitKat'. Analytics operations frequently need these values. Ready availability of these values from the index data itself improves the throughput in Big Data HDFS environment. These values do not need to compute again and again using aggregation functions, such as min, max and count.

An ORC thus, optimizes for reading serially the column fields in HDFS environment. The throughput increases due to skipping and reading of the required fields at contents-column key. Reading less number of ORC file content-columns reduces the workload on the NameNode.

Parquet is nested hierarchical columnar-storage concept. Nesting sequence is the table, row group, column chunk and chunk page. Apache Parquet file is columnar-family store file. Apache Spark SQL executes user defined functions (UDFs) which query the Parquet file columns. A programmer writes the codes for an UDF and creates the processing function for big long queries. A Parquet file uses an HDFS block. The block stores the file for processing queries on Big Data. The file compulsorily consists of metadata, though the file need not consist of data.

Tabular Data: Parquet File Formats

The Parquet file consists of row groups. A row-group columns data process in-memory after data cache and buffer at the memory from the disk. Each row group has a number of columns. A row group has Ncol columns, and row group consists of Ncol column chunks. This means each column chunk consists of values saved in each column of each row group.

A column chunk can be divided into pages and thus, consists of one or more pages. The column chunk consists of a number of interleaved pages, Npg. A page is a conceptualized unit which can be compressed or encoded together at an instance. The unit is minimum portion of a chunk which is read at an instance for in-memory analytics.

Parquet format file does not consist of extra column per nesting level, just one column per leaf in the schema.

Row-group_ID	Column Chunk 1 key			
	Page 1 key	Page 2 key	...	Page m key

	Column Chunk 2 key			
	Page 1key	Page 2 key	...	Page key m'

Object Data Store

An object store refers to a repository which stores the:

1. Objects (such as files, images, documents, folders, and business reports)
2. System metadata which provides information such as filename, creation_date, last_modified, language_used (such as Java, C, C#, C++, Smalltalk, Python), access permissions, supported query languages)
3. Custom metadata which provides information, such as subject, category, sharing permissions.

Metadata enables the gathering of metrics of objects, searches, find the contents and specifies the objects in an object data-store tree. Metadata finds the relationships among the objects, maps the object relations and trends. Object Store metadata interfaces with the Big Data. API first mines the metadata to enable mining of the trends and analytics. The metadata defines classes and properties of the objects. Each Object Store may consist of a database. Document content can be stored in either the object store database storage area or in a file storage area. A single file domain may contain multiple Object Stores.

Eleven Functions Supporting APis An Object data store consists of functions supporting APis for:

- (i) scalability,
- (ii) indexing,
- (iii) large collections,
- (iv) querying language, processing and optimization(s),
- (v) Transactions,
- (vi) data replication for high availability, data distribution model, data integration (such as with relational database, XML, custom code),
- (vii) schema evolution,
- (viii) persistency,
- (ix) persistent object life cycle,

(x) adding modules and

(xi) locking and caching strategy.

Object Store may support versioning for collaboration. Object Store can be created using IBM 'Content Platform Engine'. Creation needs installing and configuring the engine (Engine is software which drives forward.). Console of the engine makes creation of process easy. Amazon S3 and Microsoft Azure BLOB support the Object Store.

Amazon S3 (Simple Storage Service) S3 refers to Amazon web service on the cloud named S3. The S3 provides the Object Store. The Object Store differs from the block and file-based cloud storage. Objects along with their metadata store for each object store as the files. S3 assigns an ID number for each stored object. The service has two storage classes: Standard and infrequent access. Interfaces for S3 service are REST, SOAP and Bit Torrent. S3 uses include web hosting, image hosting and storage for backup systems. S3 is scalable storage infrastructure, same as used in Amazon e-commerce service. S3 may store trillions of objects.

List the functions of Minio, Riak, VERSANT Object Database (VOD), GEMSTONE, Amazon S3 and Microsoft Azure BLOB that support using Object Store APIs.

SOLUTION

1. An open-source multi-clouds object storage server is Minio, which is API compatible with Amazon S3 API and number of widely used public and private clouds. Compatibility enables data export to S3 and usages of APIs.
2. Riak CS (Cloud Storage) is object storage management software on top of Riak. It models on open-source distributed-database which is Amazon-compliant. This means database exports to S3 and use S3 APIs.
3. VOD consists of 11 functions supporting APIs listed above. VOD enables use by multiple concurrent users. VOD supports cross-platform operating systems (OSs), such as Linux, Windows NT, AIX, HP-UX and Solaris (both 32 and 64 bits for all platforms).
4. GEMSTONE Object DB APIs development language is SmallTalk. The platform supports in-memory DBs, object-oriented processing and distributed caches. GEMSTONE provides cross platform support, OSs AIX, Linux, MacOS and Solaris.

The following example explains object relational mapping.

How does an HTML object and XML based web service relate with tabular data stores?

SOLUTION

Figure 3.7 shows the object relational mapping of HTML document and XML web services store with a tabular data store.

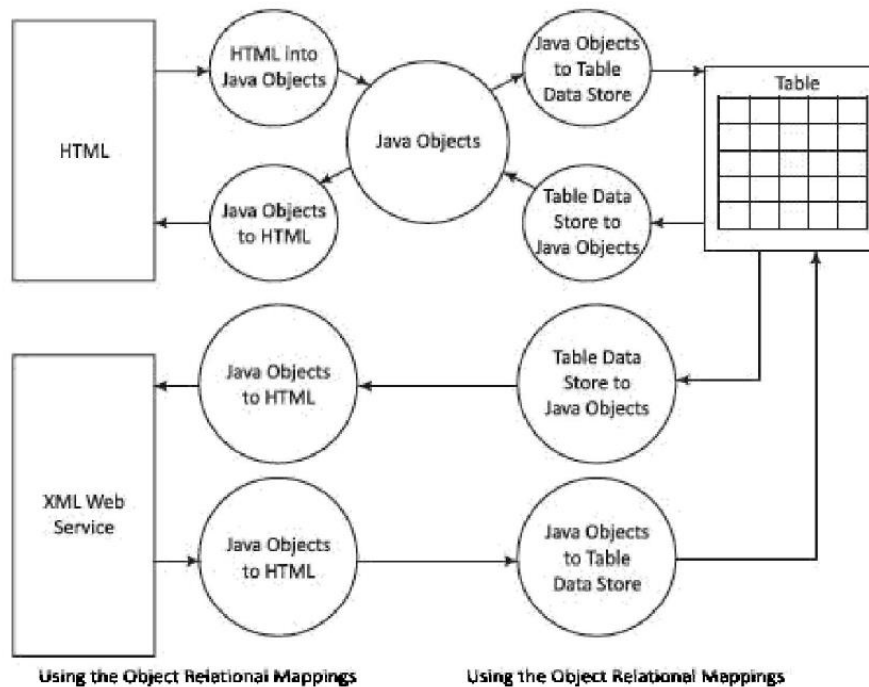


Figure 3.7 HTML document and XML web services

Graph Database

Data Store focuses on modeling interconnected structure of data. Data stores based on graph theory relation $G = (E, V)$, where E is set of edges e_1, e_2, \dots and V is set of vertices, v_1, v_2, \dots, v_n . Nodes represent entities or objects. Edges encode relationships between nodes. Some operations become simpler to perform using graph models. Examples of graph model usages are social networks of connected people. The connections to related persons become easier to model when using the graph model.

The following example explains the graph database application in describing entities relationships and relationship types

EXAMPLE 3.11

Let us assume a car company represents a node entity, which has two connected nodes comprising two model entities, namely Hexa and Zest. Draw graph with directed lines, joining the car company with two entities. (i) How do four directed lines relate to four weeks and two directed lines? One directed line corresponds to a car model. Only directed line corresponds to weekly total sales. (ii) How will the yearly sales compute? (iii) Show the path traversals for computations exhibit BASE properties.

SOLUTION

- (i) Figure 3.8 shows section of a graph database for the sales of two car models.



Figure 3.8 Section of the graph database for car-model sales

Figure 3.8 Section of the graph database for car-model sales

- (ii) The yearly sales compute by path traversals from nodes for weekly sales to yearly sales data.
- (iv) The path traversals exhibit BASE properties because during the intermediate paths, consistency is not maintained. Eventually when all the path traversals complete, the data becomes consistent.

Graph databases enable fast network searches. Graph uses linked datasets, such as social media data. Data store uses graphs with nodes and edges connecting each other through relations, associations and properties.

Querying for data uses graph traversal along the paths. Traversal may use single-step, path expressions or full recursion. A relationship represents key. A node possesses property including ID. An edge may have a label which may specify a role.

Characteristics of graph databases are:

1. Use specialized query languages, such as RDF uses SPARQL
2. Create a database system which models the data in a completely different way than the key-values, document, columnar and object data store models.
3. Can have hyper-edges. A hyper-edge is a set of vertices of a hypergraph. A hypergraph is a generalization of a graph in which an edge can join any number of vertices (not only the neighbouring vertices).
4. Consists of a collection of small data size records, which have complex interactions between graph-nodes and hypergraph nodes. Nodes represent the entities or objects. Nodes use Joins. Node identification can use URI or other tree-based structure. The edge encodes a relationship between the nodes.

When a new relationship adds in RDBMS, then the schema changes. The task of adding relations in graph database is simpler. The nodes assign internal identifiers to the nodes and use these identifiers to join the network. Traversing the joins or relationships is fast in graph databases. It is due to the simpler form of graph nodes. The graph data may be kept in RAM only. The relationship between nodes is consistent in a graph store.

Graph databases have poor scalability. They are difficult to scale out on multiple servers. This is due to the close connectivity feature of each node in the graph. Data can be replicated on multiple servers to enhance read and the query processing performance. Write operations to multiple servers and graph queries that span multiple nodes, can be complex to implement.

Typical uses of graph databases are:

- (i) link analysis,
- (ii) friend of friend queries,
- (iii) Rules and inference,
- (iv) rule induction and
- (v) Pattern matching. Link analysis is needed to perform searches and look for patterns

and relationships in situations, such as social networking, telephone, or email. Rules and inference are used to run queries on complex structures such as class libraries, taxonomies and rule-based systems.

Examples of graph DBs are Neo4J, AllegroGraph, HyperGraph, Infinite Graph, Titan and FlockDB. Neo4J graph database enable easy usages by Java developers. Neo4J can be designed fully ACID rules compliant. Design consists of adding additional path traversal in between the transactions such that data consistency is maintained and the transactions exhibit ACID properties.

Spark provides a simple and expressive programming model that includes supports to a wide range of applications, including graph computation.

Six data architectures are SQL-table, key-value pairs, in-memory column-family, document, graph and object. Selected architecture may need variations due to business requirements. Business requirements are ease of using an architecture and long-term competitive advantage. The following example explains the requirements for the database of students of a University that offers multiple courses in their various academic programmes for several years

EXAMPLE 3.12

List the selection requirements for the database of University students in successive years. The University runs various Under Graduate and Post Graduate programmes. Students are registered to Multiple courses in a programme.

SOLUTION

Following are the selection requirements:

1. **Scalability:** Since the University archives the data for several years, data store should be scalable.
2. **Search ability:** Search of required information needs to be fast.
3. **Quarrying ability:** All applications need to query the data. Query retrieves the required data among the Big Data of several years.
4. **Security:** Database needs security and fault tolerance.

5. **Affordability:** Open source is a requirement.
6. **Interoperability:** Needs ease in search from different platforms. Search from any computer operating system, such as Windows, Mac, Linux, Android and iOS should be feasible.
7. **Importability:** Database needs to import data from other platforms, such as import of slides, video lectures, tutorials, e-books, webinars should be facilitated in store.
8. **Transformability:** Queries may be written in one language and may require transformation to another language, such as HTML.

Analysis of the above requirements suggests the document architecture pattern will be more suitable.

Variations of NoSQL Architectural Patterns

Kelly-McCreary, co-founder of 'NoSQL Now' suggested that when selecting a NoSQL-pattern, the pattern may need change and require variation to another pattern(s).

Some reasons for this are:

1. Focus changing from performance to scalability
2. Changing from modifiability to agility
3. Greater emphasis on Big Data, affording capacity, availability of support, ability for searching and monitoring the actions

Steps for selecting a NoSQL data architectural pattern can be :

1. Select an architecture
2. Perform a use-case driven difficulty analysis for each of the six architectural patterns. Difficulties may be low, medium or high in the following processes:
 - (i) ingestion,
 - (ii) validation of structure and its fields,
 - (iii) updating process using batch or record by record approach, (iv) searching process using full text or by changing the sorting order, and
 - (v) export the reports or application results in HTML, XML or JSON.
3. Estimate the total efforts for each architecture for all business requirements. Process the choice of architecture using trade-off. For example, between the MongoDB document data store and Cassandra column-family data store.

Using NoSQL to Manage Big Data

This describe how to use a NoSQL data store to manage Big Data.

NoSQL

- (i) limits the support for Join queries, supports sparse matrix like columnar-family,

(ii) characteristics of easy creation and high processing speed, scalability and storability of much higher magnitude of data (terabytes and petabytes).

NoSQL sacrifices the support of ACID properties, and instead supports CAP and BASE properties.

NoSQL data processing scales horizontally as well vertically.

Big Data solution needs scalable storage of terabytes and petabytes, dropping of support for database Joins, and storing data differently on several distributed servers (data nodes) together as a cluster. A solution, such as CouchDB, DynamoDB, MongoDB or Cassandra follow CAP theorem (with compromising the consistency factor) to make transactions faster and easier to scale. A solution must also be partitioning tolerant.

Characteristics of Big Data NoSQL solution are:

1. High and easy scalability: NoSQL data stores are designed to expand horizontally. Horizontal scaling means that scaling out by adding more machines as data nodes (servers) into the pool of resources (processing, memory, network connections). The design scales out using multi-utility cloud services.
2. Support to replication: Multiple copies of data store across multiple nodes of a cluster. This ensures high availability, partition, reliability and fault tolerance.
3. Distributable: Big Data solutions permit sharding and distributing of shards on multiple clusters which enhances performance and throughput.
4. Usages of NoSQL servers which are less expensive. NoSQL data stores require less management efforts. It supports many features like automatic repair, easier data distribution and simpler data models that makes database administrator (DBA) and tuning requirements less stringent.
5. Usages of open-source tools: NoSQL data stores are cheap and open source. Database implementation is easy and typically uses cheap servers to manage the exploding data and transaction while RDBMS databases are expensive and use big servers and storage systems. So, cost per gigabyte data store and processing of that data can be many times less than the cost of RDBMS.
6. Support to schema-less data model: NoSQL data store is schema less, so data can be inserted in a NoSQL data store without any predefined schema. So, the format or data model can be changed any time, without disruption of application. Managing the changes is a difficult problem in SQL.
7. Support to integrated caching: NoSQL data store support the caching in system memory. That increases output performance. SQL database needs a separate infrastructure for that.
8. No inflexibility unlike the SQL/RDBMS, NoSQL DBs are flexible (not rigid) and have no structured way of storing and manipulating data. SQL stores in the form of tables consisting of rows and columns. NoSQL data stores have flexibility in following ACID rules.

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

The columns of two tables relate by a relationship. A relational algebraic equation specifies the relation. Keys share between two or more SQL tables in RDBMS. Shared nothing (SN) is a cluster architecture. A node does not share data with any other node.

Big Data store consists of SN architecture. Big Data store, therefore, easily partitions into shards. A partition processes the different queries on data of the different users at each node independently. Thus, data processes run in parallel at the nodes. A node maintains a copy of running-process data. A coordination protocol controls the processing at all SN nodes. An SN architecture optimizes massive parallel data processing.

Data of different data stores partition among the number of nodes (assigning different computers to deal with different users or queries). Processing may require every node to maintain its own copy of the application's data, using a coordination protocol. Examples are using the partitioning and processing are Hadoop, Flink and Spark.

The features of SN architecture are as follows:

1. Independence: Each node with no memory sharing; thus possesses computational self-sufficiency
2. Self-Healing: A link failure causes creation of another link
3. Each node functioning as a shard: Each node stores a shard (a partition of large DBs)
4. No network contention.

Big Data requires distribution on multiple data nodes at clusters. Distributed software components give advantage of parallel processing; thus providing horizontal scalability. Distribution gives

- (i) ability to handle large-sized data, and
- (ii) processing of many read and write operations simultaneously in an application. A resource manager manages, allocates, and schedules the resources of each processor, memory and network connection. Distribution increases the availability when a network slows or link fails. Four models for distribution of the data store are given below:

Four models for distribution of the data store are:

- Single Server Model
- Sharding Very Large Databases
- Master-Slave Distribution Model
- Peer-to-Peer Distribution Model
- Choosing Master-Slave versus Peer-to-Peer

Simplest distribution option for NoSQL data store and access is Single Server Distribution (SSD) of an application. A graph database processes the relationships between nodes at a server. The SSD model suits well for graph DBs.

Aggregates of datasets may be key-value, column-family or BigTable data stores which require sequential processing. These data stores also use the SSD model. An application executes the data sequentially on a single server. Figure 3.9(a) shows the SSD model. Process and datasets distribute to a single server which runs the application.

Figure 3.9(b) shows sharding of very large datasets into four divisions, each running the application on four i,j, k and l different servers at the cluster. DBi, DBj, DBk and DBl are four shards.

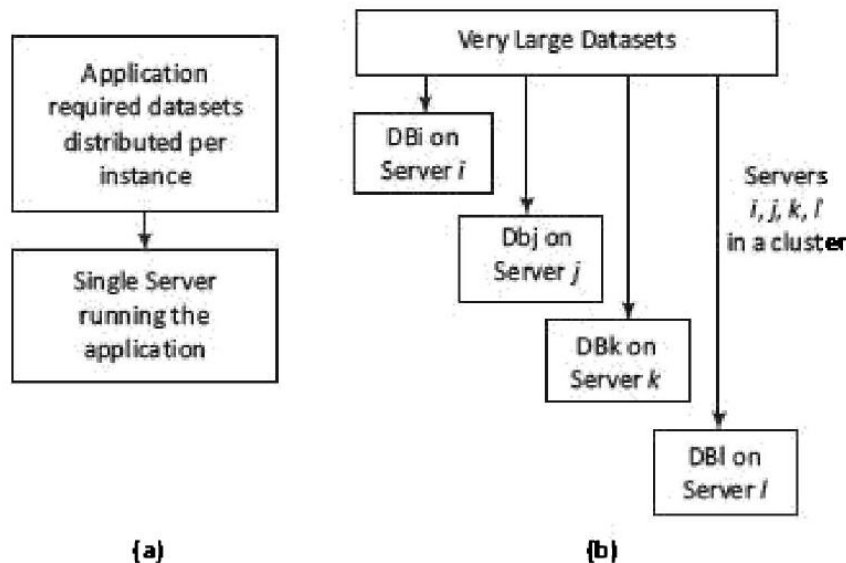


Figure 3.9 (a) Single server model (b) Shards distributed on four servers in a cluster.

Choosing the Distribution Models: Master-Slave Distribution Model

A node serves as a master or primary node and the other nodes are slave nodes. Master directs the slaves. Slave nodes data replicate on multiple slave servers in

Master Slave Distribution (MSD) model. When a process updates the master, it updates the slaves also.

A process uses the slaves for read operations. Processing performance improves when process runs large datasets distributed onto the slave nodes. Figure 3.10 shows an example of MongoDB. MongoDB database server is mongod and the client is mongo.

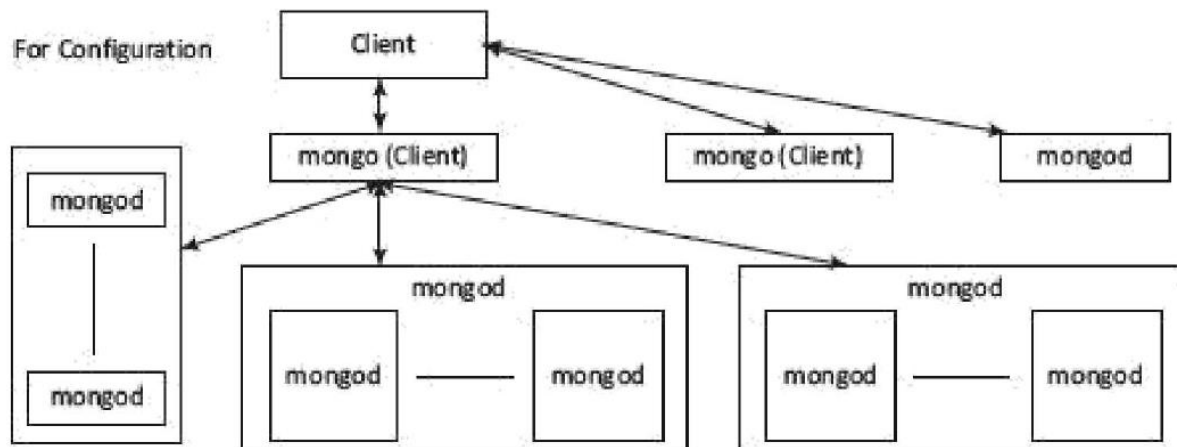


Figure 3.10 Master-slave distribution model. Mongo is a client and mongod is the server

Choosing the Distribution Models: Master-Slave Distribution Model

Master-Slave Replication Processing performance decreases due to replication in MSD distribution model. Resilience for read operations is high, which means if in case data is not available from a slave node, then it becomes available from the replicated nodes. Master uses the distinct write and read paths.

Complexity Cluster-based processing has greater complexity than the other architectures. Consistency can also be affected in case of problem of significant time taken for updating.

Peer-to-Peer distribution (PPD) model and replication show the following characteristics: (1) All replication nodes accept read request and send the responses. (2) All replicas function equally. (3) Node failures do not cause loss of write capability, as other replicated node responds.

Cassandra adopts the PPD model. The data distributes among all the nodes in a cluster. Performance can further be enhanced by adding the nodes. Since nodes read and write both, a replicated node also has updated data. Therefore, the biggest advantage in the model is consistency. When a write is on different nodes, then write in consistency occurs.

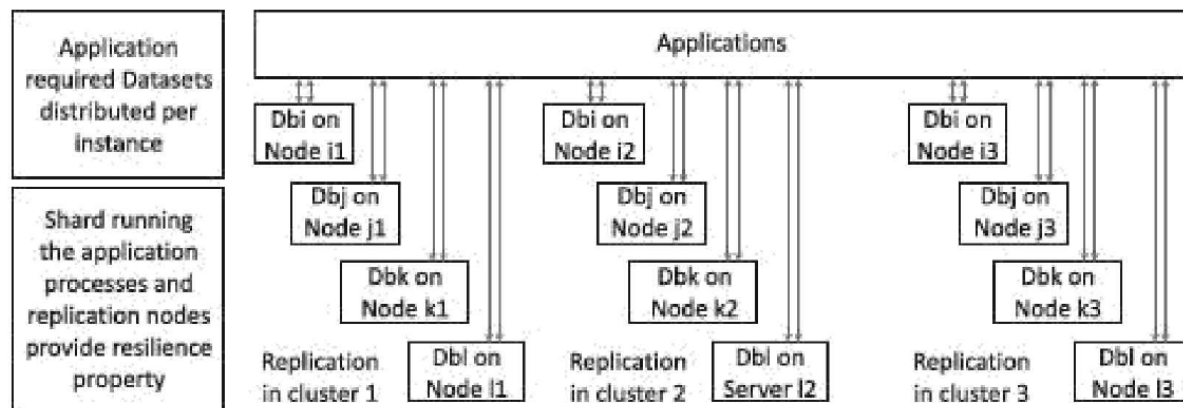


Figure 3.11 Shards replicating on the nodes, which does read and write operations both

Master-slave replication provides greater scalability for read operations. Replication provides resilience during the read. Master does not provide resilience for writes. Peer-to-peer replication provides resilience for read and writes both.

Sharing Combining with Replication Master-slave and sharding creates multiple masters. However, for each data a single master exists. Configuration assigns a master to a group of datasets. Peer-to-peer and sharding use same strategy for the column-family data stores. The shards replicate on the nodes, which does read and write operations both.

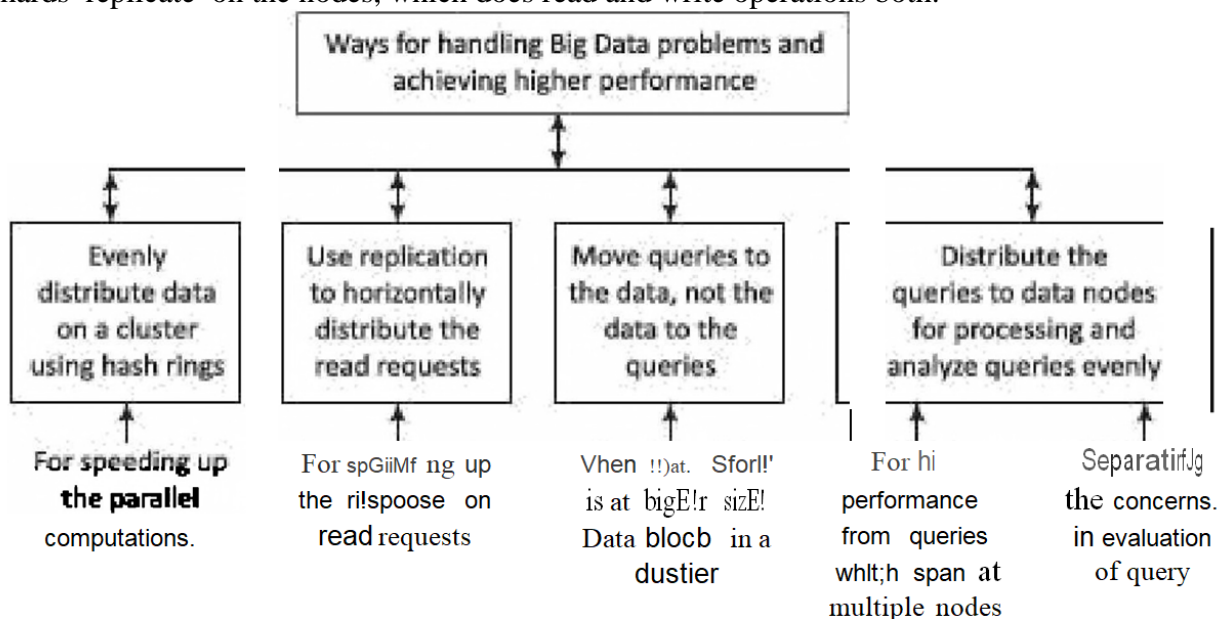


Figure 3.12 Four ways for handling big data problems

Ways of Handling Big Data Problems

1. Evenly distribute the data on a cluster using the hash rings: Consistent hashing refers to a process where the datasets in a collection distribute using a hashing algorithm which generates the pointer for a collection. Using only the hash of Collection_ID, a Big Data

solution client node determines the data location in the cluster. Hash Ring refers to a map of hashes with locations. The client, resource manager or scripts use the hash ring for data searches and Big Data solutions. The ring enables the consistent assignment and usages of the dataset to a specific processor.

2. Use replication to horizontally distribute the client read-requests: Replication means creating backup copies of data in real time. Many Big Data clusters use replication to make the failure-proof retrieval of data in a distributed environment. Using replication enables horizontal scaling out of the client requests.

3. Moving queries to the data, not the data to the queries: Most NoSQL data stores use cloud utility services (Large graph databases may use enterprise servers). Moving client node queries to the data is efficient as well as a requirement in Big Data solutions.

4. Queries distribution to multiple nodes: Client queries for the DBs analyze at the analyzers, which evenly distribute the queries to data nodes/ replica nodes. High performance query processing requires usages of multiple nodes. The query execution takes place separately from the query evaluation (The evaluation means interpreting the query and generating a plan for its execution sequence).

MongoDB is an open source DBMS.

MongoDB programs create and manage databases.

MongoDB manages the collection and document data store.

MongoDB functions do querying and accessing the required information.

The functions include viewing, querying, changing, visualizing and running the transactions.

Changing includes updating, inserting, appending or deleting.

MongoDB is

- (i) non-relational,
- (ii) NoSQL
- (iii) distributed,
- (iv) open source,
- (v) document based,
- (vi) cross-platform,
- (vii) Scalable,
- (viii) flexible data model,
- (ix) Indexed,
- (x) multi-master and
- (xi) fault tolerant.

Document data store in JSON-like documents. The data store uses the dynamic schemas.

The typical MongoDB applications are content management and delivery systems, mobile applications, user data management, gaming, e-commerce, analytics, archiving and logging.

Features of MongoDB:

1. MongoDB data store is a physical container for collections. Each DB gets its own set of files on the file system. A number of DBs can run on a single MongoDB server. DB is default DB in MongoDB that stores within a data folder. The database server of MongoDB is mongod and the client is mongo.

2. Collection stores a number of MongoDB documents. It is analogous to a table of

RDBMS. A collection exists within a single DB to achieve a single purpose. Collections may store documents that do not have the same fields. Thus, documents of the collection are schema-less. Thus, it is possible to store documents of varying structures in a collection.

Practically, in an RDBMS, it is required to define a column and its data type, but does not need them while working with the MongoDB.

3. Document model is well defined. Structure of document is clear, Document is the unit of storing data in a MongoDB database. Documents are analogous to the records of RDBMS table. Insert, update and delete operations can be performed on a collection. Document use JSON (javascript Object Notation) approach for storing data. JSON is a lightweight, self-describing format used to interchange data between various applications. JSON data basically has key-value pairs. Documents have dynamic schema.

4. MongoDB is a document data store in which one collection holds different documents. Data store in the form of JSON-style documents. Number of fields, content and size of the document can differ from one document to another.

5. Storing of data is flexible, and data store consists of JSON-like documents.

This implies that the fields can vary from document to document and data structure can be changed over time; JSON has a standard structure, and a scalable way of describing hierarchical data.

6. Storing of documents on disk is in BSON serialization format. BSON is a binary representation of JSON documents. The mongo JavaScript shell and MongoDB language drivers perform translation between BSON and language-specific document representation.

7. Querying, indexing, and real time aggregation allows accessing and analyzing the data efficiently.

8. Deep query-ability-Supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.

9. No complex Joins.

10. Distributed DB makes availability high, and provides horizontal scalability.

11. Indexes on any field in a collection of documents: Users can create indexes on any field in a document. Indices support queries and operations. By default, MongoDB creates an index on the _id field of every collection.

12. Atomic operations on a single document can be performed even though support of multi-document transactions is not present. The operations are alternate to ACID transaction requirement of a relational DB.

13. Fast-in-place updates: The DB does not have to allocate new memory location and write a full new copy of the object in case of data updates. This results into high performance for frequent update use cases. For example, incrementing a counter operation does not fetch the document from the server. Here, the increment operation can simply be set.

14. No configurable cache: MongoDB uses all free memory on the system automatically by way of memory-mapped files (The operating systems use the similar approach with their file system caches). The most recently used data is kept in RAM. If indexes are created for queries and the working dataset fits in RAM, MongoDB serves all queries from memory.

15. Conversion/mapping of application objects to data store objects not needed

Dynamic Schema

Dynamic schema implies that documents in the same collection do not need to have the same set of fields or structure. Also, the similar fields in a document may contain different types of data. Table 3.8 gives the comparison with RDBMS.

Any relational DB has a typical schema design that shows the number of tables and the relationship between these tables. While in MongoDB, there is no concept of relationship.

Table 3.8 Comparison of RDBMS and MongoDB databases

RDBMS	MongoDB
Database	Data store
Table	Collection
Column	Key
Value	Value
Records / Rows / Tuple	Document / Object
Joins	Embedded Documents
Index	Index
Primary key	Primary key (_id) is default key provided by MongoDB itself

Replication ensures high availability in Big Data. Presence of multiple copies increases on different database servers. This makes DBs fault-tolerant against any database server failure. Multiple copies of data certainly help in localizing the data and ensure availability of data in a distributed system environment.

MongoDB replicates with the help of a replica set.

A replica set in MongoDB is a group of mongod (MongoDB server) processes that store the same dataset.

Replica sets provide redundancy but high availability.

A replica set usually has minimum three nodes.

Any one out of them is called primary.

The primary node receives all the write operations.

All the other nodes are termed as secondary.

The data replicates from primary to secondary nodes.

A new primary node can be chosen among the secondary nodes at the time of automatic failover or maintenance.

The failed node when recovered can join the replica set as secondary node again.

Replica set starts a mongod instance by specifying replSet option before running these commands from mongo (MongoDb Client). Table 3.9 gives the commands used for replication (Recoverability means even on occurrences of failures; the transactions ensure consistency).

Commands	Description
rs.initiate ()	To initiate a new replica set
rs.conf ()	To check the replica set configuration
rs.status ()	To check the status of a replica set
rs.add ()	To add members to a replica set

Auto-sharding Sharding is a method for distributing data across multiple machines in a distributed application environment. MongoDB uses sharding to provide services to Big Data applications. Vertical scaling by increasing the resources of a single machine is quite expensive. Thus, horizontal scaling of the data can be achieved using sharding mechanism where more database servers can be added to support data growth and the demands of more read and write operations. Sharding automatically balances the data and load across various servers. Sharding provides additional write capability by distributing the write load over a number of mongod (MongoDBServer) instances.

(Figure 3.10) Basically, it splits the dataset and distributes them across multiple DBs, called shards on the different servers. Each shard is an independent DB. The whole collection of shards forms a single logical DB. If a DB has a 1 terabyte dataset distributed amongst 20 shards, then each shard contains only 50 Giga Byte of data.

A shard stores lesser data than the actual data and handles lesser number of operations in a single instance.

For example, to insert data into a collection, the application needs to access only the shard that contains the specified collection. A cluster can thus easily increase its capacity horizontally.

Type	Description
Double	Represents a float value.
String	UTF-8 format string.
Object	Represents an embedded document.
Array	Sets or lists of values.
Binary data	String of arbitrary bytes to store images, binaries.
Object id	ObjectIds (MongoDB document identifier, equivalent to a primary key) are: small, likely unique, fast to generate, and ordered. The value consists of 12-bytes, where the first four bytes are for timestamp that reflects the instance when ObjectId creates.
Boolean	Represents logical true or false value.
Date	BSON Date is a 64-bit integer that represents the number of milliseconds since the Unix epoch (Jan 1, 1970).
Null	Represents a null value. A value which is missing or unknown is Null.
Regular Expression	RegExp maps directly to a JavaScript RegExp
32-bit integer	Numbers without decimal points save and return as 32-bit integers.
Timestamp	A special timestamp type for internal MongoDB use and is not associated with the regular date type. Timestamp values are a 64-bit value, where first 32 bits are time, t (seconds since the Unix epoch), and next 32 bits are an incrementing ordinal for operations within a given second.

64-bit integer	Number without a decimal point save and return as 64-bit integer.
Min key	MinKey compare less than all other possible BSON element values, respectively, and exist primarily for internal use.
Max key	MaxKey compares greater than all other possible BSON element values, respectively, and exist primarily for internal use.

Rich Queries and Other DB Functionalities MongoDB offers a rich set of features and functionality compared to those offered in simple key-value stores. They can be comparable to those offered by any RDBMS. MongoDB has a complete query language, highly-functional secondary indexes (including text search and geospatial), and a powerful aggregation framework for data analysis. MongoDB provides functionalities and features for more diverse data types than a relational DB, and at scale.

Table 3.11 gives a comparison of features.

The ability to derive a document-based data model is also a distinct advantage of MongoDB. The

method of storing data in the form of BSON (Binary JSON) helps to store the data in a very rich way while can hold arrays and other documents.

Table 3.11 Comparison of features MongoDB with respect to RDBMS

Features	RDBMS	MongoDB
Rich Data Model	No	Yes
Dynamic Schema	No	Yes
Typed Data	Yes	Yes
Data Locality	No	Yes
Field Updates	Yes	Yes
Complex Transactions	Yes	No
Auditing	Yes	Yes
Horizontal Scaling	No	Yes

Table 3.12 MongoDB querying commands

Following explains the sample usages of the commands:

To Create database Command `use` - `use` command creates a database; For example, Command `use lego` creates a database named *lego*. (A sample database is created to demonstrate subsequent queries. The Lego is an international toy brand). Default database in MongoDB is *test*.

To see the existence of database Command `db` - `db` command shows that *lego* database is created.

To get list of all the databases Command `show dbs` - This command shows the names of all the databases.

To drop database Command `db.dropDatabase()` - This command drops a database. Run `use lego` command before the `db.dropDatabase()` command to drop *lego* Database. If no database is selected, the default database *test* will be dropped.

To create a collection Command `insert()` -To create a collection, the easiest way is to insert a record (a document consisting of keys (Field names) and Values) into a collection. A new collection will be created, if the collection does not exist. The following statements demonstrate the creation of a collection with three fields (ProductCategory, ProductId and ProductName) in the *lego*


```

db.lego.insert
(
    {
        "ProductCategory": "Airplane",
        "ProductId": 10725,
        "ProductName": "Lost Temple"
    }
)

```

To add array in a collection Command insert() - Insert command can also be used to insert multiple documents into a collection at one time.

```

db.lego.insert
(
    [
        {
            "ProductCategory": "Airplane",
            "ProductId": 10725,
            "ProductName": "Lost Temple"
        },
        {
            "ProductCategory": "Airplane",
            "ProductId": 31047,
            "ProductName": "Propeller Plane"
        },
        {
            "ProductCategory": "Airplane",
            "ProductId": 31049,
            "ProductName": "Twin Spin Helicopter"
        }
    ]
)

```

To view all documents in a collection Command `db.<database name>.find()` - Find command is equivalent to select query of RDBMS. Thus, "Select * from lego" can be written as `db.lego.find()` in MongoDB. MongoDB created unique objectId ("_id") on its own. This is the primary key of the collection. Command `db.<database name>.find().pretty()` gives a prettier look.

To update a document Command `db.<database name>.update()` - Update command is used to change the field value. By default, multi attribute is false. If {multi: true} is not written then it will update only the first document.

To delete a document Command `db.<database name>.remove()` - Remove command is used to delete the document. The query `db.<database name>.remove({"ProductID":10725})` removes the document whose productId is 10725.

Cassandra

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

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

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

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

Characteristics of Cassandra are

- (i) open source,
- (ii) scalable
- (iii) Non-relational
- (iv) NoSQL
- (v) Distributed
- (vi) column based,
- (vii) decentralized,
- (viii) fault tolerant and
- (ix) tunable consistency.

Features of Cassandra are as follows:

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

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

CASSANDRA DATABASES

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

Table 3.13 Components of cassandra

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

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

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

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

The replica placement strategies are:

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

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

CQL Type	Description
ascii	US-ASCII character string
bigint	64-bit signed long integer
blob	Arbitrary bytes (no validation), BLOB expressed in hexadecimal
boolean	True or false
counter	Distributed counter value (64-bit long)
decimal	Variable-precision decimal integer, float
double	64-bit IEEE-754 <i>double precision</i> floating point integer, float
float	32-bit IEEE-754 <i>single precision</i> floating point integer, float
inet	IP address string in IPv4 or IPv6 format, used by the python-cql driver and CQL native protocols
int	32-bit signed integer
list	A collection of one or more ordered elements
map	A JSON-style array of literals: {literal: literal, literal: literal ...}
set	A collection of one or more elements
text	UTF-8 encoded string
timestamp	Date plus time, encoded as 8 bytes since epoch integers, strings
varchar	UTF-8 encoded string
varint	Arbitrary-precision integer

Cassandra Data Model Cassandra Data model is based on Google's BigTable.

Each value maps with two strings (row key, column key) and timestamp, similar to Hbase.

The database can be considered as a sparse distributed multi-dimensional sorted map.

Google file system splits the table into multiple tablets (segments of the table) along a row.

Each tablet, called MET AI tablet, maximum size is 200 MB, above which a compression algorithm used.

MET AO is the master-server.

Querying by MET AO server retrieves a MET AI tablet.

During execution of the application, caching of locations of tablets reduces the number of queries.

Cassandra Data Model consists of four main components:

- (i) Cluster: Made up of multiple nodes and keyspaces,
- (ii) Keyspace: a namespace to group multiple column families, especially one per partition,
- (iii) Column: consists of a column name, value and timestamp and
- (iv) Column-family: multiple columns with row key reference.

Cassandra does key space management using partitioning of keys into ranges and assigning different key-ranges to specific nodes.

Following Commands prints a description (typically a series of DDL statements) of a schema element or the cluster:

Consistency Command `CONSISTENCY` shows the current consistency level.
`CONSISTENCY<LEVEL>` sets a new consistency level.

Valid consistency levels are `ANY`, `ONE`, `TWO`, `THREE`, `QUORUM`, `LOCAL_ONE`, `LOCAL_QUORUM`, `EACH_QUORUM`, `SERIAL` AND `LOCAL_SERIAL`.

DESCRIBE CLUSTER

DESCRIBE SCHEMA

DESCRIBE KEYSPACES

DESCRIBE KEYSPACE <keyspace name>

DESCRIBE TABLES

DESCRIBE TABLE <table name>

DESCRIBE INDEX <index name>

DESCRIBE MATERIALIZED VIEW <view name>

DESCRIBE TYPES

DESCRIBE TYPE <type name>

DESCRIBE FUNCTIONS

DESCRIBE FUNCTION <function name>

DESCRIBE AGGREGATES

DESCRIBE AGGREGATE <aggregate function name>

Following are their meanings

1. ALL: Highly consistent. A write must be written to commitlog and memtable on all replica nodes in the cluster.
2. EACH_QUORUM: A write must be written to commitlog and memtable on quorum of replica nodes in all data centers.
3. LOCAL_QUORUM: A write must be written to commitlog and memtable on quorum of replica nodes in the same center.
4. ONE: A write must be written to commit log and memtable of at least one replica node.
5. TWO, THREE: Same as One but at least two and three replica nodes, respectively.
6. LOCAL_ONE: A write must be written for at least one replica node in the local data center.
7. ANY: A write must be written to at least one node.
8. SERIAL: Linearizable consistency to prevent unconditional update.
9. LOCAL_SERIAL: Same as Serial but restricted to the local data center.

Keyspaces A keyspace (or key space) in a NoSQL data store is an object that contains all column families of a design as a bundle. Keyspace is the outermost grouping of the data in the data store. It is similar to relational database. Generally, there is one keyspace per application. Keyspace in Cassandra is a namespace that defines data replication on nodes. A cluster contains one keyspace per node.

Create Keyspace Command `CREATE KEYSPACE <Keyspace Name> WITH replication = {'class': '<Strategy name>', 'replication_factor': '<No. of replicas>'} AND durable_writes = '<TRUE/FALSE>';`

`CREATE KEYSPACE` statement has attributes replication with option class and replication factor, and durable_write.

Default value of durable_writes properties of a table is set to true. That commands the Cassandra to use Commit Log for updates on the current Keyspace true or false. The option is not compulsory.

1. `ALTER KEYSPACE` command changes (alter) properties, such as the number of replicas and the durable_writes of a keyspace: `ALTER KEYSPACE <Keyspace Name> WITH replication = {'class': '<Strategy name>', 'replication_factor': '<No. of replicas>'}`;
2. `DESCRIBE KEYSPACE` command displays the existing keyspaces.
3. `DROP KEYSPACE` command drops a keyspace:
4. Re-executing the drop command to drop the same keyspace will result in configuration exception.
5. Use `KEYSPACE` command connects the client session with a keyspace.

Table 3.15 CQL commands and their functionalities

Command	Functionality
CQLSH	A command line shell for interacting with Cassandra through CQL
HELP	Runs help. This displays the list of all the commands
CONSISTENCY	Shows the current consistency level
EXIT	Terminate the CQL shell
SHOW HOST	Displays the host
SHOW VERSION	Displays the details of current cqlsh session such as host, Cassandra version, or data type assumptions
CREATE KEYSPACE <Keyspace Name>	Creates keyspace with a name
DESCRIBE KEYSPACE <Keyspace Name>	Displays the keyspace with a name
ALTER KEYSPACE <Keyspace Name>	Modifies keyspace with a name
DROP KEYSPACE <Keyspace Name>	Deletes keyspace with a name
CREATE (TABLE COLUMNFAMILY)	Creates a table or column family
COLLECTIONS	Lists the Collections

(1) **Create Table Command:** CREATE TABLE command creates a table in the current keyspace:

```
CREATE (TABLE | COLUMNFAMILY) <tablename>
(<'<column-definition>','<column-definition>'>)
(WITH <option> AND <option>);
```

Primary key is a column used to uniquely identify a row. Therefore, defining a primary key is compulsory while creating a table. A primary key is made of one or more columns of a table.

Example: Create a table *ProductInfo* in the keyspace *lego*, with primary key field *ProductId*.

Use *lego*;

```
Create table ProductInfo(ProductId int primary
key, ProductType text);
```

(2) **Describe Tables Command:** DESCRIBE TABLE Command displays all the tables in the current keyspace:

```
DESCRIBE TABLE <TABLE NAME>;
```

Example: Display the details of a table *ProductInfo*:

```
DESCRIBE TABLE ProductInfo;
```

(3) **Alter Tables Command:**

```
ALTER TABLE Command ALTER (TABLE | COLUMNFAMILY)
<tablename> (ADD | DROP) <column name>
```

The above command adds a column in the table or to delete a column of the table:

Example: Add a column *dateOfManufacturing* in the table *ProductInfo*:

```
ALTER TABLE ProductInfo add dateOfManufacturing
timestamp;
```

* *timestamp* is a datatype used for date fields.

(4) Cassandra CURD Operations: (CURD—Create, Update, Read and Delete data into tables) :

(a) Insert Command:

INSERT command creates data in a table:

```
INSERT INTO <tablename> (<column1 name>, <column2  
name>....) VALUES (<value1>, <value2>....) USING  
<option>
```

(b) Update Command:

UPDATE command updates data in a table. The following keywords are used while updating data in a table:

Where – This clause is used to select the row to be updated.

Set – Set the value using this keyword.

Must – Includes all the columns composing the primary key.

If a given row is unavailable, then UPDATE creates a new row.

```
UPDATE <tablename> SET <column name> = <new value>  
<column name> = <value>.... WHERE <condition>
```

[A WHERE clause can be used only on the columns that are a part of primary key or have a secondary index on them.]

(c) Select Command

SELECT command reads the data from a table. The command can read a whole table, a single column, or a particular cell:

```
SELECT <column name(s)> FROM <Table Name>
```

To select all records:

```
SELECT * FROM <Table Name>
```

To select records that fulfils required condition:

```
SELECT <column1, column2,..> FROM <Table Name>  
where <Condition>
```

```
from ProductInfo where ProductId = 31047;
```

(d) Delete Command

DELETE command deletes data from a table:

```
DELETE FROM <identifier> WHERE <condition>;
```

Example: Delete row from a table where Product id is 31047:

```
DELETE FROM ProductInfo WHERE ProductId = 31047;
```

(5) Creating a Table with List

CREATE Table command is used for creating a table with a list.

The following query creates a table with two columns, one is the primary key and the other has multiple items (List):

```
CREATE TABLE data (<column name>, <data type>  
PRIMARY KEY, <column name list<data type>);
```

Example : Create a sample table *ContactInfo* with three columns: *Sno*, *name* and *EmailId*. To store multiple Email Ids, use a list:

```
create table ContactInfo (Sno int Primary key,  
Name text, emailId list <text>);
```

(6) Insert Command for inserting data into a list

INSERT Command also inserts data into a list. To insert data into the elements in a list, enter all the values separated by a comma within square braces []:

```
INSERT INTO <table name> (column1, column2,)
VALUES (value1, value2, [list value1, list value2,
...])
```

Example: Insert data of three persons into the *ContactInfo* Table:

```
Insert into ContactInfo (Sno, Name, EmailId)
values
(1, 'Rahul', ['rahul@gmail.com',
'rahul@yahoo.com']);
```

```
Insert into ContactInfo (Sno, Name, EmailId)
values (1, 'Geetika', ['geetika@gmail.com',
'geetika@yahoo.com']);
```

```
Insert into ContactInfo (Sno, Name, EmailId)
values (1, 'Deepika', ['deepika@gmail.com',
'deepika@yahoo.com']);
```

(7) Update Command for updating Data into a List

UPDATE command also updates data into a list:

```
UPDATE <table Name> SET <New data> where
<condition>.
```

Example : Add one more email Id to the *emailId* list in *ContactInfo* table :

```
UPDATE ContactInfo SET emailId = emailId +
['preeti@ymail.com'] where SNo=1.
```

Cassandra Client A relational database client connects to DB server using drivers. Java JDBC driver API enables storing and retrieving data. Cassandra has peer-to-peer distribution architecture. Several instances require the clients. The driver enables the use of different languages for connecting to DBs. Cassandra does not include the drivers.

A client-generation layer enables the database interactions. AVRO project provides the client generation layer. Third party sources provide Cassandra clients in Java, Ruby, C#, Python, Perl, PHP, C++, Scala and other languages. The Cassandra client can be included in the applications.

Cassandra Hadoop Support Cassandra 2.1 has Hadoop 2 support. The setup and configuration overlays a Hadoop cluster on the Cassandra nodes. A server is configured for the NameNode and JobTracker. Each Cassandra node then installs the TaskTracker and Data Node.

The nodes in the Cassandra cluster can read data from the data in the Data Node in HDFS as well as from Cassandra. A client application sends the MapReduce input to Job Tracker/Resource Manager. RM/JobTracker sends a MapReduce request of job to the Task Trackers/Node Managers and clients such as MapReduce and Pig. The Reducer output writes to Cassandra. The client gets the results from Cassandra.

MODULE 4

MapReduce, Hive and Pig

Syllabus to Discuss

MapReduce, Hive and Pig: Introduction, MapReduce Map Tasks, Reduce Tasks and MapReduce Execution, Composing MapReduce for Calculations and Algorithms, Hive, HiveQL, Pig.

INTRODUCTION

The data processing layer is the application support layer, while the application layer is the data consumption layer in Big-Data architecture design, when using HDFS, the Big Data processing layer includes the API's of Programs such as **MapReduce** and **Spark**.

- ✓ The application support layer includes HBase which creates column-family data store using other formats such as key-value pairs or JSON file.
- ✓ HBase stores and processes the columnar data after translating into MapReduce tasks to run in HDFS.
- ✓ The support layer also includes Hive which creates SQL-like tables. Hive stores and processes table data after translating it into MapReduce tasks to run in HDFS.
- ✓ Hive creates SQL-like tables in Hive shell. Hive uses HiveQL processes queries, ad hoc (unstructured) queries, aggregation functions and summarizing functions, such as functions to compute maximum, minimum, average of selected or grouped datasets. HiveQL is a restricted form of SQL.
- ✓ The support layer also includes Pig. Pig is a data-flow language and an execution framework.
- ✓ Pig enables the usage of relational algebra in HDFS. MapReduce is the processing framework and YARN is the resource managing framework.

Figure 4.1 shows Big Data architecture design layers: (i) data storage, (ii) data processing and data consumption, (iii) support layer APIs for MapReduce, Hive and Pig running on top of the HDFS Data Store, and (v) application tasks. Pig is a dataflow language, which means that it defines a data stream and a series of transformations.

The smallest unit of data that can be stored or retrieved from the disk is a block. HDFS deals with the data stored in blocks.

The Hadoop application is responsible for distributing the data blocks across multiple nodes. The tasks, therefore, first convert into map and reduce tasks.

This requirement arises because the mapping of stored values is very important. The number of map tasks in an application is handled by the number of blocks of input files.

Reduce task uses those values for further processing such as counting, sorting or aggregating.

Application sub-task assigned for processing needs only the outputs of reduce tasks. For example, a query needs the required response for a data store.

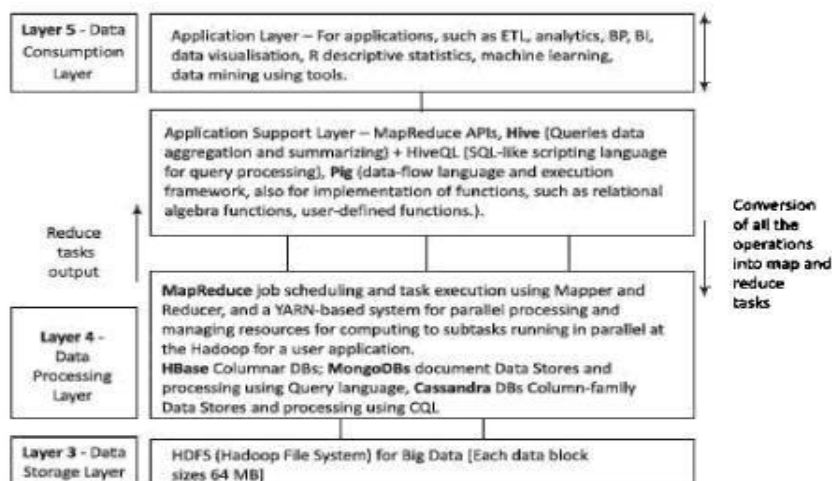


Figure 4.1 Big Data architecture design layers

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

YARN refers to provisioning of running and scheduling parallel programs for map and reduce tasks and allocating parallel processing resources for computing sub-tasks running in parallel at the Hadoop for a user application.

The YARN resources management enables large-scale data analytics using multiple machines (data nodes) in the HDFS cluster.

Script refers to a small program (codes up to few thousand lines of code) in a language used for purposes such as query processing, text processing, or refers to a small code written in a dynamic high-level general-purpose language, such as Python or PERL.

SQL-like scripting language means a language for writing script that processes queries similar to SQL. SQL lets us: (i) write structured queries for processing in DBMS, (ii) create and modify schema, and control the data access, (iii) Create client for sending query scripts, and create and manage server databases, and (iv) view, query and change (update, insert or append or delete) databases.

A theorem known as CAP (Consistency, Availability and £artitions) states that out of three properties, at least two must be present for the application/service/process.

NoSQL relies upon another model known as the BASE model. This model has three principles: Basic availability (the availability of data even in the presence of multiple failures), Soft state (data consistency is the developer's problem and should not be handled by the database).

Eventual consistency (when no new changes occur on existing data, eventually all accesses to that data will return the last updated value).

Data-architecture patterns refer to formats used in NoSQL DBs. The examples are Key-Value Data Stores, Object Data Stores, Column family Big Data Stores, Tabular Data Stores and Document Stores.

Key-Value Data Store refers to a simplest way to implement a schema-less database. A string called key maps to values in a large data string or BLOB (basic large object)..

Object Data Store refers to a repository which stores the (i) objects (such as files, images, documents, folders and business reports), (ii) system metadata which provides information such as filename, creation_date, last_modified, language_used, access_permissions, supported Query languages, and (iii) Custom metadata which provides information such as subject, category and sharing permission.

Tabular Data Store refers to table, column-family or BigTable like Data Store.

Column family Big Data store refers to storage in logical groups of column families. The storage may be similar to columns of sparse matrix. They use a pair of row and column keys to access the column fields.

BigTable Data Store is a popular column-family based Data Store.

Row key, column key and timestamp uniquely identify a value. Google BigTable, HBase and Cassandra DBs use the BigTable Data Store model.

Document Store means a NoSQL DB which stores hierarchical information in a single unit called document. Document stores data in nested hierarchies, for example in XML document object model, JSON formats data model or machine-readable data as one BLOB.

Tuple means an ordered list of elements. An n-tuple relates to set theory, a collection (sequence) of "n" elements. Tuples implement the records.

Collection means a well-defined collection of distinct objects in a set, the objects of a set are the elements. A collection may be analogous to a table of RDBMS. A collection in a database also refers to storage of a number of documents.

Aggregate refers to collection of data sets in the key value, column family or BigTable data stores which usually require sequential processing.

Aggregation function refers to a function to find counts, sum, maximum, minimum, other statistical or mathematical function using a collection of datasets, such as column or column-family.

Sequence refers to an enumerated collection of objects, (the repetitions can be there) which contain members similar to a set. Sequence length equals the number of elements (can also be infinite). Sequence should reflect an order which matters, unlike a set.

Document refers to a container for the number of collections. The container can be a unit of storing data in a database, such as MongoDB.

Natural join is where two tables join based on all common columns. Both the tables must have the same column name and the data type.

MAPREDUCE MAP TASKS, REDUCE TASKS AND MAPREDUCE EXECUTION

Big Data Processing employs the Map Reduce Programming Model. A job means a Map Reduce Program. Each job consists of several smaller unit, called MapReduce Tasks.

A software execution framework in MapReduce programming defines the parallel tasks.

The Hadoop MapReduce implementation uses Java framework.

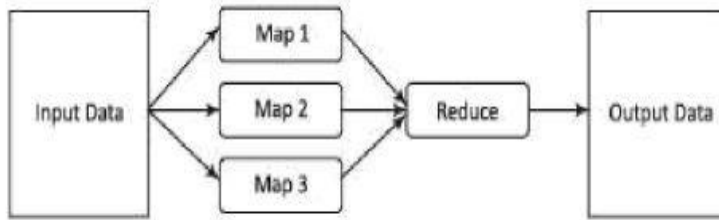


Figure 4.2 MapReduce Programming Model

The model defines two important tasks, namely Map and Reduce.

Map takes input data set as pieces of data and maps them on various nodes for parallel processing.

The reduce task, which takes the output from the maps as an input and combines those data pieces into a smaller set of data. A reduce task always run after the map task (s).

Many real-world situations are expressible using this model.

Inner join is the default natural join. It refers to two tables that join based on common columns mentioned using the ON clause. Inner Join returns all rows from both tables if the columns match.

Node refers to a place for storing data, data block or read or write computations.

Data center in a DB refers to a collection of related nodes. Many nodes form a data center or rack.

Cluster refers to a collection of many nodes.

Keyspace means a namespace to group multiple column families, especially one per partition.

Indexing to a field means providing reference to a field in a document of collections that support the queries and operations using that index. A DB creates an index on the `_id` field of every collection.

The input data is in the form of an HDFS file. The output of the task also gets stored in the HDFS.

The compute nodes and the storage nodes are the same at a cluster, that is, the MapReduce program and the HDFS are running on the same set of nodes.

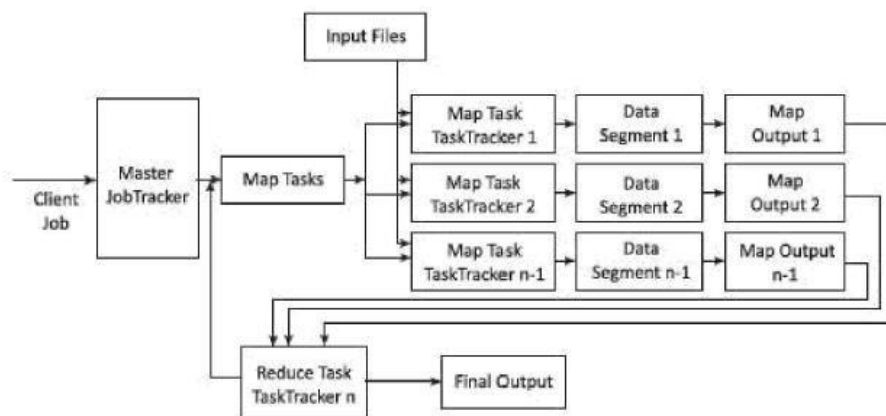


Figure 4.3 MapReduce process on client submitting a job

Figure 4.3 shows MapReduce process when a client submits a job, and the succeeding actions by the JobTracker and TaskTracker.

JobTracker and Task Tracker MapReduce consists of a single master JobTracker and one slave TaskTracker per cluster node.

The **master** is responsible for scheduling the component tasks in a job onto the slaves, monitoring them and re-executing the failed tasks. The slaves execute the tasks as directed by the master.

The data for a MapReduce task is initially at input files. The input files typically reside in the HDFS. The files may be line-based log files, binary format file, multi-line input records, or something else entirely different.

The MapReduce framework operates entirely on key, value-pairs. The framework views the input to the task as a set of (key, value)pairs and produces a set of (key, value) pairs as the output of the task, possibly of different types.

Map-Tasks

Map task means a task that implements a `map()`, which runs user application codes for each key-value pair (**k1**, **v1**). Key **k1** is a set of keys. Key **k1** maps to group of data values (Section 3.3.1). Values **v1** are a large string which is read from the input file(s).

The **output** of `map()` would be zero (when no values are found) or intermediate key-value pairs (**k2**, **v2**). The value **v2** is the information for the transformation operation at the reduce task using aggregation or other reducing functions.

Reduce task refers to a task which takes the output **v2** from the map as an input and combines those data pieces into a smaller set of data using a *combiner*. The reduce task is always performed after the map task.

The **Mapper** performs a function on individual values in a dataset irrespective of the data size of the input. That means that the Mapper works on a single data set. Figure 4.4 shows logical view of functioning of `map()`.

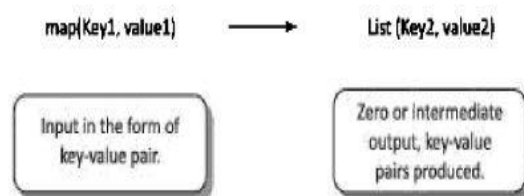


Figure 4.4 Logical view of functioning of `map()`

Hadoop Java API includes Mapper class. An abstract function map() is present in the Mapper class. Any specific Mapper implementation should be a subclass of this class and overrides the abstract function, map ().

The Sample Code for Mapper Class

```
public class SampleMapper extends Mapper<kl, Vl, k2, v2>  
  
{  
  
void map (kl key, Vl value, Context context) throwe IOException,  
InterruptedException  
  
{ .. }
```

Individual Mappers do not communicate with each other.

Number of Maps The number of maps depends on the size of the input files, i.e., the total number of blocks of the input files.

If the input files are of 1TB in size and the block size is 128 MB, there will be 8192 maps. The number of map task Nmap can be explicitly set by using *setNumMapTasks(int)*. Suggested number is nearly 10-100 maps per node. Nmap can be set even higher.

Key-Value Pair

Each phase (Map phase and Reduce phase) of MapReduce has key-value pairs as input and output. Data should be first converted into key-value pairs before it is passed to the Mapper, as the Mapper only understands key-value pairs of data.

Key-value pairs in Hadoop MapReduce are generated as follows:

InputSplit - Defines a logical representation of data and presents a Split data for processing at individual map().

RecordReader - Communicates with the InputSplit and converts the Split into

records which are in the form of key-value pairs in a format suitable for reading by the Mapper.

RecordReader uses **TextInputFormat** by default for converting data into key-value pairs.

RecordReader communicates with the **InputSplit** until the file is read.

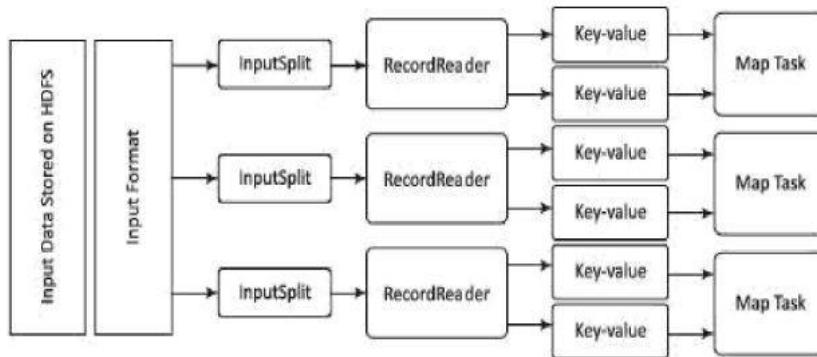


Figure 4.5 Key-value pairing in MapReduce

Figure 4.5 shows the steps in MapReduce key-value pairing.

Generation of a key-value pair in MapReduce depends on the dataset and the required output. Also, the functions use the key-value pairs at four places: `map()` input, `map()` output, `reduce()` input and `reduce()` output.

Grouping by Key

When a map task completes, Shuffle process aggregates (combines) all the Mapper outputs by grouping the key-values of the Mapper output, and the value `v2` append in a list of values. A "Group By" operation on intermediate keys creates **v2**.

Shuffle and Sorting Phase

All pairs with the same group key (**k2**) collect and group together, creating one group for each key.

Shuffle output format will be a List of $\langle k_2, \text{List}(v_2) \rangle$. Thus, a different subset of the intermediate key space assigns to each reduce node.

These subsets of the intermediate keys (known as "partitions") are inputs to the reduce tasks.

Each reduce task is responsible for reducing the values associated with partitions. HDFS sorts the partitions on a single node automatically before they input to the Reducer.

Partitioning

- ✓ The Partitioner does the partitioning. The partitions are the semi-mappers in MapReduce.
- ✓ Partitioner is an optional class. MapReduce driver class can specify the Partitioner.
- ✓ A partition processes the output of map tasks before submitting it to Reducer tasks.
- ✓ Partitioner function executes on each machine that performs a map task.
- ✓ Partitioner is an optimization in MapReduce that allows **local partitioning** before reduce-task phase.
- ✓ The same codes implement the Partitioner, Combiner as well as reduce() functions.
- ✓ Functions for Partitioner and sorting functions are at the mapping node.
- ✓ The main function of a Partitioner is to split the map output records with the same key.

Combiners

Combiners are semi-reducers in MapReduce. Combiner is an optional class. MapReduce driver class can specify the combiner.

The combiner() executes on each machine that performs a map task. Combiners optimize MapReduce task that locally aggregates before the shuffle and sort phase.

The same codes implement both the combiner and the reduce functions, combiner() on map node and reducer() on reducer node.

The main function of a Combiner is to consolidate the map output records with the same key.

The output (key-value collection) of the combiner transfers over the network to the Reducer task as input.

This limits the volume of data transfer between map and reduce tasks, and thus reduces the cost of data transfer across the network. Combiners use grouping by key for carrying out this function.

The combiner works as follows:

- ✓ It does not have its own interface and it must implement the interface at reduce().
- ✓ It operates on each map output key. It must have the same input and output key-value types as the Reducer class.
- ✓ It can produce summary information from a large dataset because it replaces the original Map output with fewer records or smaller records.

Reduced Tasks

Java API at Hadoop includes Reducer class. An abstract function, reduce() is in the Reducer.

- ✓ Any specific Reducer implementation should be subclass of this class and override the abstract reduce().
- ✓ Reduce task implements reduce() that takes the Mapper output (which shuffles and sorts), which is grouped by key-values (k2, v2) and applies it in parallel to each group.
- ✓ Intermediate pairs are at input of each Reducer in order after sorting using the key.

- ✓ Reduce function iterates over the list of values associated with a key and produces outputs such as aggregations and statistics.
- ✓ The reduce function sends output zero or another set of key-value pairs (k3, v3) to the final the output file. Reduce: $\{(k2, \text{list}(v2)) \rightarrow \text{list}(k3, v3)\}$

Sample code for Reducer Class

```
public class ExampleReducer extends Reducer<k2, v2, k3, v3>
```

```
void reduce (k2 key, Iterable<V2> values, Context context) throws
IOException, InterruptedException
```

```
{ ... }
```

Details of Map Reduce processing Steps.

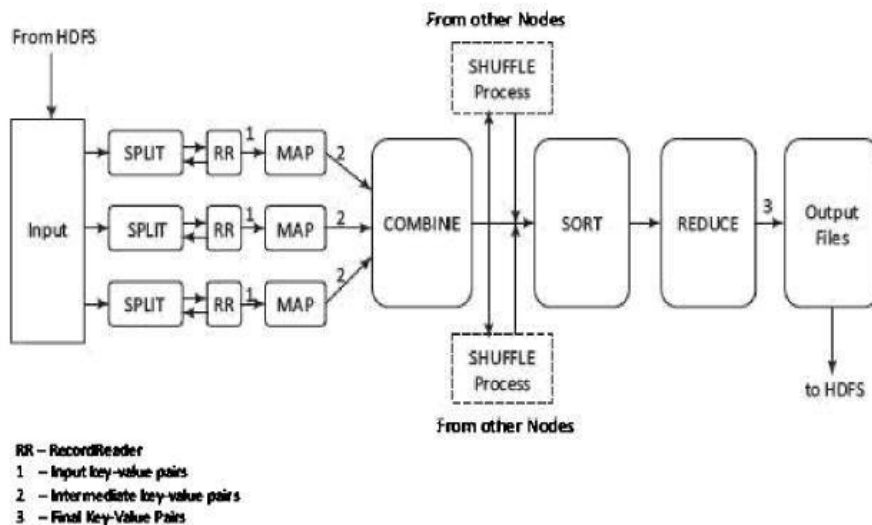


Figure 4.6 MapReduce execution steps

Execution of MapReduce job does not consider how the distributed processing implements. Rather, the execution involves the formatting (transforming) of data at each step

Figure 4.6 shows the execution steps, data flow, splitting, partitioning and sorting on a map node and reducer on reducer node.

Copying with Node Failure

The primary way using which Hadoop achieves fault tolerance is through restarting the tasks.

- ✓ Each task nodes (TaskTracker) regularly communicates with the master node, JobTracker. If a TaskTracker fails to communicate with the JobTracker for a pre-defined period (by default, it is set to 10 minutes), a task node failure by the JobTracker is assumed.
- ✓ The JobTracker knows which map and reduce tasks were assigned to each TaskTracker.
- ✓ If the job is currently in the mapping phase, then another TaskTracker will be assigned to re-execute all map tasks previously run by the failed TaskTracker.
- ✓ If the job is in the reducing phase, then another TaskTracker will re-execute all reduce tasks that were in progress on the failed TaskTracker.
- ✓ Once reduce tasks are completed, the output writes back to the HDFS. Thus, if a TaskTracker has already completed nine out of ten reduce tasks assigned to it, only the tenth task must execute at a different node.

The failure of JobTracker (if only one master node) can bring the entire process down; Master handles other failures, and the MapReduce job eventually completes. When the Master compute-node at which the JobTracker is executing fails, then the entire

MapReduce job must restart. Following points summarize the coping mechanism with distinct Node Failures:

- ✓ MapTaskTracker failure:
 - Map tasks completed or in-progress at TaskTracker, are reset to idle on failure
 - Reduce TaskTracker gets a notice when a task is rescheduled on another TaskTracker
- ✓ Reduce TaskTracker failure:
 - Only in-progress tasks are reset to idle
- ✓ MasterJobTracker failure:
 - Map-Reduce task aborts and notifies the client (in case of one master node).

COMPOSING MAPREDUCE FOR CALCULATIONS ANDALGORITHMS

MapReduce program composition in counting and summing, algorithms for relational algebraic operations, projections, unions, intersections, natural joins, grouping and aggregation, matrix multiplicationand other computations.

Composing Map-Reduce for Calculations

The calculations for various operations compose are:

Counting and Summing

- ✓ The number of alerts or messages generated during a specific maintenance activity of vehicles need counting for a month.
- ✓ From Figure 4.8 showed the pseudocode using emit() in the map() of *Mapper* class. *Mapper* emits 1 for each message generated.
- ✓ The reducer goes through thelist of ls and sums them. Counting is used in the data querying application.

- ✓ For example, count of messages generated, word count in a file, number of cars sold, and analysis of the logs, such as number of tweets per month. Application is also in business analytics field.

Sorting

- ✓ From figure 4.6 illustrated MapReduce execution steps, i.e., dataflow, splitting, partitioning and sorting on a map node and reduce on a reducer node.
- ✓ *Mappers* just emit all items as values associated with the sorting keys which assemble as a function of items.
- ✓ *Reducers* combine all emitted parts into a final list.

Finding Distinct Values (Counting unique values)

Applications such as web log analysis need counting of unique users.

Evaluation is performed for the total number of unique values in each field for each set of records that belongs to the same group.

Two solutions are possible:

- ✓ The *Mapper* emits the dummy counters for each pair of field and group id, and the *Reducer* calculates the total number of occurrences for each such pair.
- ✓ The *Mapper* emits the values and group id, and the *Reducer* excludes the duplicates from the list of groups for each value and increments the counter for each group.
- ✓ The final step is to sum all the counters emitted at the *Reducer*. This requires only one MapReduce job but the process is not scalable, and hence has limited applicability in large data sets.

Collating

- ✓ Collating is a way to collect all items which have the same value of function in one document or file, or a way to process items with the same value of the function together.

- ✓ Examples of applications are producing inverted indexes and extract, transform and load operations.
- ✓ *Mapper* computes a given function for each item, produces value of the function as a key, and the item itself as a value.
- ✓ *Reducer* then obtains all item values using group-by function, processes or saves them into a list and outputs to the application task or saves them.

Filtering or Parsing

- ✓ Filtering or parsing collects only those items which satisfy some condition or transform each item into some other representation.
- ✓ Filtering/parsing include tasks such as text parsing, value extraction and conversion from one format to another.
- ✓ Examples of applications of filtering are found in data validation, log analysis and querying of datasets.
- ✓ *Mapper* takes items one by one and accepts only those items which satisfy the conditions and emit the accepted items or their transformed versions.
- ✓ *Reducer* obtains all the emitted items, saves them into a list and outputs to the application.

Distributed Tasks Execution

- ✓ Large computations divide into multiple partitions and combine the results from all partitions for the final result.
- ✓ Examples of distributed running of tasks are physical and engineering simulations, numerical analysis and performance testing.
- ✓ *Mapper* takes a specification as input data, performs corresponding computations and emits results. *Reducer* combines all emitted parts into the final result.

Graph Processing using Iterative Message Passing

- ✓ Graph is a network of entities and relationships between them. A node corresponds to an entity. An edge joining two nodes corresponds to a relationship.

- ✓ Path traversal method processes a graph. Traversal from one node to the next generates a result which passes as a message to the next traversal between the two nodes. Cyclic path traversal uses iterative message passing.
- ✓ A set of nodes stores the data and codes at a network. Each node contains a list of neighboring node IDs. MapReduce jobs execute iteratively. Each node in an iteration sends messages to its neighbors.
- ✓ Each neighbor updates its state based on the received messages. Iterations terminate on some conditions, such as completion of fixed maximal number of iterations or specified time to live or negligible changes in states between two consecutive iterations.
- ✓ *Mapper* emits the messages for each node using the ID of the adjacent node as a key. All messages thus group by the incoming node. *Reducer* computes the state again and rewrites a node new state.

Cross Correlation

Cross-correlation involves calculation using number of tuples where the items co-occur in a set of tuples of items. If the total number of items is N , then the total number of values = $N \times N$. Cross correlation is used in text analytics. (Assume that items are words and tuples are sentences). Another application is in market-analysis (for example, to enumerate, the customers who buy item x tend to also buy y).

If $N \times N$ is a small number, such that the matrix can fit in the memory of a single machine, then implementation is straightforward.

Two solutions for finding cross correlations are:

- ✓ The *Mapper* emits all pairs and dummy counters, and the *Reducer* sums these counters.
- ✓ The benefit from using combiners is little, as it is likely that all pairs are distinct.

The accumulation does not use in-memory computations as N is very large.

- ✓ The *Mapper* groups the data by the first item in each pair and maintains an associative array ("stripe") where counters for all adjacent items accumulate.
- ✓ The *Reducer* receives all stripes for the leading item, merges them and emits the same result as in the pairs approach.

The grouping:

- ✓ Generates fewer intermediate keys. Hence, the framework has less sorting to do.
- ✓ Greatly benefits from the use of combiners.
- ✓ In-memory accumulation possible.
- ✓ Enables complex implementations.
- ✓ Results in general, faster computations using stripes than "pairs".

Matrix-Vector Multiplication by MapReduce

Numbers of applications need multiplication of $n \times n$ matrix **A** with vector **B** of dimension **n**. Each element of the product is the element of vector **C** of dimension **n**. The elements of C calculate by relation,

$$c_i = \sum_{j=1}^n a_{ij} b_j . \text{ An example of calculations is given below.}$$

$$\text{Assume } \mathbf{A} = \begin{bmatrix} 1 & 5 & 4 \\ 2 & 1 & 3 \\ 4 & 2 & 1 \end{bmatrix} \text{ and } \mathbf{B} = \begin{bmatrix} 4 \\ 1 \\ 3 \end{bmatrix} .$$

$$\text{Multiplication } \mathbf{C} = \mathbf{A} \times \mathbf{B} = \begin{bmatrix} 1 \times 4 + 5 \times 1 + 4 \times 3 \\ 2 \times 4 + 1 \times 1 + 3 \times 3 \\ 4 \times 4 + 2 \times 1 + 1 \times 3 \end{bmatrix}$$

$$\text{Hence, } \mathbf{C} = \begin{bmatrix} 21 \\ 18 \\ 21 \end{bmatrix}$$

Algorithm for using MapReduce: The Mapper operates on A and emits row-wise multiplication of each matrix element and vector element ($a_{ij} \times b_j$ 'V i'). The Reducer executes sum() for summing all values associated with each i and emits the element c_i . Application of the algorithm is found in linear transformation.

Relational – Algebra Operations

Selection

Consider the attributenames (ACVM_ID, Date, chocolate_flavour, daily_sales).
Consider relation

$R = \{(524, 12122017, \text{KitKat}, 82), (524, 12122017, \text{Oreo}, 72), (525, 12122017, \text{KitKat}, 82), (525, 12122017, \text{Oreo}, 72), (526, 12122017, \text{KitKat}, 82), (526, 12122017, \text{Oreo}, 72)\}$.

Selection $ACVM_ID \leq 525$ (R) selects the subset $R = \{(524, 12122017, \text{KitKat}, 82), (524, 12122017, \text{Oreo}, 72), (525, 12122017, \text{KitKat}, 82), (525, 12122017, \text{Oreo}, 72)\}$.

Selection $chocolate_flavour = \text{Oreo}$ selects the subset $\{(524, 12122017, \text{Oreo}, 72), (525, 12122017, \text{Oreo}, 72), (526, 12122017, \text{Oreo}, 72)\}$.

The *Mapper* calls test() for each tuple in a row. When test satisfies the selection criterion then emits the tuple.

The *Reducer* transfers the received input tuple as the output.

Projection

Consider attribute names (ACVM_ID, Date, chocolate_flavour, daily_sales).

Consider relation $R = \{(524, 12122017, \text{KitKat}, 82), (524, 12122017, \text{Oreo}, 72)\}$.
Projection $\Pi_{ACVM_m}(R)$ selects the subset $\{(524)\}$.

Projection, $\Pi_{chocolate_flavour, o.s * daily_sales}$ selects the subset $\{(\text{KitKat}, 0.5 \times 82), (\text{Oreo}, 0.5 \times 72)\}$

The Mapper *calls* test() for each tuple in a row. When the test satisfies, the predicate then emits the tuple (same as in selection).

The *Reducer* transfers the received input tuples after eliminating the possible uplicates. Such operations are used in analytics

Union

Consider,

$R1 = \{(524, 12122017, \text{KitKat}, 82), (524, 12122017, \text{Oreo}, 72)\}$

$R2 = \{(525, 12122017, \text{KitKat}, 82), (525, 12122017, \text{Oreo}, 72)\}$ and

$R3 = \{(526, 12122017, \text{KitKat}, 82), (526, 12122017, \text{Oreo}, 72)\}$

Result of Union operation between R1 and R3 is:

$R1 \cup R3 = \{(524, 12122017, \text{KitKat}, 82), (524, 12122017, \text{Oreo}, 72), (526, 12122017, \text{KitKat}, 82), (526, 12122017, \text{Oreo}, 72)\}$

The *Mapper* executes all tuples of two sets for union and emits all the resultant tuples.

The *Reducer* class object transfers the received input tuples after eliminating the possible duplicates.

Intersection

Consider, $R1 = \{(524, 12122017, \text{Oreo}, 72)\}$

$R2 = \{(525, 12122017, \text{KitKat}, 82)\}$

and $R3 = \{(526, 12122017, \text{KitKat}, 82), (526, 12122017, \text{Oreo}, 72)\}$

Result of Intersection operation between R1 and R3 are

$R1 \cap R3 = \{(12122011, \text{Oreo})\}$

The *Mapper* executes all tuples of two sets for intersection and emits all the resultant tuples.

The *Reducer* transfers only tuples that occurred twice. This is possible only when tuple includes primary key and can occur once in a set. Thus, both the sets contain this tuple.

Difference

Consider:

$R1 = \{(12122017, \text{KitKat}, 82), (12122017, \text{Oreo}, 72)\}$ and

$R3 = \{(12122017, \text{KitKat}, 82), (12122017, \text{Oreo}, 25)\}$

Difference means the tuple elements are not present in the second relation. Therefore, difference

set_1 is $R1 - R3 = (12122017, \text{Oreo}, 72)$ and

set_2 is $R3 - R1 = (12122017, \text{Oreo}, 25)$.

The *Mapper* emits all the tuples and tag. A tag is the name of the set (say, set_1 or set_2 to which a tuple belongs to).

The *Reducer* transfers only tuples that belong to set_1.

Symmetric Difference

Symmetric difference (notation is $A \text{ fl. } B$ (or $A \oplus B$)) is another relational entity. It means the set of elements in exactly one of the two relations A or B. $R3 \oplus R1 = (12122017, \text{Oreo}, 25)$.

The *Mapper* emits all the tuples and tag. A tag is the name of the set (say, set_1 or set_2 this tuple belongs to).

The *Reducer* transfers only tuples that belong to neither set_1 or set_2.

Natural Join

Consider two relations R1 and R2 for tuples a, b and c. Natural Join computes for R1 (a, b) with R2 (b, c). Natural Join is R (a, b, c).

Tuples b joins as one in a Natural Join. The *Mapper* emits the key-value pair (b, (R1, a)) for each tuple (a, b) of R1, similarly emits (b, (R2, c)) for each tuple (b, c) of R2.

The *Mapper* is mapping both with Key for b. The *Reducer* transfers all pairs consisting of one with first component R1 and the other with first component R2, say (R1, a) and (R2, c).

The output from the key and value list is a sequence of key-value pairs. The key is of no use and is irrelevant. Each value is one of the triples (a, b, c) such that (R1, a) and (R2, c) are present in the input list of values.

Grouping and Aggregation by MapReduce

Grouping means operation on the tuples by the value of some of their attributes after applying the aggregate function independently to each attribute. A Grouping operation denotes by $\langle \text{grouping attributes} \rangle_j \langle \text{function-list} \rangle (R)$. Aggregate functions are count(), sum(), avg(), min() and max().

Assume $R = \{(524, 12122017, \text{KitKat}, 82), (524, 12122017, \text{Oreo}, 72), (525, 12122017, \text{KitKat}, 82), (525, 12122017, \text{Oreo}, 72), (526, 12122017, \text{KitKat}, 82), (526, 12122017, \text{Oreo}, 72)\}$.

Chocolate_flavour i count ACVM_ID, sum (daily_sales (chocolate_flavour))

will give the output (524, KitKat, sale_month), (525, KitKat, sale_month), and (524, Oreo, sale_month), (525, Oreo, sale_month), for all **ACVM_IDs**.

The *Mapper* finds the values from each tuple for grouping and aggregates them. The *Reducer* receives the already grouped values in input for aggregation.

Matrix Multiplication

Consider matrices named A (i rows and j columns) and B (rows and k columns) to produce the matrix C; (i rows and k columns). Consider the elements of matrices **A**, **B** and C as follows:

$c_i = \sum_{j=1}^n a_{ij} b_j$. An example of calculations is given below.

Assume $\mathbf{A} = \begin{bmatrix} 1 & 5 & 4 \\ 2 & 1 & 3 \\ 4 & 2 & 1 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} 4 \\ 1 \\ 3 \end{bmatrix}$.

Multiplication $\mathbf{C} = \mathbf{A} \times \mathbf{B} = \begin{bmatrix} 1 \times 4 + 5 \times 1 + 4 \times 3 \\ 2 \times 4 + 1 \times 1 + 3 \times 3 \\ 4 \times 4 + 2 \times 1 + 1 \times 3 \end{bmatrix}$

Hence, $\mathbf{C} = \begin{bmatrix} 21 \\ 18 \\ 21 \end{bmatrix}$

$\mathbf{A} \cdot \mathbf{B} = \mathbf{C}$; Each element evaluates as follow:

First Row of C

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

Second row of C

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

The i^{th} row of C

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

HIVE

Hive was created by Facebook. Hive is a data warehousing tool and is also a data store on the top of Hadoop. An enterprise uses a data warehouse as large data repositories that are designed to enable the tracking, managing, and analyzing the data.

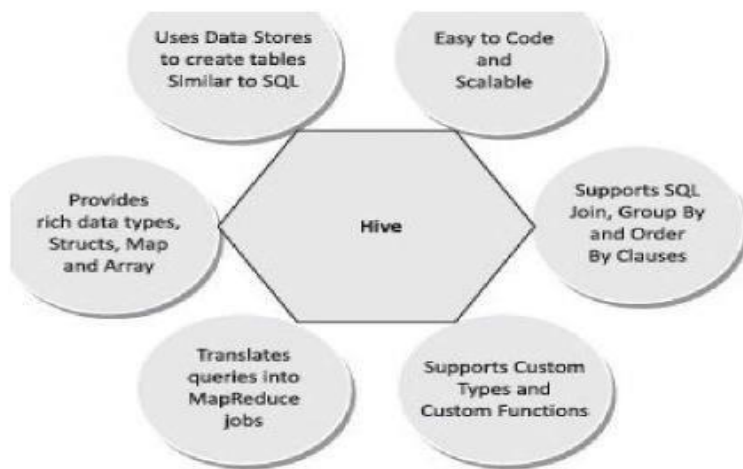


Figure 4.9 Main features of Hive

Hive Characteristics

- ✓ Has the capability to translate queries into MapReduce jobs. This makes Hive scalable, able to handle data warehouse applications, and therefore, suitable for the analysis of static data of an extremely large size, where the fast response-time is not a criterion.
- ✓ Supports web interfaces as well. Application APIs as well as web-browserclients, can access the Hive DB server.
- ✓ Provides an SQL dialect (Hive Query Language, abbreviated HiveQL or HQL).

Results of HiveQL Query and the data load in the tables which store at the

Hadoop cluster at HDFS.

Limitations of Hive is:

- ✓ Not a full database. Main disadvantage is that Hive does not provide update, alter and deletion of records in the database.
- ✓ Not developed for unstructured data.
- ✓ Not designed for real-time queries.
- ✓ Performs the partition always from the last column.

HIVE ARCHITECTURE

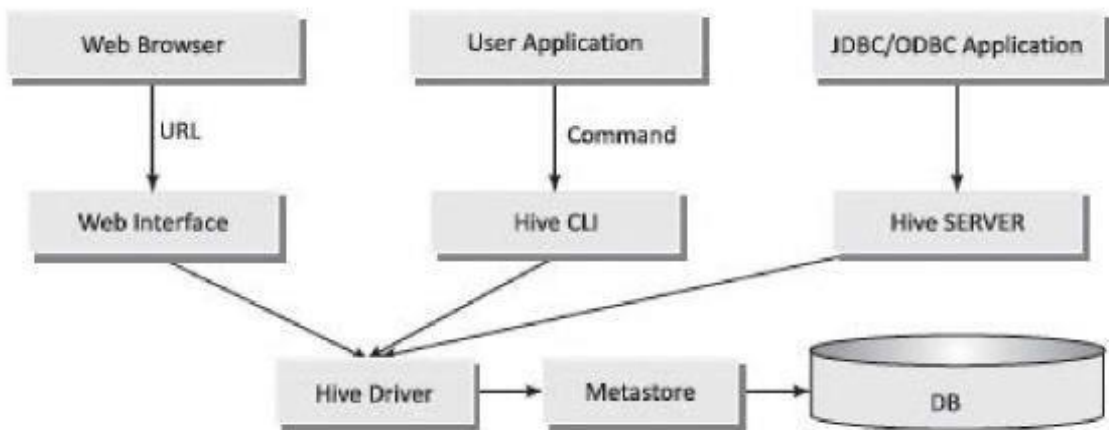


Figure 4.10 Hive architecture

Components of Hive architecture are:

- ✓ **Hive Server (Thrift)** - An optional service that allows a remote client to submit requests to Hive and retrieve results. Requests can use a variety of programming languages.
- ✓ Thrift Server exposes a very simple client API to execute HiveQL statements.
- ✓ **Hive CLI (Command Line Interface)** - Popular interface to interact with Hive. Hive runs in local mode that uses local storage when running the CLI on a Hadoop cluster instead of HDFS.
- ✓ **Web Interface** - Hive can be accessed using a web browser as well. This requires

a HWI Server running on some designated code. The URL *http://hadoop:<port no.> / hwi* command can be used to access Hive through the web.

- ✓ **Metastore** - It is the system catalog. All other components of Hive interact with the Metastore. It stores the schema or metadata of tables, databases, columns in a table, their data types and HDFS mapping.
- ✓ **Hive Driver** - It manages the life cycle of a HiveQL statement during compilation, optimization and execution.

Comparison with RDBMS

Hive is a DB system which defines databases and tables. Hive analyzes structured data in DB. Hive has certain differences with RDBMS.

Characteristics	Hive	RDBMS
Record level queries	No Update and Delete	Insert, Update and Delete
Transaction support	No	Yes
Latency	Minutes or more	In fractions of a second
Data size	Petabytes	Terabytes
Data per query	Petabytes	Gigabytes

Query language	HiveQL	SQL
Support JDBC/ODBC	Limited	Full

Hive Data Types and File Formats

Hive defines various primitive, complex, string, date/time, collection data types and file formats for handling and storing different data formats. The following Table gives primitive, string, date/time and complex Hive data types and their descriptions.

Data TypeName	Description
TINYINT	1 byte signed integer. Postfix letter is Y.
SMALLINT	2 byte signed integer. Postfix letter is S.
INT	4 byte signed integer
BIGINT	8 byte signed integer. Postfix letter is L.
FLOAT	4 byte single-precision floating-point number
DOUBLE	8 byte double-precision floating-point number
BOOLEAN	True or False
TIMESTAMP	UNIX timestamp with optional nanosecond precision. It supports Oava .sql.Timestamp format "YYYY-MM-DD HH:MM:SS.ffffff"
DATE	YYYY-MM-DD format
VARCHAR	1 to 65355 bytes. Use single quotes("") or double quotes("")

CHAR	255 bytes
DECIMAL	Used for representing immutable arbitrary precision. DECIMAL (precision,scale) format

The following Table gives Hive three Collection data types and their descriptions.

File Format	Description
Text file	The default file format, and a line represents a record. The delimiting characters separate the lines. Text file examples are CSV, TSV,JSON and XML(Section 3.3.2).
Sequenti alfile	Flat file which stores binary key-value pairs, and supports compression.
RCFile	Record Columnar file (Section 3.3.3.3).
ORCFILE	ORC stands for Optimized Row Columnar which means it can store data in an optimized way than in the other file formats (Section 3.3.3.4).

HIVE Data Model

Name	Description
Database	Namespace for tables
Tables	Similar to tables in RDBMS Support filter, projection, join and union operationsThe table data stores in a directory in HDFS

Partitions	Table can have one or more partition keys that tell how the data stores
Buckets	Data in each partition further divides into buckets based on hash of a column in the table. Stored as a file in the partition directory.

Hive Integration and Workflow Steps

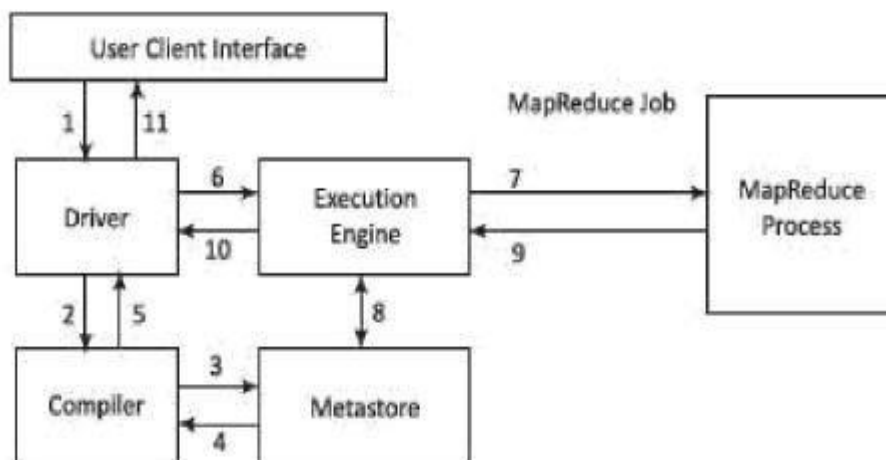


Figure 4.11 Dataflow sequences and workflow steps

The workflow steps are as follows :

Execute Query: Hive interface (CLI or Web Interface) sends a query to DatabaseDriver to execute the query.

Get Plan: Driver sends the query to query compiler that parses the query to check the syntax and query plan or the requirement of the query.

Get Metadata: Compiler sends metadata request to Metastore (of any database, such as MySQL).
Send Metadata: Metastore sends metadata as a response to compiler.
Send Plan: Compiler checks the requirement and resends the plan to driver. The parsing and compiling of the query is complete at this place.
Execute Plan: Driver sends the execute plan to execution engine.
Execute Job: Internally, the process of execution job is a MapReduce job. The execution engine sends the job to JobTracker, which is in Name node and it assigns this job to TaskTracker, which is in Data node. Then , the query executes the job.
Metadata Operations: Meanwhile the execution engine can execute the metadata operations with Metastore.
Fetch Result: Execution engine receives the results from Data nodes.
Send Results: Execution engine sends the result to Driver.
Send Results: Driver sends the results to Hive Interfaces.

Hive Built-in Functions

Return Type	Syntax	Description
BIGINT	round(doublea)	Returns the rounded BIGINT (8 Byte integer) value of the 8 Byte double-precision floating point number a
BIGINT	floor(doublea)	Returns the maximum BIGINT value that is equal to or less thanthe double.

BIGINT	ceil(double a)	Returns the minimum BIGINT value that is equal to or greater than the double.
double	rand(), rand(int seed)	Returns a random number (double) that distributes uniformly from 0 to 1 and that changes in each row. Integer seed ensures that random number sequence is deterministic.
string	concat(string str1, string str2, ...)	Returns the string resulting from concatenating str1 with str2,
string	substr(string str, int start)	Returns the substring of str starting from a start position till the end of string str.
string	substr(string str, int start, int length)	Returns the substring of str starting from the start position with the given length.
string	upper(string str), ucase (string str)	Returns the string resulting from converting all characters of str to upper case.
string	lower(string str), lcase(string str)	Returns the string resulting from converting all characters of str to lower case.
string	trim(string str)	Returns the string resulting from trimming spaces from both ends. trim ('12A34 56') returns '12A3456'

string	ltrim(string str); rtrim(stringstr)	Returns the string resulting from trimming spaces (only one end, left or right hand side or right-handside spaces trimmed). ltrim('12A34 56') returns '12A3456' and rtrim(' 12A34 56 ')returns '12A3456'.
string	rtrim(stringstr)	Returns the string resulting from trimming spaces from the end (right hand side) of str.

int	year(string date)	Returns the year part of a date or a timestamp string.
int	month(string date)	Returns the month part of a date or a timestamp string .
int	day(string date)	Returns the day part of a date or a timestamp string.

HIVEQL

- ✓ Hive Query Language (abbreviated HiveQL) is for querying the large datasets which reside in the HDFS environment.
- ✓ HiveQL script commands enable data definition, data manipulation and query processing.
- ✓ HiveQL supports a large base of SQL users who are acquainted with SQL to extract information from data warehouses.

HiveQL Process Engine	HiveQL is similar to SQL for querying on schema information at the Metastore. It is one of the replacements of traditional approach for MapReduce program . Instead of writing MapReduce program in Java, we can write a query for MapReduce job and process it.
Execution Engine	The bridge between HiveQL process Engine and MapReduce is Hive Execution Engine. Execution engine processes the query and generates results same as MapReduce results. It uses the flavor of MapReduce.

HiveQL Data Definition Language (DDL)

HiveQL database commands for data definition for DBs and Tables are CREATE DATABASE, SHOW DATABASE {list of all DBs}, CREATE SCHEMA, CREATE TABLE.

Following are HiveQL commands which create a table:

```
CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [<database name>.]
```

```
<table name>
```

```
[(<column name> <data type> [COMMENT <column comment>], ...)]  
[COMMENT <table comment>]
```

```
[ROW FORMAT <row format>][STORED AS <file format>]
```

A command is

```
CREATE DATABASE|SCHEMA [IF NOT EXISTS] <database name>;
```

IF NOT EXISTS is an optional clause. The clause notifies the user that a database with the same name already exists. SCHEMA can be also created in place of DATABASE using this command

A command is written to get the list of all existing databases.
SHOW DATABASES;

A command is written to delete an existing database.

```
DROP (DATABASE|SCHEMA) [RESTRICT | CASCADE]; [IF EXISTS]  
<database name>
```

HiveQL Data Manipulation Language (DML)

HiveQL commands for data manipulation are USE <database name>, DROP DATABASE, DROP SCHEMA, ALTER TABLE, DROP TABLE, and LOAD DATA.

The following is a command for inserting (loading) data into the Hive DBs.

```
LOAD DATA [LOCAL] INPATH '<file path>' [OVERWRITE] INTO  
TABLE <table name> [PARTITION (partcol1=val1,partcol2=val2 ...)]
```

LOCAL is an identifier to specify the local path. It is optional. OVERWRITE is optional to overwrite the data in the table. PARTITION

is optional. val1 is value assigned to partition column 1 (partcol1) and val2 is value assigned to partition column 2 (partcol2).

HiveQL For Querying the Data

Partitioning and storing are the requirements. A data warehouse should have a large number of partitions where the tables, files and databases store. Querying then requires sorting, aggregating and joining functions.

Querying the data is to SELECT a specific entity *satisfying* a condition, *having* presence of an entity or selecting specific entity using GroupBy .

```
SELECT [ALL | DISTINCT] <select expression>, <selectexpression>, ...  
FROM <table name>  
  
[WHERE <where condition>] [GROUP BY <column List>] [HAVING  
<having condition>]  
  
[CLUSTER BY <column List>] [DISTRIBUTE BY <columnList>] [SORT  
BY <column List>]]  
  
[LIMIT number];
```

PIG

- ✓ It is an abstract over MapReduce
- ✓ It is an execution framework for parallel processing
- ✓ Reduces the complexities of writing a MapReduce program
- ✓ Is a high-level dataflow language. Dataflow language means that a Pig operation node takes the inputs and generates the output for the next node
- ✓ Is mostly used in HDFS environment
- ✓ Performs data manipulation operations at files at data nodes in Hadoop.

Applications of Apache Pig

- ✓ Analyzing large datasets
- ✓ Executing tasks involving adhoc processing
- ✓ Processing large data sources such as web logs and streaming online data
- ✓ Data processing for search platforms. Pig processes different types of data
- ✓ Processing time sensitive data loads; data extracts and analyzes quickly.

Differences between Pig and MapReduce

Pig	MapReduce
A dataflow language	A data processing paradigm
High level language and flexible	Low level language and rigid
Performing Join, filter, sorting or ordering operations are quite simple	Relatively difficult to perform Join, filter, sorting or ordering operations between datasets
Programmer with a basic knowledge of SQL can work conveniently	Complex Java implementations require exposure to Java language
Uses multi-query approach, thereby reducing the length of the codes significantly	Require almost 20 times more the number of lines to perform the same task
No need for compilation for execution; operators convert internally into MapReduce jobs	Long compilation process for Jobs
Provides nested data types like tuples, bags and maps	No such data types

Differences between Pig and SQL

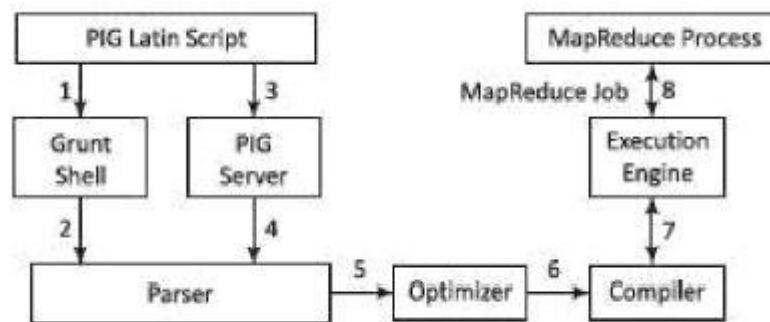
Pig	SQL
Pig Latin is a procedural language	A declarative language
Schema is optional, stores data without assigning a schema	Schema is mandatory
Nested relational data model	Flat relational data model
Provides limited opportunity for Query optimization	More opportunity for query optimization

Pig and Hive codes, both create MapReduce jobs when execute. Hive in some cases, operates on HDFS in a similar way Apache Pig does.

Differences between Pig and Hive

Pig	Hive
Originally created at Yahoo	Originally created at Facebook
Exploits Pig Latin language	Exploits HiveQL
Pig Latin is a dataflow language	HiveQL is a query processing language
Pig Latin is a procedural language and it fits in pipeline paradigm	HiveQL is a declarative language
Handles structured, unstructured and semi-structured data	Mostly used for structured data

Pig Architecture



The three ways to execute scripts are:

1. **Grunt Shell:** An interactive shell of Pig that executes the scripts.
2. **Script File:** Pig commands written in a script file that execute at Pig Server.
3. **Embedded Script:** Create UDFs for the functions unavailable in Pig built in operators. UDF can be in other programming languages. The UDFs can embed in Pig Latin Script file.

Parser A parser handles Pig scripts after passing through Grunt or Pig Server. The Parser performs type checking and checks the script syntax. The output is a Directed Acyclic Graph (DAG).

Acyclic means only one set of inputs are simultaneously at a node, and only one set of output generates after node operations.

DAG represents the Pig Latin statements and logical operators. Nodes represent the logical operators. Edges between sequentially traversed nodes represent the dataflows.

Optimizer The DAG is submitted to the logical optimizer. The optimization activities, such as split, merge, transform and reorder operators execute in this phase. The optimization is an automatic feature.

The optimizer reduces the amount of data in the pipeline at any instant of time, while processing the extracted data. It executes certain functions for carrying out this task, as explained as follows:

PushUpFilter: If there are multiple conditions in the filter and the filter can be split, Pig splits the conditions and pushes up each condition separately. Selecting these conditions at an early stage helps in reducing the number of records remaining in the pipeline.

PushDownfor EachFlatten: Applying flatten, which produces a cross product between a complex type such as a tuple, bag or other fields in the record, as late as possible in the plan. This keeps the number of records low in the pipeline.

ColumnPruner: Omits never used columns or the ones no longer needed, reducing the size of the record. This can be applied after each operator, so that the fields can be pruned as aggressively as possible.

MapKeyPruner: Omits never used map keys, reducing the size of the record.

Limit Optimizer: If the limit operator is immediately applied after *load* or *sort* operator, Pig converts the load or sort into a limit-sensitive implementation, which does not require processing the whole dataset. Applying the limit earlier reduces the number of records.

Compiler The compiler compiles after the optimization process. The optimized codes are a series of MapReduce jobs.

Execution Engine Finally, the MapReduce jobs submit for execution to the engine. The MapReduce jobs execute and it outputs the final result.

Apache- Pig Grunt Shell

Main use of Grunt shell is for writing Pig Latin scripts. Any shell command invokes using `sh` and `ls`. Syntax of `sh` command is:

```
grunt> sh shell command parameters
```

Syntax of `ls` command:

```
grunt> sh ls
```

Pig Latin Data Model

Pig Latin supports primitive data types which are atomic or scalar data types. Atomic data types are `int`, `float`, `long`, `double`, `char[]`, `byte []`.

The language also defines complex data types. Complex data types are `tuple`, `bag` and `map`.

Data types and examples

Data type	Description	Example
bag	Collection of tuples	{(1,1), (2,4)}
tuple	Ordered set of fields	(1,1)
map (data map)	Set of key-value pairs	[Number#l]
int	Signed 32-bit integer	10
long	Signed 64-bit integer	l0L or 10l
float	32-bit floating point	22.7F or 22.7f
double	64-bit floating point	3.4 or 3.4e2 or 3.4E2
chararray	Char [], Character array	data analytics
bytearray	BLOB (Byte array)	ffoo

Pig Latin and Developing Pig Latin scripts

Pig Latin enables developing the scripts for data analysis. A number of operators in Pig Latin help to develop their own functions for reading, writing and processing data. Pig Latin programs execute in the Pig run-time environment.

Pig Latin

- ✓ Basic constructs to process the data.

- ✓ Include schemas and expressions.
- ✓ End with a semicolon.
- ✓ LOAD statement reads the data from file system, DUMP displays the result and STORE stores the result.
- ✓ Single line comments begin with - - and multiline begin with/* and end with*/
- ✓ Keywords (for example, LOAD, STORE, DUMP) are not case-sensitive. Function names, relations and paths are case-sensitive.

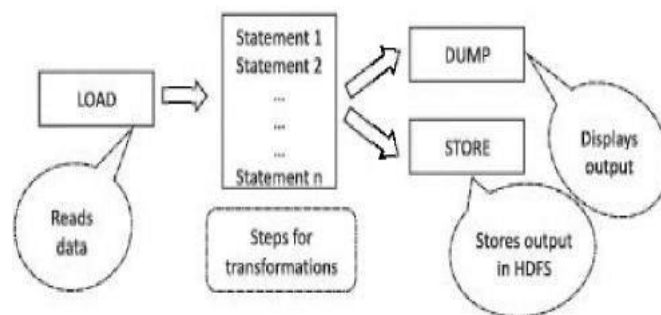


Figure 4.15 Order of processing Pig statements—Load, dump, and store

Apache Pig Execution

Pig Execution Modes Local Mode: All the data files install and run from a local host using the local file system. Local mode is mostly used for testing purpose.

COMMAND:

```
pig -x local
```

MapReduce Mode:

All the data files load or process that exists in the HDFS. A MapReduce job invokes in the back-end to perform a particular operation on the data that exists in the HDFS when a Pig Latin statement executes to process the data.

COMMAND:

```
pig -x mapreduce or pig
```

Pig Latin Script Execution Modes

- ✓ Interactive Mode - Using the Grunt shell.

- ✓ Batch Mode - Writing the Pig Latin script in a single file with .pig extension.
- ✓ Embedded Mode - Defining UDFs in programming languages such as Java, and using them in the script.

Commands

- ✓ To get the list of pig commands: *pig-help*;
- ✓ To get the version of pig: *pig-version*.
- ✓ To start the Grunt shell, write the command: *pig*

LOAD Command The first step to a dataflow is to specify the input.

Load statement in Pig Latin loads the data from PigStorage.

To load data from HBase: book load 'MyBook' using HBaseStorage();

For reading CSV file, PigStorage takes an argument which indicates which character to use as a separator.

For example,

```
book = LOAD 'PigDemo/Data/Input/myBook.csv' USING PigStorage (,);
```

To specify the data-schema for loading: book = LOAD 'MyBook' AS (name, author, edition, publisher);

Store Command Pig provides the store statement for writing the processed **data after the processing is complete. It is the mirror image of the load statement in certain ways.**

By default, Pig stores data on HDFS in a tab-delimited file using PigStorage:

```
STORE processed into '/PigDemo/Data/Output/Processed';
```

To store in HBaseStorage with a *using* clause: STORE processed into 'processed' using HBaseStorage();

To store data as comma-separated text data, PigStorage takes an argument to indicate which character to use as a separator: STORE processed into 'processed' using PigStorage(',');

Dump Command Pig provides dump command to see the processed data on the screen. This is particularly useful during debugging and prototyping sessions. It can also be useful for quick adhoc jobs.

The following command directs the output of the Pig script on the display screen:

DUMP processed;

Relational Operations

The relational operations provided at Pig Latin operate on data. They transform data using sorting, grouping, joining, projecting and filtering. Followings are the basic relational operators:

Foreach FOREACH gives a simple way to apply transformations based on columns. It is Pig's projection operator.

Module 5

OBJECTIVES

- Introduction
- Estimating the Relationships, Outliers, Variances, Probability Distributions and Correlations
- Simple Linear Regression
- Finding Similar Items, Similarity of Sets and Collaborative Filtering
- Frequent Itemset and Association Rule Mining
- Text Mining
- Web Mining, Web Content and Web Usage Analytics
- Page Rank, Structure of Web and Analyzing a Web Graph
- Social Networks as Graphs and Social Network Analytics
- SimRank
- Counting Triangles and Graph Matches

Machine Learning Algorithms for Big Data Analytics

- Analytics uses the mathematical equations, formulae and models. Analytics also uses the statistics, AI, ML and DL, and predict the behavior of entities, objects and events. Statistics refers to studying organization, analysis of a collection of data, making interpretations and presentation of analyzed results.

Machine Learning - Definition and Usage Examples

- Artificial Intelligence (AI) refers to the science and engineering of making computers perform tasks, which normally require human intelligence. For example, tasks such as predicting future results, visual perception, speech recognition, decision making and natural language processing.
- Two concepts in AI, 'machine learning' and 'deep learning' provide powerful tools for advanced analytics and predictions.

- Machine Learning (ML) is a field of computer science based on AI which deals with learning from data in three phases, i.e. collect, analyze and predict. It does not rely on explicitly programmed instructions.
 - An ML program learns the behavior of a process. The program uses data generated from various sources for training. Learning from the outcomes from common inputs improves future performance from previous outcomes. Learning applies in many fields of research and industry. Learning from study of data enables efficient and logical decisions for future actions.
 - Advanced ML techniques use unsupervised, semi-supervised or supervised learning. Supervised learning uses a known dataset (called training dataset). Learning enables creation of a model program for evaluating outcomes. The program makes future predictions and leads to knowledge discovery. Supervised learning uses output datasets, which are used to train a machine (program) such that the program leads to the desired outputs. Unsupervised learning does not use output datasets to train a machine.
- a convention for fonts when denoting an absolute value, mean value, function value, vector element, set member, entity or variable using a character or set of characters, entities or elements.
 - $|u|$ represents absolute value of u , means value without sign. For example, consider $|-3|$ and $|+3|$, the value of both is 3.
 - \bar{x} represents mean, average or expected value of x .
 - $F(y, x)$ represents a function with an expression, which finds value of F from the given values of y and x . $F(y, x)$ values depend on one or more dependent variables as a function of one or more independent variables. For example, F depends y as well as x is $F(y, x) = 1/\sqrt{(y+x)^2 + k^2}$. The F also depends on constant k . Another example is $y = F(x)$, for example, $y = \cos(x)$. $F(x)$ represents a function F , which gives value of y , is a dependent variable. The x is an independent variable.
 - \mathbf{V} denotes a vector \mathbf{V} ($V_1, V_2 \dots$). \mathbf{V} is in bold font. V_1 and V_2 are in text font and are elements 1 and 2 of \mathbf{V} . The \mathbf{V} consists of number of elements $V_1, V_2 \dots$
 - $\|\mathbf{U}\|$ represents length of vector \mathbf{U} .
 - \mathcal{S} denotes a set \mathcal{S} ($A, B, C \dots$). Font \mathcal{S} is in French script MT or distinct font for English \mathcal{S} . The A, B and C are in text font (no bold), and are the members of \mathcal{S} . The members can be vectors or subsets. They, when denoted in bold, represent vector elements.

ESTIMATING THE RELATIONSHIPS, OUTLIERS, VARIANCES, ROBABILITY DISTRIBUTIONS AND CORRELATIONS

- Methods of studying relationships use variables.

Types of variables used are as follows:

1. Independent variables represent directly measurable characteristics.

For example, year of sales figure or semester of study.

Dependent variables represent the characteristics.

For example, profit during successive years or grades awarded in successive semesters.

2. Predictor variable is an independent variable, which computes a dependent variable using some equation, function or graph, and does a prediction. For example, predicts sales growth of a car model after five years from given input datasets for the sales, or predicts sentiments about higher sales of particular category of toys next year.

3. Outcome variable represents the effect of manipulation(s) using a function, equation or experiment. For example, CGPA (Cumulative Grade Points Average) of the student or share of profit to each shareholder in a year using profit as the dependent variable. CGP A of a student computes from the grades awarded in the semesters for which student completes his/her studies.

4. Explanatory variable is an independent variable, which explains the behavior of the dependent variable, such as linearity coefficient, non-linear parameters or probabilistic distribution of profit-growth as a function of additional investment in successive years.

5. Response variable is a dependent variable on which a study, experiment or computation focuses. For example, improvement in profits over the years from the investments made in successive years or improvement in class performance is measured from the extra teaching efforts on individual students of a class.

6. Feature variable is a variable representing a characteristic. For example, apple feature red, pink, maroon, yellowish, yellowish green and green. Feature variables are generally represented by text characters. Numbers can also represent features. For example, red with 1, orange with 2, yellow with 3, yellowish green 4 and green 5.

7. Categorical variable is a variable representing a category. For example, car, tractor and truck belong to the same category, i.e., a four-wheeler automobile. Categorical variables are generally represented by text characters

Relationships-Using Graphs, Scatter Plots and Charts

- A relationship between two or more quantitative dependent variables with respect to an independent variable can be well-depicted using graph, scatter plot or chart with data points, shown in distinct shapes. Conventionally, independent variables are on the x-axis, whereas the dependent variables on the y-axis in a graph. A line graph uses a line on an x-y axis to plot a continuous function.
- A scatter plot is a plot in which dots or distinct shapes represent values of the dependent variable at the multiple values of the independent variable. Whether two variables are related to each other or not, can be derived from statistical analysis using scatter plots.
- Whether two variables are related to each other or not, can be derived from statistical analysis using scatter plots.
- A data point is (x_i, Y_i) when dependent variable value = Y_i at the independent variable value = x_i . The

1. Linear and Non-linear Relationships

- A linear relationship exists between two variables, say x and y , when a straight line ($y = a_0 + a_1 \cdot x$) can fit on a graph, with at least some reasonable degree of accuracy. The a_1 is the linearity coefficient. For example, a scatter chart can suggest a linear relationship, which means a straight line.

Figure 6.1 shows a scatter plot, which fits a linear relationship between the number of students opting for computer courses in years between 2000 and 2017.

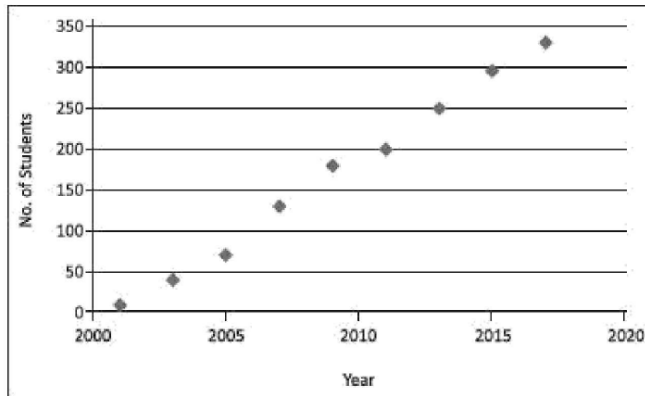


Figure 6.1 Scatter plot for linear relationship between students opting for computer courses in years between 2000 and 2017

A linear relationship can be positive or negative. A positive relationship implies if one variable increases in value, the other also increases in value. A negative relationship, on the other hand, implies when one increases in value, the other decreases in value. Perfect, strong or weak linearship categories depend upon the bonding between the two variables.

A non-linear relationship is said to exist between two quantitative variables when a curve ($y = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \dots$) can be used to fit the data points. The fit should be with at least some reasonable degree of accuracy for the fitted parameters, a_0 , a_1 , $a_2 \dots$ Expression for y then generally predicts the values of one quantitative variable from the values of the other quantitative variable with considerably more accuracy than a straight line.

Consider an example of non-linear relationship: The side of a square and its area are not linear. In fact, they have quadratic relationship. If the side of a square doubles, then its area increases four times. The relationship predicts the area from the side

Figure 6.2 shows a scatter plot in case of a non-linear relationship between side of square and its area

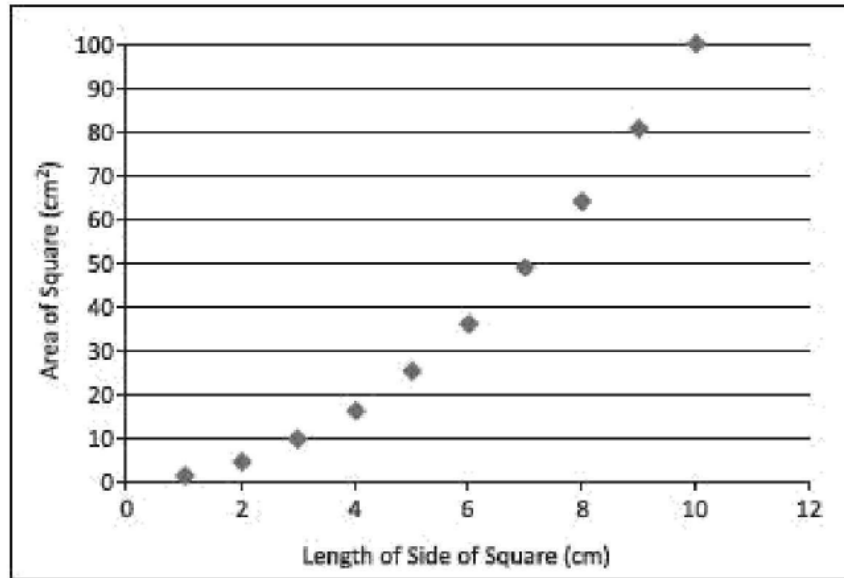


Figure 6.2 Scatter plot in case of a non-linear relationship between side of square and its area

Estimating the Relationships

Estimating the relationships means finding a mathematical expression, which gives the value of the variable according to its relationship with other variables. For example, assume Y_m = sales of a car model m in x th year of the start of manufacturing that model.

Assume that computations show that the y_m relates by a mathematical expression ($y_m = a_0 + a_1.x_m + a_2.x_m^2$) up to an acceptable degree of accuracy, when $a_0 = 490$, $a_1 = 10$ and $a_2 = 5$.

Estimated first year sales, $Y_m(1) = (490 + 10) = 500$, second year $Y_m(2) = (490 + 10 \times 2 + 5 \times 2^2) = 530$, third year $Y_m(3) = (490 + 10 \times 3 + 5 \times 3^2) = 565$, if fit with the desired accuracy, then the results are showing that the expression of y_m estimates the relationship between model m sales in next and other years. The y_m can also predict the sales in 6th or later years. Predictions are up to a certain degree of certainty.

Outliers

Outliers are data, which appear as they do not belong to the dataset. Outliers are data points that are numerically far distant from the rest of the points in a dataset, are termed as outliers. Outliers show significant variations from the rest of the points (Section 1.5.2.2). Identification of outliers is important to improve data quality or to

detect an anomaly. The estimating parameters mathematically, statistically, describing an outcome, predicting a dependent variable value, or taking the decisions based on the datasets given for the analysis are sensitive to the outliers.

There are several reasons for the presence of outliers in relationships.

Some of these are:

- Anomalous situation
- Presence of a previously unknown fact
- Human error (errors due to data entry or data collection)
- Participants intentionally reporting incorrect data (This is common in self-reported measures and measures that involve sensitive data which participant doesn't want to disclose)
- Sampling error (when an unfitted sample is collected from population).

Population means any group of data, which includes all the data of interest. For example, when analysing 1000 students who gave an examination in a computer course, then the population is 1000. 100 games of chess will represent the population in analysis of 100 games of chess of a grandmaster.

Sample means a subset of the population. Sample represents the population for uses, such as analysis and consists of randomly selected data.

Variance

A random variable is a variable whose possible values are outcomes of a random phenomenon.

A random variable is a function that maps the outcomes of unpredictable processes to numerical quantities.

A random variable is also called stochastic variable or random quantity. Randomness can be around some expected mean value or outcome, and with some normal deviation.

Variance measures by the sum of squares of the difference in values of a variable with respect to the expected value. Variance can alternatively be a sum of squares of the difference with respect to value at an origin. Variance indicates how widely data points in a dataset vary. If data points vary greatly from the mean value in a dataset, the

variance is large; otherwise, the variance is less. The variance is also a measure of dispersion with respect to the expected value.

A high variance indicates that the data in the dataset is very much spread out over a large area (random dataset), whereas a low variance indicates that the data is very similar in nature.

No variance is sometimes hard to understand in real datasets. The following example illustrates no variance

EXAMPLE 6.1

Consider an examination where everyone gets the same grades. What does it signify?

SOLUTION

Some measurement problem may have taken place in a situation where either the semester examination questions were so easy that everyone got full marks, or it was so hard that everyone got a zero. Now consider the two types of examinations. After each examination, everyone gets the same score on the test, i.e., everyone gets 'A' grade in one test and everyone gets 'B' in the second test. This is again not telling much

about the study or intelligent quotient of the students. Now, these no variance results signify the extreme case and hard to understand or explain. But in general, differences in scores are always found.

Standard Deviation and Standard Error Estimates

The variance is not a standalone statistical parameter. Estimations of other statistical parameters, such as standard deviation and standard error are also used.

Standard Deviation With the help of variance, one can find out the standard deviation. Standard deviation, denoted by s , is the square root of the variance. The s says, "On an average how far do the data points fall from the mean or expected outcome?" Though the interpretation is the same as variance but s is squared rooted, therefore, less susceptible to the presence of outliers.

The formulae for the population and the sample standard deviations are as follows:

$$\text{The Population Standard Deviation: } \sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \mu)^2} \quad (6.1a)$$

$$\text{The Sample Standard Deviation: } \sigma = \sqrt{\frac{1}{S-1} \sum_{i=1}^S (x_i - \bar{x})^2}, \quad (6.1b)$$

where N is number of data points in population, S is number in the sample, μ is expected in the population or average value of x, and \bar{x} is expected x in the sample.

Standard Error The standard error estimate is a measure of the accuracy of predictions from a relationship. Assume the linear relationship in a scatter plot of y (Figure 6.1). The scatter plot line, which fits, is defined as the line that minimizes the sum of squared deviations of prediction (also called the sum of squares error). The standard error of the estimate is closely related to this quantity and is defined below

$$\sigma_{\text{est}} = \sqrt{\frac{\sum (y - y')^2}{N}}, \quad \dots (6.2)$$

where σ_{est} is the standard error in the estimate, y is an observed value, y' is a predicted value, and N is the number of values observed. The standard error estimate is a measure of the dispersion (or variability) in the predicted values from the expression for relationship. Following are three interpretations from the σ_{est} :

1. When σ_{est} is small, most of the observed values (y) dots are fairly close to the fitting line in the scatter plot, and better is the estimate based on the equation of the line.
2. When the σ_{est} is large, many of the observed values are far away from the line.
3. When the standard error is zero, then no variation exists corresponding to the computed line for predictions. The correlation between the observed and estimation is perfect.

Probabilistic Distribution of Variables, Items or Entities

Probability is the chance of observing a dependent variable value with respect to some independent variable. Suppose a Grandmaster in chess has won 22 out of 100 games, drawn 78 times, and lost none.

Then, probability P of winning P_w is 0.22, P of drawn game P_0 is 0.78 and P of losing, $P_L = 0$.

The sum of the probabilities is normalized to 1, as only one of the three possibilities exist.

Probability distribution is the distribution of P values as a function of all possible independent values, variables, situations, distances or variables.

For example, if P is given by a function $P(x)$, then P varies as x changes. Variations in $P(x)$ with x can be discrete or continuous.

The values of P are normalized such that sum of all P values is 1.

Assuming distribution is around the expected value \bar{x} , the standard normal distribution formula is

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\bar{x})^2}{2\sigma^2}} \quad (6.3)$$

Normal distribution relates to Gaussian function. Figure 6.3 shows a PDF with normal distribution around $x = \bar{x}$ standard deviation = s and variance = s^2

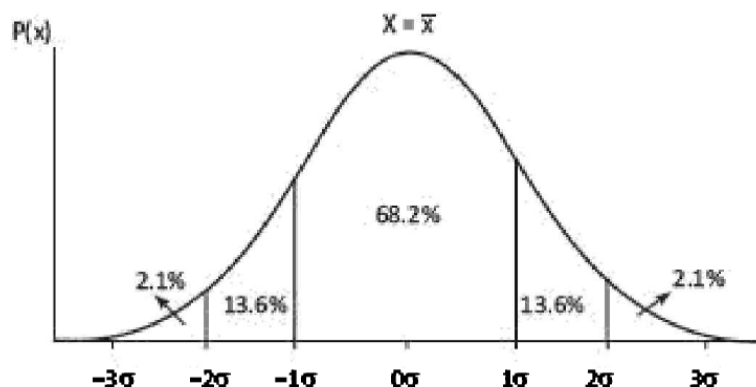


Figure 6.3 Probability distribution function as a function of x assuming normal distribution around $x = \bar{x}$, and standard deviation = s

The figure also shows the percentages of areas in five regions with respect to the total area under the curve for $P(x)$.

The variance for probability distribution represents how individual data points relate to each other within a dataset.

The variance is the average of the squared differences between each data value and the mean.

Moments (0, 1, 2 ...) refer to the expected values to the power of (0, 1, 2,) of random variable variance (Section 6.2.5.3). The variance is the second central moment of a distribution, which equals to the square of the standard deviation, and the covariance of the random variable with itself, and it is often represented by s^2 or $\text{var}(x)$.

The variance is computed as follows:

$$\sigma^2 = \frac{\sum (x_i - \bar{x})^2}{N} \quad (6.4)$$

Assume that probability distribution (PDF) is normal, called Gaussian distribution, which is like a bell-shaped curve (Figure 6.3). The PDF of the normal distribution is such that 68% of area under the PDF is within $(\bar{x} + s)$ and $(\bar{x} - s)$, 95% of area under the PDF is within $(\bar{x} + 2s)$ and $(\bar{x} - 2s)$ and 99.7% is within $(\bar{x} + 3s)$ and $(\bar{x} - 3s)$.

Standard deviation and empirical rule help in computing the population distribution over 68%, 95% and 99.7% of data under normally distributed population. This further helps in forecasting. The following example explains the meaning of population, expected values, normalized probabilities, PDF and interpretation using mean value

EXAMPLE 6.2

Assume that N students gave the examination. Let N_1 is number of students obtained grade pointer average = 1, N_2 got 2, ..., N_{10} got 10. Highest-grade pointer is 10.0. Grade pointer obtained is not a random variable. Grade pointer variation is a random variable with an expected value and standard deviation.

Expected value among the distributed x_i values, where i varies discretely from 0.0 to 10.0 will depend on the expected performance of the student. If teaching in the class is very good and students prepare for the examination very well, then expected value of GPA is 8.0 for very good performing students and standard deviation found is 1.0.

- (i) What do you mean by population? What do you mean by sample?
- (ii) What will be the normalized probabilities?
- (iii) How will you define Probability Distribution Function (PDF)?
- (iv) How will you interpret the results in terms of normal distribution?
- (v) When will you interpret the results as poor and poorer in terms of normal distribution?

SOLUTION

- (i) Population is GPA of all the students of the university who gave the examination. Population size is N . Sample means datasets used in the analysis. It can be N or less than N students and GPA of each one.
- (ii) Probability that students obtained grade pointer 1 is $\left(\frac{N_1}{N}\right)$, 2 is $\left(\frac{N_2}{N}\right)$, ... on normalization of probability. ($N = N_1 + N_2 + \dots$)
- (iii) PDF represents a curve for independent variable x between GPA = 0 and GPA = 10, such that the sum of all P values is 1, where P_i is the ratio of number of students getting GPA = i with respect to the total population N or the sample.

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\bar{x})^2}{2\sigma^2}} \quad (6.5)$$

between $x = 0$ and 10.0, where $s = 1.0$ and $\bar{x} = 8.0$.

- (iv) GPA value is 8.0 and standard deviation is 1.0, which means 68% of the students will get GPAs between 7.0 and 9.0, 95% between 6.0 and 10.0, and 99.7% between 5.0 and 10.0.
- (v) The expected value of 3.0 (less than 3.0) and standard deviation of 1.0 means poor performance of students because 68% students get between 2.0 and 4.0. The expected value of 3.0- (less than 3.0, say 2.5) and standard deviation of 1.5 means poorer performance of students because 68% students get between 1.0 and 4.0.

A probability or weight can be represented by a kernel function¹ like a Gaussian or tricube function. (Kernel in English means some thing central and key (important) part. For example, the kernel inside a walnut's shell is important because it is the edible part. Kernel in an operating system is key or central component.)

Kernel function is a function which is a central or key part of another function. For example, Gaussian kernel function is the key part of the probability distribution function [Equation (6.5)]. Figure 6.3 shows the probability normal distribution, which is a Gaussian function based on the Gaussian kernel function.

A kernel function¹, K^* defines as

$$K^*(u) = \lambda \cdot K(\lambda \cdot u), \quad (6.6a)$$

where $\lambda > 0$. Gaussian kernel function is

$$K^*(x) = \left[\frac{1}{(\sqrt{2\pi})} \right] e^{\left[-\frac{x^2}{2} \right]}, \quad (6.6b)$$

and when $u = \frac{\left\{ \frac{x - \bar{x}}{2} \right\}}{\sigma}$, the distribution function is proportional to

$$P(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{\frac{-(x - \bar{x})^2}{2\sigma^2}}.$$

$$\lambda = \left(\frac{1}{\sigma \sqrt{2}} \right) \text{ in Equation (6.3).}$$

Tricube kernel function is:

$$K^*(u) = (70/81) (1 - |u|^3)^3 \lambda \cdot K(\lambda \cdot u), \quad (6.6c)$$

where $|u| \leq 1$.

Moments

Moments (0, 1, 2, ...) refer to expected values to the powers of (0, 1, 2 ...) of random variable variance. 0th moment is 1, 1st moment = $E(x) = \bar{x}$, (expected value), 2nd moment is squared $V[(x_i - \bar{x})^2]$ = sum of product of $(x_i - \bar{x})^2$, and $P(x = x_i)$.

Here, P is the probability at $x = x_i$ when i is varying from 1 to n , for n values of random variable x . The r^{th} moment is r^{th} power of variance $V[(x_i - \bar{x})^r]$.

Moments are evaluated from the results obtained for the randomly distributed probabilistic values of the variable, such as sales. 1st moment assigns equal weight to variances of outliers and inliers, i.e., equal weight for variance of each. 2nd moment

assigns higher weight to outliers compared to inliers. 3rd moment assigns greater weight to outliers compared to inliers. Moment can be defined with respect to the origin, and in that case, x is considered 0.

Let P is along y axis and variable x on x axis. Central moment means that moments compute taking \bar{x} , equals to variable x at x axis point where the probability curve partitions equally by a vertical axis, parallel to y axis.

Unequal Variance Welch's t-test

A test in statistics is unequal-variance t test, also called Welch t -test.

(i) The test assumes that two groups of data are sampled data which consist of Gaussian distributed populations (Equation (6.3)).

(ii) The test does not assume those two populations have the same standard deviation.

Unequal variances t -test is a two-sample location test.

It tests the hypothesis that two populations have equal means. (Hypothesis means making assumption statements about certain characteristics of the population.

For example, an assumption that most students of a specific professor will excel as a programmer.

Hypothesis when tested for a decade may pass or fail depending up on whether the statistically significant results show that the students of that professor really excelled as programmers.)

Welch's t -test is an adaptation of student's t -test in statistics. The t -test is more reliable when the two samples have unequal variances and unequal sample sizes.

Analysis of Variance (ANOVA)

An ANOVA test is a method which finds whether the fitted results are significant or not. This means that the test finds out (infer) whether to reject or accept the null hypothesis. Null hypothesis is a statistical test that means the hypothesis that "no significant difference exists between the specified populations". Any observed difference is just due to sampling or experimental error.

Consider two specified populations (datasets) consisting of yearly sales data of Tata Zest and Jaguar Land Rover models. The statistical test is for proving that yearly sales of both

the models, means increments and decrements of sales are related or not. Null hypothesis starts with the assumption that no significant relation exists in the two sets of data (population).

The analysis (ANOVA) is for disproving or accepting the null hypothesis. The test also finds whether to accept another alternate hypothesis. The test finds that whether testing groups have any difference between them or not.

Analysis of variance (ANOVA) is a useful technique for comparing more than two populations, samples, observations or results of computations. It is used when multiple sample cases are involved. Variation between samples and also within sample items may exist.

For example, compare the effect of three different types of teaching methodologies on students. This may be done by comparing the test scores of the three groups of 20 students each. This technique provides inferences about whether the samples have been drawn from populations having the same mean. It is done by examining the amount of variation within each of these samples, relative to the amount of variation between the samples.

F-test F-test requires two estimates of population variance- one based on variance between the samples and the other based on variance within the samples. These two estimates are then compared for F-test:

$$F = \frac{E1(V)}{E2(V)} \quad (6.7)$$

where $E1(V)$ is an estimate of population variance between the two samples and $E2(V)$ is an estimate of population variance within the two samples. Several different F-tables exist. Each one has a different level of significance. Thus, look up the numerator degrees of freedom and the denominator degrees of freedom to find the critical value.

The value of F calculated using the above-mentioned formula is to be compared to the critical value of F for the given degrees of freedom. If the F value calculated is equal or exceeds the critical value, then significant differences between the

means of samples exist. This reveals that the samples are not drawn from the same population and thus null hypothesis is rejected.

No Relationship Case

Statistical relationship is a dependence or association between two random variables or bivariate data. Bivariate means 'two variables'. In other words, there are two types of data. Relationships between variables need to be studied and analyzed before drawing conclusions based on it. One cannot determine the right conclusion or association when no relationship between the variables exists.

Correlation

Correlation means analysis which lets us find the association or the absence of the relationship between two variables, x and y . Correlation gives the strength of the relationship between the model and the dependent variable on a convenient 0-100% scale.

R-Square R is a measure of correlation between the predicted values y and the observed values of x . R-squared (R^2) is a goodness-of-fit measure in linear regression model. It is also known as the coefficient of determination. R^2 is the square of R , the coefficient of multiple correlations, and includes additional independent (explanatory) variables in regression equation.

Interpretation of R-squared The larger the R^2 , the better the regression model fits the observations, i.e., the correlation is better. Theoretically, if a model shows 100% variance, then the fitted values are always equal to the observed values, and therefore, all the data points would fall on the fitted regression line.

Correlation differs from a regression analysis. Regression analysis predicts the value of the dependent predictor or response variable based on the known value of the independent variable, assuming a more or less mathematical relationship between two or more variables within the specified variances.

Correlation Indicators of Linear Relationships

Correlation is a statistical technique that measures and describes the 'strength' and 'direction' of the relationship between two variables. Let us explore the relations between only two variables.

Does y increase or decrease with x? For example, expenditure increases with income or does the number of patients decrease with proper medication. (Direction)

(i) Suppose y does increase with x; then, how fast? (ii) Is this relationship strong?

(iii) Can reliable predictions be made? That is, if one tells the income, can the expenditure be predicted?

Relationships and correlations enable training model on sample data using statistical or ML algorithms. Statistical correlation is measured by the coefficient of correlation. The most common correlation coefficient, called the Pearson product-moment correlation coefficient. It measures the strength of the linear association between variables.

The correlation r between the two variables x and y is:

$$r = \left[\frac{1}{(n-1)} \right] \times \sum \left\{ \left[\frac{(x_i - \bar{x})}{\sigma_x} \right] \times \left[\frac{(y_i - \bar{y})}{\sigma_y} \right] \right\}, \quad (6.8a)$$

where n is the number of observations in the sample, x_i is the x value for observation i , \bar{x} is the sample mean of x , y_i is the y value for observation i , \bar{y} is the sample mean of y , σ_x is the sample standard deviation of x , and σ_y is the sample standard deviation of y .

Summation is over all n values of i , $i = 1, 2, \dots, n$.

[r^2 is square of sample correlation coefficient between the observed outcomes and the observed predictor values, and includes intercept on y -axis in case of linear regression.]

Use of Statistical Correlation Assume one sample dataset is $\{u_1, \dots, u_n\}$

containing n values of a parameter r . The $u_{i,j}$ is i -th data point in dataset u . ($i = 1,$

$2, \dots, n$). Another sample dataset is $\{v_1, \dots, v_n\}$

containing n values of r . $v_{i,j}$ is i -th data point in dataset v . Let the correlation among two samples is being measured.

Use of Statistical Correlation Assume one sample dataset is $\{u_1, \dots, u_n\}$

containing n values of a parameter r . The $u_{i,j}$ is i -th data point in dataset u .

($i = 1, 2, \dots, n$). Another sample dataset is $\{v_1, \dots, v_n\}$

containing n values of r . $r_{v,i}$ is i -th data point in dataset v . Let the correlation among two samples is being measured.

Sample Pearson correlation metric c ; measures how well two sample datasets fit on a straight line.

$$c_r(u, v) = \frac{\sum_i (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_i (r_{u,i} - \bar{r}_u)^2 \sum_i (r_{v,i} - \bar{r}_v)^2}} \quad \dots (6.8b)$$

where the summations are over the values of parameter in the datasets.

Three other similarities based on correlation are:

- (i) Constrained Pearson correlation - It is a variation of Pearson correlation that uses midpoint instead of mean rate.
- (ii) Spearman rank correlation - It is similar to Pearson correlation, except that the ratings are ranks.
- (iii) Kendall's G correlation - It is similar to the Spearman rank correlation, but instead of using ranks themselves, only the relative ranks are used to calculate the correlation.

Numerical value of correlation coefficient ranges from + 1.0 to -1.0. It gives an indication of both the strength and direction of the relationship between variables.

In general, a correlation coefficient $r > 0$ indicates a positive relationship; $r < 0$ indicates a negative relationship; $r = 0$ indicates no relationship (or that the variables are independent of each other and not related). Here $r = + 1.0$ describes a perfect positive correlation and $r = -1.0$ describes a perfect negative correlation.

The closer the coefficients are to + 1.0 and -1.0, the greater is the strength of the relationship between the variables.

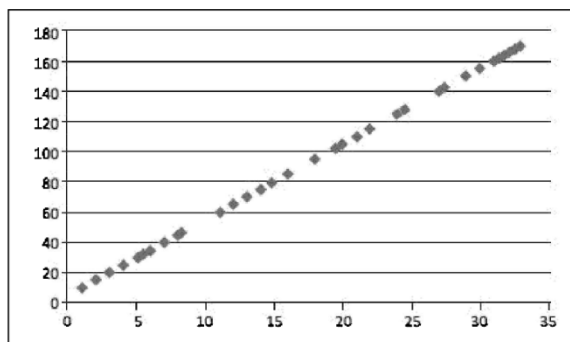
Table 6.1 gives rough guidelines on the strength of the relationship (though many experts would somewhat disagree on the choice of boundaries).

Table 6.1 The strength of the relationship as a function of r

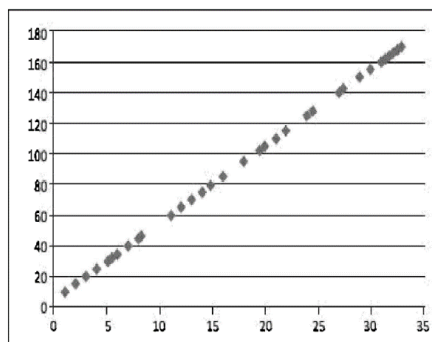
Value of r	Strength of relationship
-1.0 to -0.5 or 1.0 to 0.5	Strong
-0.5 to -0.3 or 0.3 to 0.5	Moderate
-0.3 to -0.1 or 0.1 to 0.3	Weak
-0.1 to 0.1	None or very weak

Correlation is only appropriate for examining the relationship between meaningful quantifiable data (such as, temperature, marks, score) rather than categorical data, such as gender, color etc. Figure 6.4 shows perfect and imperfect, linear positive and negative relationships. And the strength and

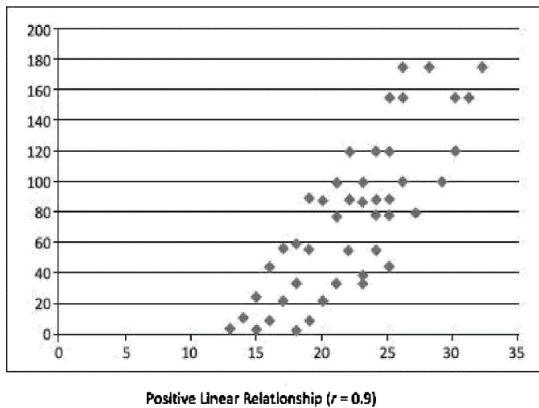
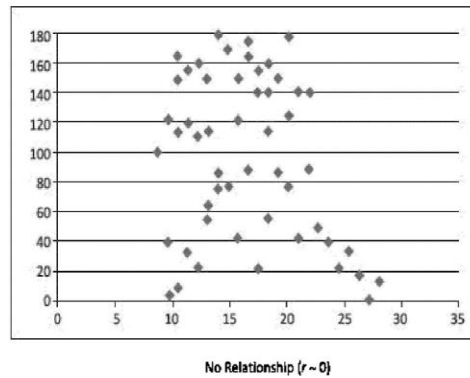
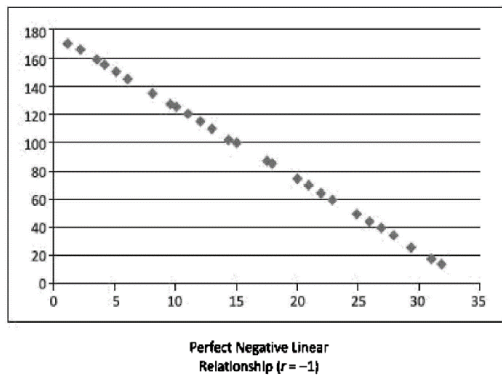
Direction of the relationship between variables.



Perfect Positive Linear
Relationship ($r = 1$)



Perfect Positive Linear
Relationship ($r = 1$)



REGRESSION ANALYSIS

- Correlation and regression are two analyses based on multivariate distribution. A multivariate distribution means a distribution in multiple variables.
- Suppose a company wishes to plan the manufacturing of Jaguar cars for coming years. The company looks at sales data regressively, i.e., data of previous years' sales. Regressive analysis means estimating relationships between variables
- Regression analysis is a set of statistical steps, which estimate the relationships among variables. Regression analysis may require many techniques for modeling and performing the analysis using multiple variables. The aim of the analysis is to find the relationships between a dependent variable and one or more independent, outcome, predictor or response variables. Regression analysis facilitates prediction of future values of dependent variables.
- Regression analysis is a set of statistical steps, which estimate the relationships among variables. Regression analysis may require many techniques for modeling and performing the analysis using multiple variables. The aim of the analysis is to find the relationships between a dependent variable and one or more

independent, outcome, predictor or response variables. Regression analysis facilitates prediction of future values of dependent variables.

- It helps to find how a dependent variable changes when variation is in an independent variable among a set of them, while the remaining independent variables in the set are kept fixed.
- Non-linear regression equation is as follows:

$$y = a_0 + a_1x + a_2x^2 + a_3x^3, \quad (6.9)$$

where number of terms on the right-hand side are 3 or 4. Linear regression means only the first two terms are considered. The following subsections describe regression analysis in detail.

Simple Linear Regression

- Linear regression is a simple and widely used algorithm. It is a supervised ML algorithm for predictive analysis. It models a relationship between the independent predictor or explanatory, and the dependent outcome or variable, y using a linearity equation.

$$y = f(a_0, a_1) = a_0 + a_1x, \quad (6.10)$$

where a_0 is a constant and a_1 is the linearity coefficient.

Simple linear regression is performed when the requirement is prediction of values of one variable, with given values of another variable. The following example explains the meaning of linear regression.

EXAMPLE 6.3

How can a university student's GPA be predicted from his/her high school percentage (HSP) of marks?

SOLUTION

Consider a sample of ten students for whom their GPAs and high school scores, HSPs, are known. Assume linear regression. Then,

$$\text{GPA} = b_1 \cdot \text{HSP} + A \quad \dots (6.11)$$

Figure 6.5 shows a simple linear regression plot for the relationship between the college GPA and the percentage of high school marks. Plot the values on a graph, with high school scores in percentage on the x axis and GPA on the y axis.

Figure 6.5 shows a simple linear regression plot for the relationship between the college GPA and the percentage of high school marks. Plot the values on a graph, with high school scores in percentage on the x axis and GPA on the y axis.

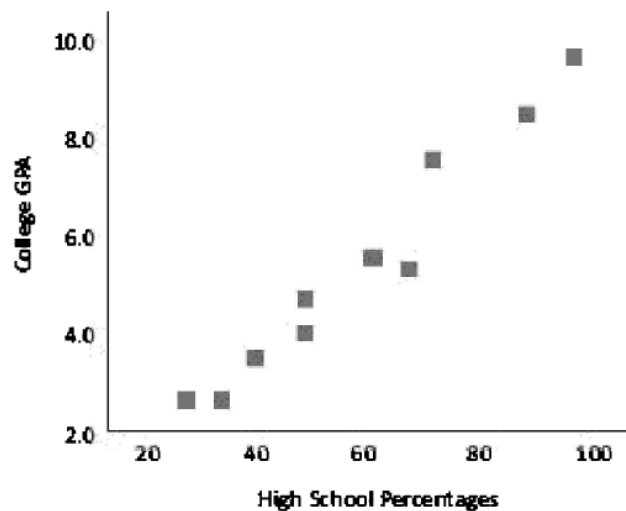


Figure 6.5 Linear regression relationship between college GPA and percentage of high school marks

Whenever a perfect linear relationship between GPA and high school score exists, all 10 points on the graph would fit on a straight line. However, this is never the case. Whenever an imperfect linear relationship exists between these two variables, a cluster of points on the graph, which slope upward, may be obtained. In other words, students who got more marks in high school should get more GPA in college as well.

One variable, denoted by x , is regarded as the predictor, explanatory or independent variable. The other variable, denoted by y , is regarded as the response, outcome or dependent variable.

Figure 6.6 shows a simple linear regression with two regression lines with different regression equations. Looking at the scatter plot, two lines can fit best to summarize the relation between GPA and high school percentage.

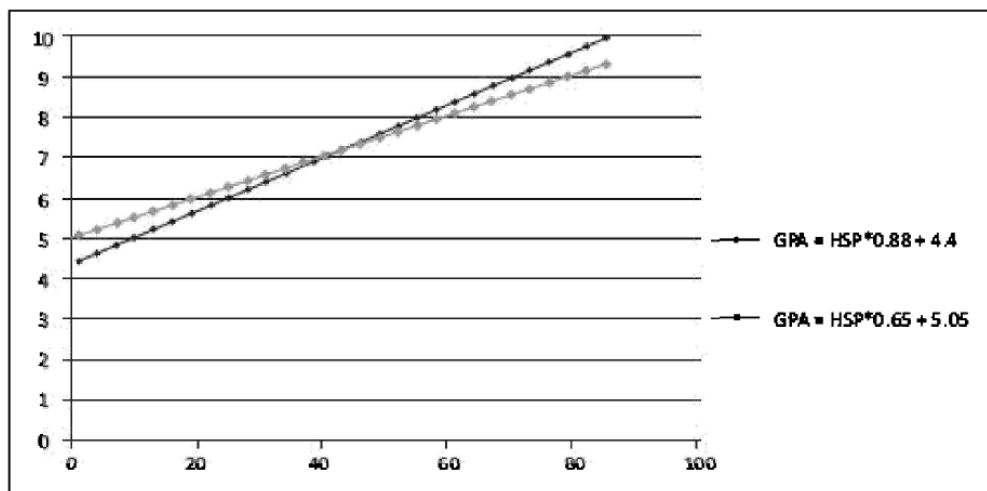


Figure 6.6 Linear regression relationship with two regression lines with different coefficient in regression equation

Following notations can be used for examining which of the two lines is a better fit:

1. Y_i denotes the observed response for experimental unit i
2. x_i denotes the predictor value for experimental unit i
3. \hat{Y}_i is the predicted response (or fitted value) for experimental unit i

Then, the equation for the best fitting line using a sum of the error estimating function is:

$$\hat{y}_i' = a_0' + a_1' x_i \quad (6.12)$$

where a_0' and a_1' are the coefficients in Equation (6.10). Use of the above equation to predict the actual response Y_i , leads to a prediction error (or residual error) of size:

$$e_i = y_i - \hat{y}_i \quad (6.13)$$

FINDING SIMILAR ITEMS, SIMILARITY OF SETS AND COLLABORATIVE FILTERING

- Similar item search refers to a data mining method which helps in discovering items which have similarities in datasets. (Data mining means discovering previously unknown interesting patterns and knowledge from apparently unstructured data. The process of data mining uses the ML algorithms. Data mining enables analysis, categorization and summarization of data and relationships among data.)

Finding Similar Items

An analysis requires many times to find similar items. For example, finding similar excellent performance of students in Python programming, similar showrooms of a specific car model which show high sales per month, recommending books on similar topic such as in Internet of Things by Raj Kamal from McGraw-Hill Higher Education, etc.

Application of Near Neighbor Search

- Similar items can be found using Nearest Neighbor Search (NNS). The search finds that a point in a given set is most similar (closest) to a given point. A dissimilarity function having larger value means less similar. The dissimilarity function is used to find similar items
- NNS algorithm is as follows: Consider set S having points in a space M . Consider a queried point $q \in M$, which means q is member of M . k -NNS algorithm finds the k -closest (1-NN) points to q in S .

Three problems with the Pearson similarities :

- 1. Do not consider the number of items in which two users' preferences overlap. (e.g., 2 overlap items \Rightarrow 1, more items may not be better.)
- 2. If two users overlap on only one item, no correlation can be computed.
- 3. The correlation is undefined if series of preference values are identical.

FREQUENT ITEMSETS AND ASSOCIATION RULE MINING

Frequent Itemset Mining

- Extracting knowledge from a dataset is the main goal of data analytics and data mining. Data mining mainly deals with the type of patterns that can be mined. A method of mining is Frequent Patterns (FPs) mining method. Frequent patterns occur frequently in transactional data.
- Frequent item set refers to a set of items that frequently appear together, for example, Python and Big Data Analytics. Students of computer science frequently choose these subjects for in-depth studies. Frequent item set refers to a frequent itemset, which is a subset of items that appears frequently in a dataset.
- Frequent Itemset Mining (FIM) refers to a data mining method which helps in discovering the itemsets that appear frequently in a dataset. For example, finding a set of students who frequently show poor performance in semester examinations. Frequent subsequence is a sequence of patterns that occurs frequently. For example, purchasing a football follows purchasing of sports kit. Frequent substructure refers to different structural forms, such as graphs, trees or lattices, which may be combined with itemsets or subsequences.
- FIM is one of the popular techniques to extract knowledge from data. The technique has been an essential part of data analysis and data mining. The extraction is based on frequently occurring events. An algorithm specifies a given minimum frequency threshold for considering an itemset as frequent. The extraction generally depends on the specified threshold.
- Frequent Itemset Mining (FIM) refers to a data mining method which helps in discovering the itemsets that appear frequently in a dataset. For example, finding a set of students who frequently show poor performance in semester examinations. Frequent subsequence is a sequence of patterns that occurs frequently. For example, purchasing a football follows purchasing of sports kit. Frequent substructure refers to different structural forms, such as graphs, trees or lattices, which may be combined with itemsets or subsequences.
- FIM is one of the popular techniques to extract knowledge from data. The technique has been an essential part of data analysis and data mining. The

extraction is based on frequently occurring events. An algorithm specifies a given minimum frequency threshold for considering an itemset as frequent. The extraction generally depends on the specified threshold.

Association Rule- Overview

- An important method of data mining is association rule mining or association analysis. The method has been widely used in many application areas for discovering interesting relationships which are present in large datasets. The objective is to find uncovered relationships using some strong rules. The rules are termed as association rules for frequent itemset. Mahout includes a
- 'parallel frequent pattern growth' algorithm. The method analyzes the items in a group and then identifies which items typically appear together (association) (Section 6.8). A formal statement of the association rule problem is:

Let $I = \{I_1, I_2, \dots, I_d\}$ be a set of d distinct attributes, also called literals. Let $T =$

- $\{t_1, t_2, \dots, t_n\}$ be set of n transactions and contain a set of items such that $T \subseteq I$. An association rule is an implication of the form, $X \subseteq Y$, where X, Y belong to sets of items called itemset ($X, Y \subseteq I$), and X and Y are disjoint itemset

($X \cap Y = \emptyset$). Here, X is called antecedent, and Y consequent.

Explanation:

- \subseteq means 'subset of', \subset means 'proper (strict) subset of', \cap means intersection and \emptyset means disjoint, no commonality in members.
- 2. Consider an If() then () form of a rule. The If part of the rule (A) is known as antecedent and the THEN part of the rule (B) is known as consequent. The condition is antecedent. Result is consequent.

Applications of Association Rules

- FIM is a popular technique for market basket analysis.
- Market Basket Model
- Market basket analysis is a tool for knowledge discovery about co-occurrence of items. A co-occurrence means two or more things occur together. It can also be defined as a data mining technique to derive the strength of association

between pairs of product items. If people tend to buy two products (say A and B) together, then the buyer of product A is a potential customer for an advertisement of product B.

The concept is similar to the real market basket where we select an item (product) and put it in a basket (itemset). The basket symbolizes the transactions. The number of baskets is very high as compared to the items in a basket. A set of items that is present in many baskets is termed as a frequent itemset. Frequency is the proportion of baskets that contain the items of interest.

Market basket analysis can be applied to many areas. The following example explains the market basket model using application examples

Example :- Suggest application examples of the market basket model

SOLUTION

Application 1

1. Items = Products

Baskets = Sets of products a customer purchases at one time from a store.

Example:

Run sales on flowers; raise price of chocolates.

The knowledge is useful when many buy chocolates and flowers together of an application: Given that, many people buy chocolates and flowers together.

Application 2:

2. Items = Words

Baskets = Web pages

Unusual words appearing together in a large number of documents, for example, 'research' and 'plastic' may provide interesting information.

Finding Association

Association rules intend to tell how items of a dataset are associated with each other.

The concept of association rules was introduced in 1993 for discovering relations between items in sales data of a large retailing company.

The following examples give rules between items found associated in the sales data of a retailer.

Suggest association rules between items found in the sales data of a retailer, and rules for course choice for a computer science student in college

SOLUTION

1. {Bread} ~ {Butter}

The rule suggests a relationship between the sales of bread and butter. A customer who buys bread also buys butter.

2. {Chocolates} ~ {a Gift Box}

The rule suggests a that relationship between the sales of chocolates and empty gift boxes exists. A customer who buys chocolates also buys a gift box.

3. {Java programming} + {advanced web technology} and {Python programming} ~ {Big Data Analytics}

The rules suggest relationships between Java and advanced web technology, and Python programming and data analytics. Students who opt for Java programming also want to learn advanced web technology, and those who opt for Python programming also opt for Big Data Analytics.

Finding Similarity

Let A and B be two itemset. Jaccard similarity index of two itemsets is measured in terms of set theory using the following equation:

$$\text{Jaccard itemsets similarity index} = \frac{|A \cap B|}{|A \cup B|} \times 100\%.$$

Explanation: n means intersection, number of those elements or items which are the same in set A and B.

U means union, number of elements or items present in union of A and B

Text Mining

Text mining is the art and science of discovering knowledge, insights and patterns from an organized collection of textual databases. Textual mining can help with frequency analysis of important terms, and their semantic relationships.

Text is an important part of the growing data in the world. Social media technologies have enabled users to become producers of text and images and other kinds of information. Text mining can be applied to large-scale social media data for gathering preferences, and measuring emotional sentiments. It can also be applied to societal, organizational and individual scales.

Text Mining Applications

Text mining is a useful tool in the hands of chief knowledge officers to extract knowledge relevant to an organization. Text mining can be used across industry sectors and application areas, including decision support, sentiment analysis, fraud detection, survey analysis, and many more.

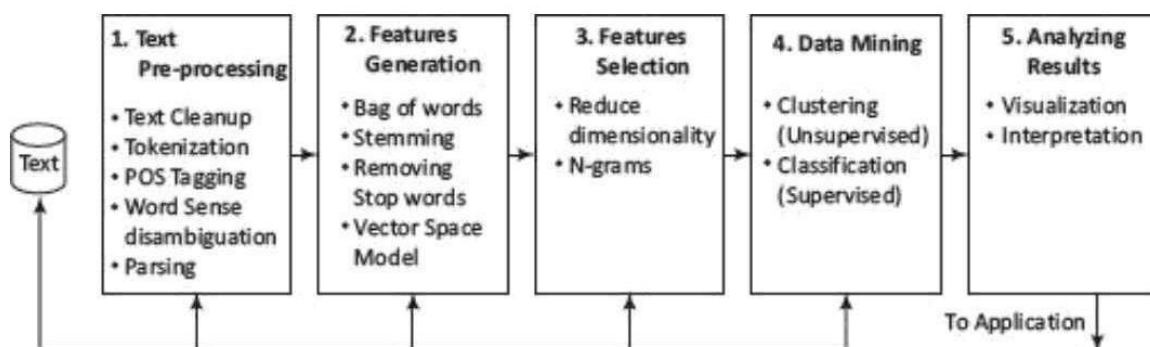
1. *Marketing*: The voice of the customer can be captured in its native and raw format and then analyzed for customer preferences and complaints.
 1. Social personas are a clustering technique to develop customer segments of interest. Consumer input from social media sources, such as reviews, blogs, and tweets, contain numerous leading indicators that can be used towards anticipating and predicting consumer behavior.
 2. A 'listening platform' is text mining application, that in real time, gathers social media, blogs, and other textual feedback, and filters out the chatter to extract true consumer sentiment. The insights can lead to more effective product marketing and better customer service.
2. The customer call center conversations and records can be analyzed for patterns of customer complaints. Decision trees can organize this data to create decision choices that could help with product management activities and to become proactive in avoiding those complaints.
3. *Business operations*: Many aspects of business functioning can be accurately gauged from analyzing text.
 1. Social network analysis and text mining can be applied to emails, blogs, social media and other data to measure the emotional states and the

mood of employee populations. Sentiment analysis can reveal early signs of employee dissatisfaction which can then be proactively managed.

2. Studying people as emotional investors and using text analysis of the social Internet to measure mass psychology can help in obtaining superior investment returns.
3. *Legal*: In legal applications, lawyers and paralegals can more easily search case histories and laws for relevant documents in a particular case to improve their chances of winning.
 1. Text mining is also embedded in e-discovery platforms that help in minimizing risk in the process of sharing legally mandated documents.
 2. Case histories, testimonies, and client meeting notes can reveal additional information, such as morbidities in a healthcare situation that can help better predict high-cost injuries and prevent costs.
4. *Governance and Politics*: Governments can be returned based on a tweet originating from a self-immolating fruit-vendor in Tunisia
 1. Social network analysis and text mining large-scale social media data can be used for measuring the emotional states and the mood of constituent populations. Micro-targeting constituents with specific messages gleaned from social media analysis can be a more efficient use of resource when fighting democratic elections.
 2. In geopolitical security, internet chatter can be processed for real-time information and to connect the dots on any emerging threats.
 3. In academia, research streams could be meta-analyzed for underlying research trends.

Text Mining Process

Text Mining is a rapidly evolving area of research. As the amount of social media and other text data grows, there is need for efficient abstraction and categorization of meaningful information from the text.



The five phases for processing text are as follows:

Phase 1: Text pre-processing enables Syntactic/Semantic text-analysis and does the followings:

1. Text *cleanup* is a process of removing unnecessary or unwanted information. Text cleanup converts the raw data by filling up the missing values, identifies and removes outliers, and resolves the inconsistencies. For example, removing comments, removing or escaping "%20" from URL for the web pages or cleanup the typing error, such as teh (the), do n't (do not) [%20 specifies space in a URL].
2. *Tokenization* is a process of splitting the cleanup text into tokens (words) using white spaces and punctuation marks as delimiters.
3. *Part of Speech (POS) tagging* is a method that attempts labeling of each token (word) with an appropriate POS. Tagging helps in recognizing names of people, places, organizations and titles. English language set includes the noun, verb, adverb, adjective, prepositions and conjunctions. Part of Speech encoded in the annotation system of the Penn Treebank Project has 36 POS tags.⁴
4. *Word sense disambiguation* is a method, which identifies the sense of a word used in a sentence; that gives meaning in case the word has multiple meanings. The methods, which resolve the ambiguity of words can be context or proximity based. Some examples of such words are bear, bank, cell and bass.
5. *Parsing* is a method, which generates parse-tree for each sentence. Parsing attempts and infers the precise grammatical relationships between different words in a given sentence.

Phase 2: Features Generation is a process which first defines features (variables, predictors). Some of the ways of feature generations are:

1. *Bag of words-Order of words* is not that important for certain applications. Text document is represented by the words it contains (and their occurrences). Document classification commonly use the bag-of-words model. The pre-processing of a document first provides a document with a bag of words. Document classification methods then use the occurrence (frequency) of each word as a feature for training a classifier. Algorithms do not directly apply on the bag of words, but use the frequencies.
2. *Stemming*-identifies a word by its root.
 - (i) Normalizes or unifies variations of the same concept, such as *speak* for three variations, i.e., speaking, speaks, speakers denoted by [speaking, speaks, speaker- + speak]
 - (ii) Removes plurals, normalizes verb tenses and remove affixes. Stemming reduces the word to its most basic element. For example, impurification --+

pure.

3. *Removing stop words* from the feature space-they are the common words, unlikely to help text mining. The search program tries to ignore stop words. For example, ignores *a, at, for, it, in* and *are*.
4. *Vector Space Model (VSM)*-is an algebraic model for representing text documents as vector of identifiers, word frequencies or terms in the document index. VSM uses the method of term frequency-inverse document frequency (TF-IDF) and evaluates how important is a

word in a document.

When used in document classification, VSM also refers to the bag-of-words model. This bag of words is required to be converted into a term-vector in VSM. The term vector provides the numeric values corresponding to each term appearing in a document. The term vector is very helpful in feature generation and selection.

Term frequency and inverse document frequency (IDF) are important metrics in text analysis. TF-IDF weighting is most common Instead of the simple TF, IDF is used to weight the importance of word in the document.

Phase 3: Features Selection is the process that selects a subset of features by rejecting irrelevant and/or redundant features (variables, predictors or dimension) according to defined criteria. Feature selection process does the following:

1. *Dimensionality reduction*-Feature selection is one of the methods of division and therefore, dimension reduction. The basic objective is to eliminate irrelevant and redundant data. Redundant features are those, which provide n extra information. Irrelevant features provide no useful or relevant information in a context.

Principal Component Analysis (PCA) and Li r D scriminate Analysis (LDA) are dimension reduction methods. Discrimination ability of a feature measures relevancy of features. Correlation helps in finding the redundancy of the feature. Two features are redundant to each other if their values correlate with each other.

2. *N-gram evaluation*-finding the number of consecutive words of interest and extract them. For example, 2-gram is a two words sequence, ["tasty food", "Good one"]. 3-gram is a three words sequence, ["Crime Investigation Department"].
3. *Noise detection and evaluation of outliers* methods do the identification of unusual or suspicious items, vents or observations fr the data set. This step helps in cleaning the data.

The feature selection algorithm reduces dimensionality that not only improves the performance of learning algorithm but lso reduces the storage requirement for a dataset. The process enhances data understanding and its visualization.

Phase 4: Data mining techniques enable insights about the structured database that resulted from the previous phases. Examples of techniques are:

1. Unsupervised learning (for example, clustering)

(i) The class labels (categories) of training data are unknown

(ii) Establish the existence of groups or clusters in the data

Good clustering methods use high intra-cluster similarity and low inter-cluster similarity.

Examples of uses - biogs,
pattern and trends.

2. Supervised learning (for example, classification)

(i) The training data is labeled indicating the class

(ii) New data is classified based on the training set

Classification is correct when the known label of test sample is identical with the resulting class computed from the classification model

Examples of uses are *news filtering application*, where it is required to automatically assign incoming documents to pre-defined categories; *email spam filtering*, where it is identified whether incoming email messages are spam or not.

Example of text classification methods are *Naive Bayes Classifier* and *SVMs*.

3. Identifying evolutionary patterns in temporal text streams-the method is useful in a wide range of applications, such as summarizing of events in news articles and extracting the research trends in the scientific literature.

Phase 5: Analysing results

(i) Evaluate the outcome of the complete process.

(ii) Interpretation of Result- If acceptable then results obtained can be used as an input for next set of sequences. Else, the result can be discarded, and try to understand what and why the process failed.

(iii) Visualization - Prepare visuals from data, and d a prototype.

(iv) Use the results for further improvement in ac ies at the enterprise, industry or institution.

Text Mining Challenges

The challenges in the area of text mining can be classified on the basis of documents area- characteristics. Some of the classifications are as fo ows:

1. NLP issues:

(i) POS Tagging

(ii) Ambiguity

(iii)Tokenization

(iv)Parsing

(v)Stemming

(vi)Synonymy and polysemy

2. Mining techniques:

(i) Identification of the suitable algorithm(s)

(ii)Massive amount of data and annotated corpora

(iii)Concepts and semantic relations extraction

(iv) When no training data is available

3. Variety of data:

(i) Different data sources require different approaches and different areas of expertise

(ii)Unstructured and language independency

4. Information visualization

5. Efficiency when processing real-time text stream

6. Scalability

1.2 Term Document Matrix

This is the heart of the structuring process. Free flowing text can be transformed into numeric data in a TDM, which can then be mined using regular data mining techniques.

1. There are several efficient techniques for identifying key terms from a text. There are less efficient techniques available for creating topics out of them. For the purpose of this discussion, one could call key words, phrases or topics as a term of interest. This approach measures the frequencies of selected important terms occurring in each document. This creates a $t \times d$ Term-by-Document Matrix (TDM) where t is the number of terms and d is the number of documents (Table 11.1). Creating a TDM requires making choices of which terms

to include. The terms chosen should reflect the stated purpose of the text mining exercise. The list of terms should be as extensive as needed, but should not include unnecessary stuff that will serve to confuse the analysis, or slow the computation.

	Term-Document Matrix				
Document/ Terms	Investment	Profit	Happy	Success	...
Doc 1	10	4	3	4	
Doc 2	7	2	2		
Doc 3			2	6	
Doc 4	1	5	3		
Doc 5		6		2	
Doc 6	4		2		
...					

Here are some considerations in creating a TDM.

1. A large collection of documents mapped to a large bag of words will likely lead to a very sparse matrix if they have few common words. Reducing dimensionality of data will help improve the speed of analysis and meaningfulness of the results. Synonyms, or terms with similar meaning, should be combined and should be counted together, as a common term. This would help reduce the number of distinct terms or 'tokens'.
2. Data should be cleaned for spelling errors. Common spelling errors should be ignored and the terms should be combined. Uppercase- lowercase terms should also be combined.
3. When many variants of the same term are used, just the stem of the word would be used to reduce the number of terms. For instance, terms like customer order, ordering, order data, should be combined into a single token word, called 'Order'.
4. On the other side, homonyms (terms with same spelling but different meanings) should be counted separately. This would enhance the quality of analysis. For example, the term order can mean a customer order, or the ranking of certain choices. These two should be treated separately. "The boss ordered that the customer orders data analysis be presented in chronological order". This statement shows three different meanings for the word 'order'. Thus, there will be a need for a manual review of the TD matrix.
5. Terms with very few occurrences in very few documents should be eliminated from the matrix. This would help increase the density of the matrix and the quality of analysis.

6. The measures in each cell of the matrix could be one of several possibilities. It could be a simple count of the number of occurrences of each term in a document. It could also be the log of that number. It could be the fraction number computed by dividing the frequency count by the total number of words in the document. Or there may be binary values in the matrix to represent whether a term is mentioned or not. The choice of value in the cells will depend upon the purpose of the text analysis.

At the end of this analysis and cleansing, a well-formed, densely populated, rectangular, TDM will be ready for analysis. The TDM could be mined using all the available data mining techniques.

1.3 Mining the TDM

The TDM can be mined to extract patterns/knowledge. A variety of techniques could be applied to the TDM to extract new knowledge.

Predictors of desirable terms could be discovered through predictive techniques, such as regression analysis. Suppose the word profit is a desirable word in a document. The number of occurrences of the word profit in a document could be regressed against many other terms in the TDM. The relative strengths of the coefficients of various predictor variables would show the relative impact of those terms on creating a profit discussion.

Predicting the chances of a document being liked is another form of analysis. For example, important speeches made by the CEO or the CFO to investors could be evaluated for quality. If the classification of those documents (such as good or poor speeches) was available, then the terms of TDM could be used to predict the speech class. A decision tree could be constructed that makes a simple tree with a few decision points that predicts the success of a speech 80 percent of the time. This tree could be trained with more data to become better over time.

Clustering techniques can help categorize documents by common profile. For example, documents containing the words investment and it more often could be bundled together. Similarly, documents containing the words, customer orders and marketing, more often could be bundled together. Thus, a few strongly demarcated bundles could capture the essence of the entire TDM. These bundles could thus help with further processing, such as handing over select documents to others for legal discovery.

Association rule analysis could show relationships of coexistence. Thus, one could say that the words, tasty and sweet, occur together often (say 5 percent of the time); and further, when these two words are present, 80 percent of the time, the word happy, is also present in the document.

1.4 Comparing Text Mining and Data Mining

Text Mining is a form of data mining. There are many common elements between Text and Data Mining. However, there are some key differences (Table 1.2). The key difference is that text mining requires conversion of text data into frequency data, before data mining techniques can be applied.

WEB MINING, WEB CONTENT AND WEB USAGE ANALYTICS

Web is a collection of interrelated files at web servers.

Web data refers to:

- (i) web content-text, image and records,
- (ii) web structure-hyperlinks and tags, and
- (iii) web usage-http logs and application server logs.

Features of web data are:

1. Volume of information and its ready availability
2. Heterogeneity
3. Variety and diversity (Information on almost every topic is available using different forms, such as text, structured tables and lists, images, audio and video.)

4. Mostly semi-structured due to the nested structure of HTMLcode
5. Hyperlinks among pages within a website, and across different websites
6. Redundant or similar information may be present in several pages
7. Mostly, the web page has multiple sections (divisions), such as main contents of the page, advertisements, navigation panels, common menu for all the pages of a website and copyright notices
8. A web form or HTMLform on a web page enables a user to enter data that is sent to a server for processing
9. Website contents are dynamic in nature where information on the web pages constantly changes, and fast information growth takes place such as conversations between users, social media, etc.

The following subsections describe web data mining and analysis methods:

Web Mining

- Data Mining is a process of discovering patterns in large datasets to gain knowledge. The process can be shown as [Raw Data - Patterns - Knowledge]. Web data mining is the mining of web data. Web mining methods are in multidisciplinary domains:
 - (i) data mining, ML, natural language,
 - (ii) processing, statistics, databases, information retrieval, and
 - (iii) multimedia and visualization.

Web consists of rich features and patterns. A challenging task is retrieving interesting content and discovering knowledge from web data. Web offers several opportunities and challenges to data mining.

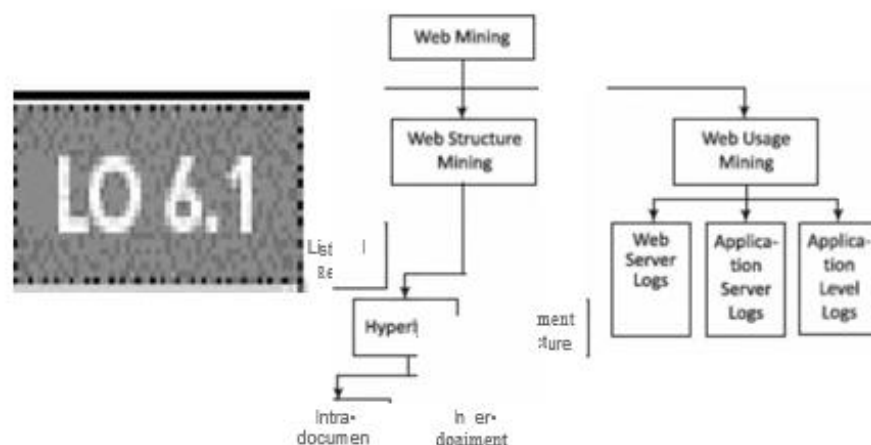
Definition of Web Mining

Web mining refers to the use of techniques and algorithms that extract knowledge from the web data available in the form of web documents and services. Web mining applications are as follows:

- (i) Extracting the fragment from a web document that represents the full web document
- (ii) Identifying interesting graph patterns or pre-processing the whole web graph to come up with metrics, such as PageRank
- (iii) User identification, session creation, malicious activity detection and filtering, and extracting usage path patterns

Web Mining Taxonomy

Web mining can broadly be classified into three categories, based on the types of web data to be mined. Three ways are web content mining, web structure mining and web usage mining. Figure 9.6 shows the taxonomy of web mining.



Web mining taxonomy

Web content mining is the process of extracting useful information from the contents of web documents. The content may consist of text, images, audio, video or structured records, such as lists and tables.

Web structure mining is the process of discovering structure information from the web. Based on the kind of structure-information present in the web resources, web structure mining can be divided into:

1. Hyperlinks: the structure that connects a location at a web page to a different location, either within the same web page (intra document hyperlink) or on a different web page (inter-document hyperlink)

2. Document Structure: The structure of a typical web graph consists of web pages as nodes, and hyper links as edges connecting the related pages.

Web Content Mining

Web Content Mining is the process of information or resource discovery from the content of web documents across the World Wide Web. Web content mining can be (i) direct mining of the contents of documents or (ii) mining through search engines. They search fast compared to direct method.

Web content mining relates to both, data mining as well as text mining. Following are the reasons:

- (i) The content from web is similar to the contents obtained from database, file system or through any other mean. Thus, available data mining techniques can be applied to the web.
- (ii) Content mining relates to text mining because much of the web content comprises texts.
- (iii) Web data are mainly semi-structured and/or unstructured, while data mining is structured and the text is unstructured.

Applications

Following are the applications of content mining from web documents:

1. Classifying the web documents into categories
2. Identifying topics of web documents
3. Finding similar web pages across the different web servers
4. Applications related to relevance.

PAGE RANK, STRUCTURE OF WEB AND ANALYZING A WEB GRAPH

Page Rank Definition

The in-degree (visibility) of a link is the measure of number of in-links from other links. The out-degree (luminosity) of a link is number of other links to which that link points.

PageRank definition according to earlier approaches

Assume a web structure of hyperlinks. Each hyperlink in-links to a number of hyperlinks and out-links to a number of pages. A page commanding higher authority (rank) has greater number of in-degrees than out-degrees. Therefore, one measure of a page authority can be in-degrees with respect to out-degrees.

PageRank refers to the authority of the page measured in terms of number of times a link is sought after.

PageRank definition according to the new approach

Earlier approach of page ranking based on in-links and out-links does not capture the relative authority (importance) of the parents. Page and co-authors (1998) defined a page ranking method,⁵ which considers the entire web in place of local neighbourhood of the pages and considers the relative authority of the parent links (over children).

Web Structure

Web structure models as directed-graphs network-organization. Vertex of the directed graph models an anchor. Let n = number of hyperlinks at the page U . Assume u is a vector with elements u_1, u_2, \dots, u_n . Each page $Pg(u)$ has anchors, called hyperlinks. Page $Pg(v)$ consists of text document with m number of hyper links. v is a vector with elements v_i, v_z, \dots, v_m . The m is number of hyper links at $Pg(v)$. A vertex u directs to another Page V . A page $Pg(v)$ may have number of hyperlinks directed by out-edges to other page $Pg(w)$.

Consider the following hypotheses:

1. Text at the hyperlink represents the property of a vertex u that describes the destination V of the out-going edge.
2. A hyperlink in-between the pages represents the conferring of the authority.

Pages U and U' hyperlinks u and u' out-linking to Page V . Let Page U has three hyperlinks parenting three Pages, V one, W two, X two, U' one, and Y two, respectively.

Web Communities

Web communities are web sites or collections of websites, which limit the contents view and links to members. Examples of web communities are social networks, such as LinkedIn, SlideShare, Twitter and Facebook.

The communities consist of sites for do-it-yourself sites, social networks, blogs or bulletin boards. The issues are privacy and reliability of information.

Metric for analysis of web-community sites are web graph parameters, such as triangle count, clustering coefficient and K-neighbourhood.

K-neighbourhood analysis means the number of 1st neighbour nodes, 2nd neighbour nodes, and so on ($K = 1, 2, 3, 4$ and so on).

K-core analysis means the number of cores within a marked area. A core may consist of a triangle of connected vertices. A core may consist of a rectangle with interconnected edges and diagonals. A core may also be a group of cores.

Spark Graphx (Section 8.5) described functions for degree centralities, degree distribution, separation of degree, betweenness centralities, closeness centralities, neighbourhoods, strongly connected components, triangle counts, PageRank, shortest path, Breadth First Search (BFS), minimum spanning tree (forest), spectral clustering and cluster coefficient.

Limitations of Link, Rank and Web Graph Analysis

Following are the limitations of link and web graph analysis:

1. Search engines rely on metatags or metadata of the documents. That enhances the rank if metadata has biased information.

2. Search engines themselves may introduce bias while ranking the pages of clients higher as the pages of advertising companies

may provide higher searches and hence lead to biased ranks.

3. A top authority may be a hub of pages on a different topic resulting in increased rank of the authority page.

4. Topic drift and content evolution can affect the rank. Off-topic pages may return the authorities.
5. Mutually reinforcing affiliates or affiliated pages/sites can enhance each other's rank and authorities.
6. The ranks may be unstable as adding additional nodes may have greater influence in rank changes.

SOCIAL NETWORKS AS GRAPHS AND SOCIAL NETWORK ANALYTICS

- A social network is a social structure made of individuals (or organizations) called "nodes," which are tied (connected) by one or more specific types of interdependency, such as friendship, kinship, financial exchange, dislike or relationships of beliefs, knowledge or prestige. (Wikipedia)
- Social networking is the grouping of individuals into specific groups, like small rural communities or some other neighbourhoods based on a requirement. The following subsections describe social networks as graph, uses, characteristics and metrics.

Representation of social networks as graphs, methods of social network analysis, finding the clustering in social network graphs, evaluating the Sim Rank counting triangles (cliques) and discovering the communities

Social Network as Graphs

Social network as graphs provide a number of metrics for analysis. The metrics enable the application of the graphs in a number of fields. Network topological analysis tools compute the degree, closeness, betweenness, egonet, K-neighbourhood, top-K shortest paths, PageRank, clustering, SimRank, connected components, K-cores, triangle count,

graph matches and clustering coefficient. Bipartite weighted graph matching does collaborative filtering.

Apache Spark Graphx and IBM System G Graph Analytics tools are the tools for social network analysis.

Centralities, Ranking and Anomaly Detection

Important metrics are degree (centrality), closeness (centrality), betweenness (centrality) and eigenvector (centrality). Eigenvector consists of elements such as status, rank and other properties. Social graph-network analytics discovers the degree of interactions, closeness, betweenness, ranks, probabilities, beliefs and potentials.

Social network analysis of closeness and sparseness enables detection of abnormality in persons. Abnormality is found from properties of vertices and edges in network graph. Analysis enables summarization and find attributes for anomaly.

Graph Network Topological Analysis using Centralities and PageRank

Social graph network can be topologically analyzed. The centralities (degree, closeness, effective closeness and betweenness) and

PageRank (vertex Rank similar to PageRank in web graph network) are the parameters analyzed.

Degree

Degree of a graph vertex means the total number of edges linked to that. In-degree of a vertex means the number of in-edges from the other vertices. Out-degree of a vertex means the number of out-edges to other vertices to which that vertex directs. Degree distribution function means the distribution function for the degrees of vertices (Section 6.2.5 described the common distribution functions).

Closeness

Graph vertex closeness $C_c(v)$ is a way of defining the centrality of a vertex in reference to other vertices. Sum is the overall vertices connected to other vertices u . The u is a subset of vertices in set V .

The centrality (closeness index), c is function of distances of vertices.

The centrality (closeness index), c is function of distances of vertices.

$$C_c(v) = \frac{1}{|V|} \sum_{u \in V} d(u, v)$$

where $d(u, v)$ is distance between u and v for path traversal.

Effective Closeness

Effective closeness $C_{ec}(v)$ can also be analyzed. Use approximate average distance from v to all other vertices in place of the shortest paths. C_{ec} reduces run time for cases with a large number of edges and near linear scalability in computations.

Betweenness

Graph vertices betweenness means the number of times a vertex exists between the shortest path and the extent to which a vertex is located 'between' other pairs of vertices. Betweenness $c_B(v)$ of a vertex v requires calculating the lengths of shortest paths among all pairs of vertices and computations of the summation for each pairing vertex in V .

PageRank

PageRank is a metric for the importance of each vertex in a graph, assuming an edge from v_1 to v_2 represents endorsement of importance of v_2 by v_1 by connecting, following, interacting, opting for relationship, sharing belief or some other means.

Contacts Size

Contacts size means a vertex connection to many vertices. The size of each vertex does not convey any meaningful information. A big social graph network will also require high maintenance cost.

Indirect

Contacts

Indirect contacts metric means betweenness, which is the sum of the shortest paths within geodesic distances from all other pairing vertices. Three-step contact metric means a number of edges to other vertices plus the number of edges from other vertices within geodesic distances ≤ 3 .

Both metrics convey meaningful information. The indirect contacts metric has meaning in terms of magnitude of betweenness centrality.

Structure Diversity

Structure diversity metric means that social graph has access to diverse sub-graphs (knowledge).

Clustering in Social Network Graphs

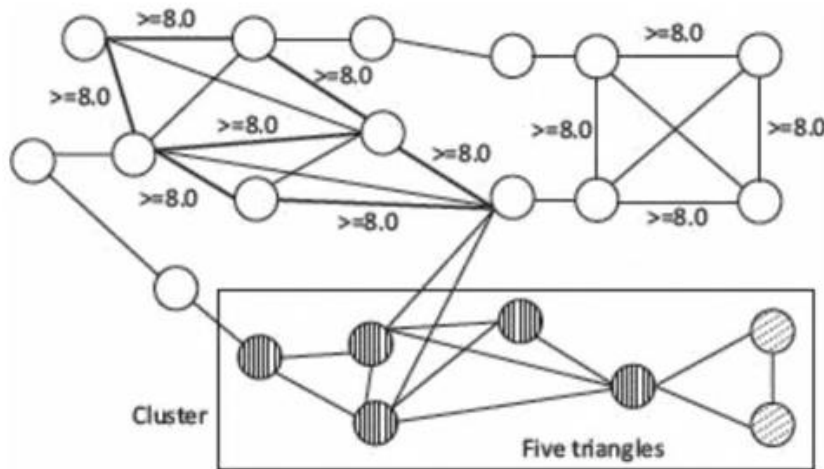
- One of the methods of detecting communities from social graph analysis is finding clustering and cluster coefficients. A clustering coefficient is a metric for the likelihood that two associated vertices of a vertex are also associated with other vertices. A higher clustering coefficient indicates a greater association and cohesiveness.
- Connected components mean components of the datasets (represented by properties of vertices) connected together. For example, finding student-teacher datasets, social network datasets, etc.

Sim Rank

- Similarity can be defined by properties of graph vertices. For example course, subject, student, scientist, Java programmer, status, values, or any other salient characteristic. Social network analysis of graphs computes SimRank.
- SimRank is the metric for measuring similarity between vertices of the same type. The computation starts from a vertex possessing specific property and path traversals through the edges search the similarities. The vertices having similar properties are counted to the SimRank.

Counting Triangles and Graph Matches

- One of the methods of detecting communities is counting of triangles. A triangle means three vertices forming a triangle with edges interconnecting them.
- Triangle count refers to the number of triangles passing through each vertex. The count is a measure of clustering. A vertex is part of a triangle when it has two adjacent vertices with an edge between them.
- Graph matches are computed using filtering or search algorithm, which uses the properties, labels of vertices, edges or the geographic locations.
- shows triangles and triangles between similar graph properties found from graph matches. Edge labels show the GPAs of students socially connected.



Figure

9.14 Clustering of five triangles and three matches of graphs

Using Spark Graph(Map-Reduce)for Network Graphs

- describes Spark GraphX algorithms for analyzing graphs. Connected components compute by graph. connected Components () . vertices method in Spark Graph. Connected Components Algorithm labels each connected component of the graph with an ID. Each connected component ID is ID of the lowest-numbered vertex. For example, in a social network, connected component objects can approximate clusters. GraphX contains an implementation of the algorithm in the Connected Components Object. The clusters are found by discovering close-by connected components using closeness centrality metric.
- SparkGraphX triangle-count algorithm computes the number of triangles passing through each vertex. The count is a measure of clustering. TriangleCount requires the edges to be in canonical orientation (srcId < dstId). Source vertex ID is srcId and Destination vertex ID is dstID. Graph is partitioned using Graph.partitionBy operator.

Direct Discovery of Communities

- Three metrics identify groups and communities from a social graph:
 1. Cliques - A clique forms by a set of vertices when each of the vertices directly connects to every other individual vertex through the edges. Detecting the cliques leads to direct discovery of communities.

2. Structurally cohesive blocks.
3. Social circles from connections and neighbourhoods

A bridge enables the link between two groups. Application of analyzing communities, SimRanks and bridges are finding a set of experts, specific areas of expertise, and ranking the expertise in an organization.

Experience in social science fields shows that the social network of a person is the key indicator of the stature of the person and his/her success potential. Social graph analysis enables finding key bridges and persons with most connections.