Big Data Overview and Injection

Session-1

Big Data

Collection of large, complex and diversa data sets that are difficult to manage.
Data can be structured, unstructured and semi-structured data that continues to grow exponentially over time.

Why to use Big Data

① large No. of Users on internet ~~platforms~~, IOT devices, social media,

Global data generation

Over ~~1000~~ 150 zettabytes expected by 2025

1 zettabytes = 1 billion Terabytes

$= 10^{21}$ bytes

Page No.
Date : / /

Categories of Data

1. Structured Data
2. Semi-structured Data
3. Unstructured Data

1. Structured Data

Data which is organized in rows and columns,

Data containing a defined data type, format, and structure.

ex → CSV files, simple spreadsheets, traditional RDBMS.

This type of data can be easily stored and queried in database.

SQL databases like MySQL, PostgreSQL, Oracle.

Real-World App^n

1. Banks having transactions & balances

(2) Semi-Structured Data

This type of data does not follow the rigid structure of traditional databases yet it contains some level of organization and can be processed.

Flexible Schema
Schema is not strict, predefind

More human readable

eg JSON files → Key-Value pair
XML

```
{ "firstname": "Jane",
"Lastname": "Doe"
}
```

(3) Unstructured Data

↳ Does not follow any definite Schema or set of rules

→ Photos, videos, text documents, Pdfs

→ Youtube story videos

Session - 2

Page No.
Date : / /

Data Storage Mechanisms

1. Flat files
2. Relational Databases
3. NoSQL

Flat files

Data stored in plain text or binary files
(ex .txt , .csv)

```
name , sapid , batch
X , 123 , B1
Y , 345 , B2
```

Rows represents records, & column (if present) are separated by delimiter like commas or tabs.

Adv-

1. Easy to create & use
2. Lightweight & portable
3. Good for small datasets or simple apps.

Disadv.

1. Not good for large datasets because no indexing
2. Querying is limited
3. No relationships b/w data Because of no foreign keys

ex - CSV exports for reporting
Configuration files

2. Relational Databases (RDBMS)

Databases that store data in a structured format using tables, rows and columns.

These tables are related to each other using keys (primary keys, foreign keys)

Employee Table:

ID	Name	Dept
1	Vish	CSE
2	Ram	SCOS

Adv

(1) Support for complex queries using SQL.

(2) Data integrity & consistency (Data normalization, 1NF, 2NF — BCNF)

(3) Relationships b/w tables for efficient data organization.

Limitations

(1) Predefined schema, so it is less flexible

(2) Not well suited for unstructured data or semi-structured data.

Can be used in Banking Systems, E-Commerce platforms

MySQL, PostgreSQL, Oracle, Microsoft SQL server

(3) NoSQL Databases

These are non-relational & designed for handling large volumes of unstructured or semi-structured data.

Doesn't rely on a fixed schema & are designed to scale horizontally

Types of NoSQL

1. Document-based → MongoDB
2. Key-value pair → Redis, DynamoDB
3. Column-based → Cassandra, HBase
4. Graph based → Neo4j

Adv

1. Flexible schema
2. Horizontal scalability for large scale app^n
3. Handle semi str. or unstr. data

Use

1. Social media app^y

Page No.
Date : / /

Comparison

Str.	Flat files simple	RDBMS Predefined schema	NoSQL Flexible schema
Data Relationships	None	Strong (via keys)	Limited or schemaless
Scalability	Limited	Vertical scaling	Horizontal scaling
Uses	Small data set	Structured data	Unstructured data

Characteristics of Big Data -

The 5 V's of Big Data

(1) Volumne (2) Velocity (3) Variety

(4) Value (5) Veracity

(1) Volume :

The enormous size of data generated & stored

storage : cloud storages like Amazon S3

Processing : Tools like Hadoop & Spark help process large data sets

② Velocity : Speed of Data Generation & processing

The rate at which data is generated, collected & processed in realtime

* Real-time processing requires tools like Apache Kafka or Flink
* Network bandwidth for high speed transmission.

③ Variety : Diversity of Data formats

Big data can store data in various formats

(str, unstr, semi-str data)

- str → Spreadsheets, RDBMS
- unstr → Videos, Images
- semi-str → JSON, XML

④ Value : Insights Derived from data

The importance or worth of data in driving decisions & generating business value.

Benefits of Big Data

Data Mining

Data mining

Risks of Big data

1. Privacy concerns
2. Security Risks
3. Data Quality issue
4. High costs of infrastructure
5. Ethical & legal challenges
6. Complexity & skills Gap

Big Data Analytics

The ability to analyze and make sense of the data is Big Data Analytics

It is a key driver of innovation, efficiency and competitive advantage.

Organizations that harness Big Data effectively can make more informed decisions, predict market trends, personalize customer experiences and uncover new opportunities

Importance of Big Data Analytics

(1) Transforming Data into Insights

Big data Analytics is the process of examining large and varied datasets to uncover hidden patterns, unknown correlations, market trends & other valuable insights.

Techniques used:-

Data Mining :→ Exploring datasets to find patterns

Statistical Analysis: Applying statistical methods to understand data.

Predictive Analytics: Using historical data to make predictions about future events.

Machine Learning: Algorithms that improve through experience, identifying complex patterns

(2) Competitive Advantage

Organizations that adeptly leverage Big Data Analytics can significantly outperform their competitors

(3) Enhancing Customer Experience
- Personalization
- Customer Engagement

(4) Operational Efficiency
- Process Optimization
- Predictive Maintenance

(5) Innovation & New Opportunities
- Product Development
- New Business Model

Impact of Big Data Analytics Across Industries

HealthCare

Personalized Medicine :→ Using genetic data to tailor treatments

Predictive Analytics :- Identifying patients at risk of disease like diabetes or heart conditions.

Resource Allocation :- Optimizing the use of medical staff & equipment.

Electronic Health Records (EHRs) :
Centralizing patient information for better care coordination.

Case ex.

IBM Watson
Assists doctors in diagnosing diseases by analyzing vast amounts of medical literature.

Finance

Fraud Detection & Risk Management

- Real time monitoring :→ Analyzing transaction to spot anomalies
- Credit Scoring :→ Incorporating diverse data sources for more accurate assessments.

Personalized Banking Services

- Customer Segmentation :- Offering tailored financial products
- Investment Strategies :- Algorithmic trading based on market data analysis

Case → JP Morgan Chase :→ Uses machine learning for contract review, significantly reducing time and errors.

Retail and E-commerce

Customer Behavior Analysis

- Sales forcasting
- Marketing Strategis :→ Targeting promotions to specific customer segments.

Recommendation Systems

- Upselling and Cross-Selling
- Personalized shopping Exp.

Case → Netflix → Recommends shows based on viewing history → 80%

Page No.
Date: / /

Manufacturing

Supply Chain Optimization
- Demand Planning
- Inventory Management

Quality Control
- Defect Detection
- Process Improvement

Public Sector
- Smart Cities
  - Traffic Management
  - Energy Efficiency
- Public Safety
  - Crime Analysis
  - Disaster Response

Page No.
Date : / /

Challenges in Big Data Analytics

1. Data Quality & Management
   - Data Silos: → issue, → soln
   - Data cleanliness
   - Data Governance
2. Privacy & Security Concerns
   - Data Protection
   - Compliance
   - Ethical use of data
3. Skill gaps & talent shortage
4. Infrastructure

Future Landscope of Big Data Analytics

1. AI and ML
   - Enhanced Capabilities
     - AI Integration
     - Automation
   - Applications
     - NLP
     - computer vision
   - Impact
     - Decision support
     - Efficiency

(2) Edge Computing & IOT
- Real-Time Processing
- IoT Expansion

(3) Ethical Considerations

(4) Data as a Service (DaaS)

Emerging Trends in Big data

What drive trends in BD
1. Growth of IOT, AI and machine learning
2. Inc. data generation from social media, sensors and digital platforms
3. Demand for real-time analytics & insights

Why Emerging Trends Matter

1. Keep pace with evolving data needs
2. Leverage advancements to solve complex problems.

Page No.
Date : / /

Trends

① Edge Computing

→ Processing data closer to the source instead of sending it to central server.

adv → ① Reduces latency, improves efficiency
② ex:- Autonomous vehicles process data locally for faster decesion-making

② Integration of AI and Machine Learning (ML)

App^{n} in Big-Data
- AI models predict outcomes & provide actionable insights.
- ML automates tasks like anomaly detection & data classification

ex → AI-driven chatbots, recommendation engines

③ Block Chain for data security

Why Block chain matters in Big data
- → Decentralized and immutable ledgers ensure data integrity.
- → Enables secure sharing of sensitive data across organizations

ex → Tracking supply chain data securely using Block chain

Page No.
Date: / /

(4) Real-time and streaming Analytics

↳ Processing data in real-time for instant insights.

uses → * Fraud detection in Banking
* personalized content delivery on streaming platforms.

(5) Cloud-Native Big Data Solutions

↳ ex Amazon Redshift, Google Big Query

adv.

Scalability, cost-efficiency & on-demand resources.

Ethical Considerations in Big Data

(1) Data Privacy & Governance

↳ importance of user consent & transparency.

(2) Bias in Analytics

↳ Algorithms trained on biased data can perpetuate inequality

↳ Need for diverse data sets & fair AI practices.

3) Cybersecurity Risks

→ As data grows, vulnerabilities inc.

→ Need for robust encryption & secure systems.

Future Directions for Big Data

1) Quantum Computing

→ Potential

→ Solves complex Big Data problems exponentially faster

→ Appⁿ in cryptography, climate modeling & genomics

2) Democratization of Big Data Tools

→ Make Big Data accessible to non-technical users

ex:- User-friendly platforms like Tableau, Power BI.

3) Autonomous Data Platforms

→ Platforms that manage & analyze data with minimal human intervention,

ex Oracle's Autonomous Data Warehouse

(4) Rise of Synthetic Data

→ Artificially generated data used to train AI and ML models

adv → Overcomes privacy concerns, fills gaps in real-world data.

Hadoop Ecosystem

Apache Hadoop is an open source framework intended to make interaction with big data easier.

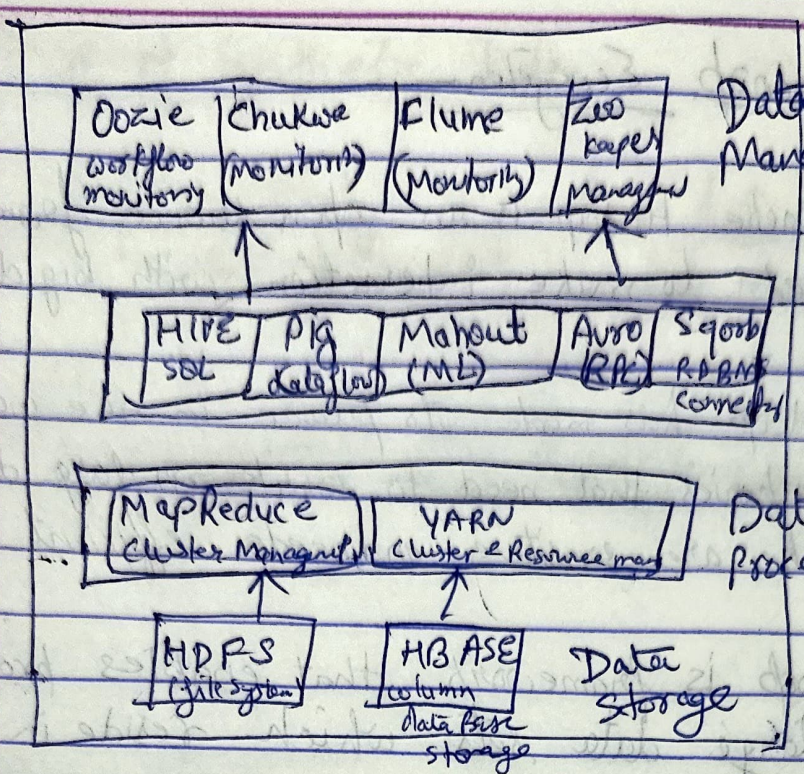
Hadoop has made its place in the industries & companies that need to work on large data sets which are sensitive and needs efficient handling.

Hadoop is framework that enables processing of large data sets which reside in the form of clusters.

Being a framework, Hadoop is made up of several modules that are supported by a large ecosystem of technologies.

Hadoop ecosystem is a platform or a suite which provides various services to solve the big data problems.

There are 4 major elements of Hadoop i.e ① HDFS ② Map Reduce ③ YARN ④ Hadoop common utilities.



Hadoop Ecosystem

HDFS :- Hadoop Distributed File System.

HBASE :- No SQL database

YARN :- Yet Another Resource Negotiater

MapReduce :- Programming based Data processing.

HIVE, PIG :- Query Based processing of data services.

Mahout, Spark, MLLib :- machine learning algorithm libraries

Sqoop,

Avro → Uses RPC to Serialize data in hadoop.

remote procedure calls

Oozie: Job Scheduling

Zookeeper: → Managing clusters

Solr, Lucene :→ Searching and Indexing

HDFS

→ primary or major component of Hadoop ecosystem and is responsible for storing large sets of structured or unstr. data across various nodes and thereby maintaining the metadata in the form of log files

HDFS consists of two core components i.e.

1. Name node :→ Prime node contains metadata (data about data) requires fewer resources than the data stores actual data
2. Data node

HDFS maintains all the coordination b/w the clusters and hardware.,

YARN → Yet Another Resource Negotiator

It is the one who helps to manage the resources across the clusters.

It performs scheduling & resource allocation for the hadoop System.

It consists of three major components :-

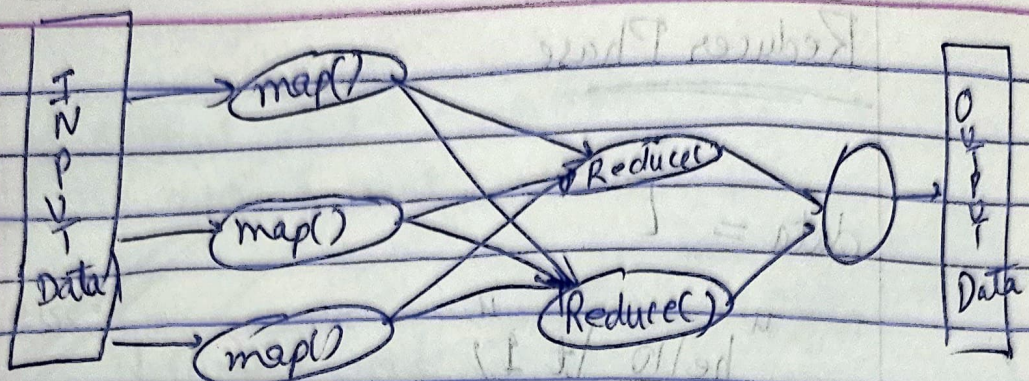
1. Resource Manager → for allocating resources for the app"
2. Nodes Manager → allocation of resources like CPU, memory, bandwidth per machine
3. Appⁿ Manager → works as an interface b/w resource manager & node manager. & performs negotiations as per the requirment of the two.

(3) Map Reduce :

→ It is used to make it possible to carry over the processing's logic & helps to write appⁿˢ which transforms big data set into a manageable one.

It has two functions (1) Map() (2) Reduce()

A MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks.



Word Count

Mapper

```
import sys

myline = myline.strip()        → Remove white space either side

words = myline.split()         → Break the line into words

for myword in words:
    print '%s \t %s' %(myword, 1)
        OR
    print(f"{word} \t 1")
```

ex Hello! How are you? How is your brother?

O/p

Hello → 1	How → 1
! → 1	is → 1
How → 1	your → 1
are → 1	brother → 1
you → 1	? → 1
? → 1	

Reducer Phase

```
data = [
    "hello \t 1",
    "! \t 1",
    "How \t 1",
    "are \t 1",
    "you \t 1",
    "? \t 1",
    "How \t 1",
    "is \t 1",
    "your \t 1",
    "brother \t 1",
    "? \t 1",
]
```

current_word = None
current_count = 0
word_county = {}
result = []

```
for myline in data:
    myline = myline.strip()
    try:                                  # split input from mapper
        word, count = myline.split('\t', 1)
        count = int(count)                # convert count to integer
    except ValueError:
        continue
```

check if the current word matches the previous one

```
if current_word == word:
    current_count += count
else:
    if current_word:
        result.append((current_word, current_count))
    current_word = word
    current_count = count
if current_word:
    result.append((current_word, current_count))
result
```

add the previous word & its total count to result

```
if word in word_counts:
    word_counts[word] += count
else:
    word_counts[word] = count
result = list(word_counts.items())
print(result)
```

Page No.
Date: / /

④ Hadoop Common utilities

Hadoop common utilities are nothing but our java library & java files or javascripts that we need for all the components present in a Hadoop cluster.

like HDFS, YARN & MapReduce for running the cluster.

Replication in HDFS & its importance

Replication ensures the availability of the data

The no. of times you make a copy of that particular thing can be expressed as it's replication factor.

default Replication factor = 3

Rack Awareness:

The rack is nothing but just the physical collection of nodes in our Hadoop cluster

A large Hadoop cluster is consists of so many racks with the help of this Racks Information Namenode chooses the closet Datanode to achieve the max. performance while performing the read/write info. which reduces the network traffic.

We divide the whole data into Blocks because of the increased seek time on the whole file.

HeartBeat:

It is the signal that datanode continuously sends to namenode. If namenode doesn't receive heartbeat from a datanode then it will be considers it dead.

Features of HDFS

1. Distributed Storage
2. Blocks reduces seek time
3. Data is highly available
4. High fault tolerance

Page No.
Date: / /

Limitations of HDFS

(1) Low latency data access

Apps which require low-latency access to data will not work well with HDFS

(2) Small file problem:-

Having lots of small files will result in lots of seeks & lots of movement from one datanode to another datanode to retrieve each small file

Features

(1) Scalable Storage

- Horizontal Scalability → Distributed Architecture
- support for large file

(2) Data Locality

HDFS moves computation tasks to the nodes where data resides, rather than transferring large dataset over the network

↓

Reduce data transfer latency & network congestion

③ Resilience

HDFS ensures high availability & reliability of data through redundancy & replication.

- Replication factor
- Automatic Recovery → in case of node failure the system automatically re-replicates blocks to maintain the replication factor.

④ Fault Tolerance

HDFS is designed to handle h/w & s/w failures without affecting data integrity or availability.

- Heart Beat & Block Reports:- namenodes continously monitors datanodes through heart beat & block reports
- failover Mechanism :- If a datanode fails, the namenode reroutes tasks & re-replicates lost blocks

HDFS Administration

HDFS administration involves managing the ~~Hadoop~~ HDFS for efficient storage & processing of big data.

Tasks of administration

1. Cluster Monitoring
   - Use tools like Ambari, Cloudera Manager or HDFS command-line utilities to monitor node health, disk usage & block distribution.
2. Replication Management
   - Adjust replication factor to balance fault tolerance & storage efficiency.

Use "hdfs dfsadmin -setrep" to configure replication levels

Page No.
Date: / /

(3) Namenode Operations

⇒ Monitor & manage the nameNode,
Regularly back up NameNode metadata

(4) Balancing cluster Load

A a tool is used to distribute data evenly across DataNode
The tool is "Balances tool"

(5) Disk Usage Management

Periodically clean up temporary & unused files to optimize storage.

Setting up Hadoop Cluster

There are two ways in which we can install it
→ Single Node Cluster
→ Multi-Node Cluster

Single Node Cluster

:- It has one DataNode running and setting up all the NameNode, DataNode, Resource Manager & Node Manager on a single machine.

This is used for studying & testing process

Page No.
Date: / /

Multi Node Cluster

Has more the one DataNode running & each DataNode is running on different machines.

Installation steps on a Single Node Cluster

1. JAVA - Java JDK Install
2. HADOOP - Hadoop package (download)
   Extract Hadoop at C:/Hadoop
3. Setting up the Hadoop-HOME variable
   (use windows environment variable setting for Hadop path setting)
   ↳ go to environment properties / or edit environment variables
4. Set JAVA_HOME Variable
5. Set Hadoop & Java bin directory path

(6) Hadoop Configuration

We need to modify Six files

1. core-site.xml
2. Mapred-site.xml
3. Hdfs-site.xml
4. Yarn-site.xml
5. Hadoop-env.cmd
6. ~~Hadoop~~ Create two folders datanode & namenode.

(1) Core-site.xml

It contains the core configuration settings for Hadoop, primarily specifying fundamental parameters related to the HDFS & general behaviour of the cluster.

⇒ Specify the Default file System

→ Defines the URI of the default Hadoop file system

(Uniform Resource Identifiers)

property: fs.defaultFS → URI of the default file system

Page No.
Date: / /

hadoop.tmp.dir :- Base directory for temporary files

io.file.buffer.size :- size of the buffer for reading & writing files

hadoop.proxyuser.<user>.hosts → Define proxy hosts for a specific user

hadoop.proxyuser.<user>.groups → Define groups allowed to use proxy for a specific user.

fs.s3a.access.key → Access key for Amazon S3 storage if using S3 as a file system

fs.s3a.secret.key → Secret key for Amazon S3 storage

Role of core-site.xml

(1) act as central configuration file that influences how hadoop interacts with underlying file system & network

(2) Plays a crucial role in setting up & running a Hadoop cluster effectively

(3) Ensures compatibility & optimal performance for distributed system

(2) hdfs-site.xml

Contains configurations specific to HDFS

dfs.replication → no. of data block replicas

dfs.namenode.name.dir (path for NameNode storage)

dfs.datanode.data.dir → path for Data Node storage

(3) mapred-site.xml

→ Contains MapReduce framework-specific configurations

mapreduce.framework.name

→ (framework name ex yarn)

mapreduce.jobtracker

(Job Tracker hostname & port in a classic MapReduce Setup)

(4) yarn-site.xml

Contains configuration for YARN which manages cluster resources

yarn.resourcemanager.hostname
(ResourceManager Hostname)

yarn.nodemanager.resource.memory-mb
(memory allocated for NodeManager)

yarn.nodemanager.vmem-check-enabled
(Virtual memory checks)

(5) hadoop-env.sh

A shell script to configure env. variables for hadoop such as java path

JAVA_HOME (path to Java installation)

HADOOP_HOME (path to the Hadoop installation)

(6)

Page No.
Date: / /

Data loading into Hadoop

HDFS is designed to store large volumes of data reliably & efficiently.

Loading data into HDFS & other Hadoop components can be achieved through various methods & tools, each suited for to different types of data sources & use cases.

Data loading Methods

1. HDFS Commands
2. Apache Sqoob
3. Apache flume
4. Apache kafka
5. Apache NiFi
6. DistCp (distributed Copy)
7. Using Hadoop APIs

HDFS comand

CLI → command line Interface

Uses cases → (1) Small to medium datasets
↳ ideal for loading data that can be managed via (CLI) without the need for automation

(2) Useful for quick, one-time data uploads or downloads

Page No.
Date: / /

Adv

① Easy to use → basic command line operation
② No additional tool required
direct interaction with HDFS

Limitation

Manual process
↳ not suitable for large-scale or automated data ingestion

Handling of massive data is troublesome.

② Apache Sqoop

↳ It is a tool designed for efficiently transferring bulk data b/w hadoop & structured datastores such as relational databases

Import data → RDBMS to HDFS
Export data → HDFS to RDBMS

Use

① Migrating data from enterprise databases to Hadoop for analytics
(ETL)

② Part of Extract, Transform, Load workflow to prepare data for processing.

Page No.
Date: / /

Adv: ① High performance → parallel data transfer utilizing Map Reduce.

② Seamlessly integrates with Hadoop ecosystem (Hive, HBase)

③ Automatically generates Hadoop schemas based on database tables.

Limitation
① Limited to structured
② Dependency on RDBMS

③ Apache Flume

It is a distributed, reliable & available service for collecting, aggregating and moving large amounts of data log or event data from various sources to HDFS

Designed to handle high throughput data ingestion.
Ensures data is not lost during transfer with built-in fault tolerance
flexible through customizable sources, channels & sinks

Use
① Logs Aggregation → Collecting logs from multiple apps into HDFS for centralized storage & analysis
② Real time Analytics → Streaming event data for near

real-time processing with frameworks like Apache storm or Spark Streaming.

Adv
1. Real time data ingestion
2. Fault tolerance
3. Easy to use

Limitation
1. Customizing beyond std setups can be challenging
2. High-throughput ingestion may require significant cluster resources.

4. Apache Kafka

It is a distributed event streaming platform capable of handling large volumes of data in real-time.
While Kafka itself is not a data loader, it serves as a robust ingestion pipeline that can transport data to Hadoop components like HDFS, Hive or Spark.

1. High Throughput & low latency → millions of messages per second
2. Durability & fault tolerance
3. Scalability :- Horizontally scalable.

Page No.
Date: / /

Adv ① Real-Time Processing
② Versatility → can handle diverse data sources & sinks
③ Scalability

Disadvantages
① Operational complexity :→ Kafka cluster managing requires expertise.
② Very careful configurations are required to ensure minimal delays.

Map/Reduce Concepts.

Map

→ Takes input data and transforms it into key-value pairs
→ Processes each record independently.
→ Runs in parallel across multiple machines

Reduce

→ Takes the output from Map phase
→ Aggregates/combines values with the same key.
→ Produce final o/p.
→ Also runs in parallel for different keys.

Input → Map → Shuffle/sort → Reduce → O/P.

Page No.
Date: / /

Managing Job Execution using Yarn

YARN involves several steps & components that coork together to efficiently allocate resources & execute appns in a distributed environment.

Detailed Overview of How YARN manages job execution

① Job Submission:

Client Interaction :- The process begins when a client submits an appn to the YARN resource manager (RM)

This submission includes the appn code, configuration and resource requirements.

② Resource Manager (RM)

Resource Allocation : The RM is the master daemon responsible for managing resources across the cluster.

→ It decides how to allocate resources to various appns based on available resources and scheduling policies.

Application Master (AM) Launch : The RM allocates a container for the Appn Master (AM) on one of the

nodes.

→ The AM is responsible for managing the lifecycle of the app^n.

③ Application Master (AM)

Resource Negotiation: The AM negotiates with the RM for additional resources (containers) needed to execute the app^n tasks.

Task Coordination: The AM coordinates the execution of tasks, monitors their progress and handles task failure

④ Node Manager (NM)

Container Management: Each node in the cluster runs a Node Manager, which is responsible for launching and monitoring containers on that node

Resource Monitoring – The NM reports resource usage and container status back to the RM.

Page No.
Date: / /

(5) Container Execution

Task Execution: Containers are the units of resource allocation in YARN. They encapsulate the environment in which app^n tasks run, including CPU, memory & other resources.

Isolation: Containers provide isolation b/w different tasks, ensuring that they do not interfere with each other.

(6) Scheduling and Resource Management

Schedulers :→ like FIFO manages how resources are allocated to app^{ns}.

Queue Management: App^{ns} are submitted to queues, which can be configured with different resource capacities & priorities.

(7) Monitoring and Logging

Web UI: YARN provides a web-based user interface for monitoring the status of app^n, resource usage & cluster health.

Logs: App^n logs are collected & can be accessed for debugging & performance analysis

Page No.
Date : / /

(8) Fault Tolerance

Failure Handling → YARN is designed to handle failures gracefully. If a container or node fails, the AM can request new resources to restart the failed tasks.

AM Recovery → In case of AM failure, YARN can restart the AM to continue managing the app^{n}.

(9) Security

Authentication & Authorization → YARN supports Kerberos-based authentication and provides mechanism for authorizing access to resources and app^{ns}.

Access Control → Queue-level access control lists (ACLs) can be configured to manage user permissions.

(10) Optimization and Best Practices

Resource Configuration → Properly configure resource allocations for container optimize performance.

Queue Configuration → Set up queues to reflect organizational priorities & adjust configurations needed to improve efficient

Data Analysis & Visualization

Data Analysis is the process of inspecting, cleaning, transforming and modeling data to extract useful information, draw conclusions & support decision-making.

Types of Data Analysis

(1) Descriptive Analysis :- Summarizing data using statistics like mean, median, mode, variance etc.

(2) Diagnostic Analysis → Identifying reasons behind past outcomes

(3) Predictive Analysis :- Using ML models to forecast future trends

(4) Prescriptive Analysis → Suggesting actions based on insights from data.

Data Visualization

It is the graphical representation of data using charts, graphs & plots to make complex data more understandable.

eg Bar Chart, Line Chart, Pie-Chart, etc.

Page No.
Date: / /

How to use Data Analysis & Visualization in projects

(1) Define the problem statement
Identify what you want to analyze

(2) Collect & Clean Data

(3) Perform Exploratory Data Analysis (EDA)
- Identify patterns & trends using summary statistics
- Create basic visualizations to understand distributions

(4) Apply the ML models

(5) Visualize Results

Real life Use

(1) Health Care Analytics → Hospitals use data analysis to predict disease outbreaks & optimize patient care.

(2) Stock Market Analysis → Traders use predictive analytics to forecast stock prices.

Different Sources of Big Data

(1) Social Media Types

FB, Insta, Twitter

Data Type :→ User Interactions, posts, likes, shares, comments, multimedia

Characteristics :- High Unstructured, high volume and real-time data that offer insights into consumer behaviour, trends & sentiments

(2) Machine-Generated Data

Log files, system logs, application logs, clickstream da

Sources :- Servers, applications, network devices

Characteristics :- Str. and Semi Str. data that can be used for monitoring performance, security analysis & user behavior tracking.

③ Transactional Data

Sources → Point-of-sale systems, Online Transactions, electronic payments, e-commerce purchases.

Type of data → Purchase details, payment records, invoices, receipts.

Char. → Structured Data critical for financial analysis, inventory management & customer relationship management.

④ Sensor Data / IOT

Sources → Wearables, smart home devices, industrial sensors, environment sensors, GPS devices.

Type of data → Temperature readings, motion detection, heart rate, location data.

Char. → High-volume, time-stamped data that is often continuous & requires real-time processing.

(5) Web Data

Sources :- Websites, blogs, forums, online news articles.

Types of Data :- Webpage content, metadata, user reviews, testimonials.

Char. :- Semi-Str. ~~and~~ or Unstr. data useful for market research, competitive analysis & sentiment analysis

(6) Scientific Research Data

Physics, CS, astronomy — etc.

Types of Data :- Simulation Data, Experimental results, etc.

Char :- Extremely large datasets that require specialized processing capabilities, often used in advanced research & development.

(7) Audio & Video Data

Sources :- Surveillance cameras, video conferencing, media podcasts, podcasts

Types of data :- Multimedia files, streaming content.

Charact. :- Unstr. data that requires large storage capacity & specialized processing for analysis like

speech recognition and video analytics.

(8) Text Data

Sources: Emails, documents, PDFs, books, SMS messages.

Type of Data: Unstructured Text requiring NLP for analysis.

Char: Valuable for extracting insights on communication patterns, topic modeling and information retrieval.

Introduction to Data Ingestion

It is a fundamental process in the realm of data management & analytics.

It involves the collection, transportation & preparation of data from various sources into a storage or processing system where it can be analyzed, visualized or used in apps.

Data ingestion refers to the process of obtaining and importing data for immediate use or storage in a database.

Introduction to Data Ingestion

It is a fundamental process in the realm of data management & analytics.

It involves the collection, transportation & preparation of data from various sources into a storage or processing system where it can be analyzed, visualized or used in apps.

Data ingestion refers to the process of obtaining and importing data for immediate use or storage in a database.

(5) The data can come from a multitude of sources like

- (1) Databases :→ Relational & Non Relational DB
- (2) Files : CSV, JSON, XML
- (3) API : → Data retrieved through RESTFUL APIs
- (4) Streaming Data : Social media, logs
- (5) Third Party Sources : External data sets, partner data

Why Data Ingestion Important :

(1) Data Driven Decision Making

(2) Scalability

(3) Data Integration

(4) Operational Efficiency

(5) Real-time Analytics

Types of Data Ingestion

It can be categorized based on freq & method of data transfer.

Page No.
Date: / /

(1) Batch Ingestion → data is transferred & collected in large block at scheduled time interval

(2) Real-time Ingestion → Data is ingested continously & processed in real-time as it arrives.

(3) Micro-Batch Ingestion

↳ A hybrid approach where data is collected in small batches at frequent intervals.

Data Ingestion Methods

(1) ETL (Extract Transform, Load)

(2) ELT (Extract, Load, Transform)

(3) CDC (Change Data Capture)

(4) Streaming Ingestion

ETL →

Extract → Retrieve data from various sources (databases, APIs, flat files etc).

Transform → Clean the data, enrich it and structure it (like removing duplicates, converting formats.

Page No.
Date: / /

Load :→ Insert the transformed data into a target system

Target system can be → data warehouse, databases.

→ ETL can be used in BI business Intellig. & to process the structured data.

Advs.

→ It ensures data quality & consistency before loading.

→ Good for structured, historical data analysis.

→ Works well for scheduled batch processing.

Disadv. → slower for large datasets as transformation occurs before loading.

→ Not ideal for real-time appns.

Tools → Apache Nifi, AWS Glue

ELT

Extract → Data is retrieved from source systems.

Load → Data is loaded directly into a cloud data lake or warehouse

Snowflake, Google Big Query.

Transform:
Data is transformed within the destination system using SQL or processing systems.

ELT can be used for Big Data & cloud-Based analytics,
storing raw data for later analysis,
for large data volumes.

Adv → Faster ingestion as no pre-transformation is required

→ leverages cloud-based storage for efficient processing.

→ More flexible for data scientists to explore raw data.

Page No.
Date: / /

Disadv
→ Requires powerful storage & processing infrastructure.

→ If data is not well-structured, may lead to inefficient queries.

Tools — Snowflake, AWS Redshift

CDC (Change Data Capture)

It is a tech. that captures changes (insert, update, delete these kind of changes) in a database & propagates them to another system in real-time or near-real time.

Detect Changes :- Identify modified records in source tables.

Capture Changes :- Log the changes as events.

Replicate changes :- Apply the changes to the target system.

Date: / /

Types of CDC

1. Log-based CDC → Reads database transaction logs. It is efficient, minimal impact.
2. Trigger-based CDC → Uses database triggers. It can slow performance.
3. Timestamp-based CDC → Compares timestamp. It will require additional indexing.

CDC is used for Real-time database replication.
→ for incremental data processing.
→ To synchronize data across systems.

Adv:
→ Real or near-Real time updates.
→ Reduces data transfer costs by sending only changed data.
→ Useful for event-driven architectures.

Disadv:
→ Maintaining logs, time stamps can lead to some additional load on source data.
→ log-based CDC may require database specific configuration.

Tools → Oracle Golden Gate, AWS DMS (Database Migration Service)

Date: / /

Streaming Data Ingestion

It is the processing of continuously collecting, processing & transferring data in real time as it is generating.

Streaming Ingestion ensures :→

1. Low latency
2. Scalability
3. fault Tolerance
4. Event-Driven Processing → It works on individual data events as they arrive

It can be used in

- → Financial Transactions
- → IoT Sensor Data
- → Social Media Analytics → (Trends)
- → Cyber Security

Challenges in Streaming Ingestion

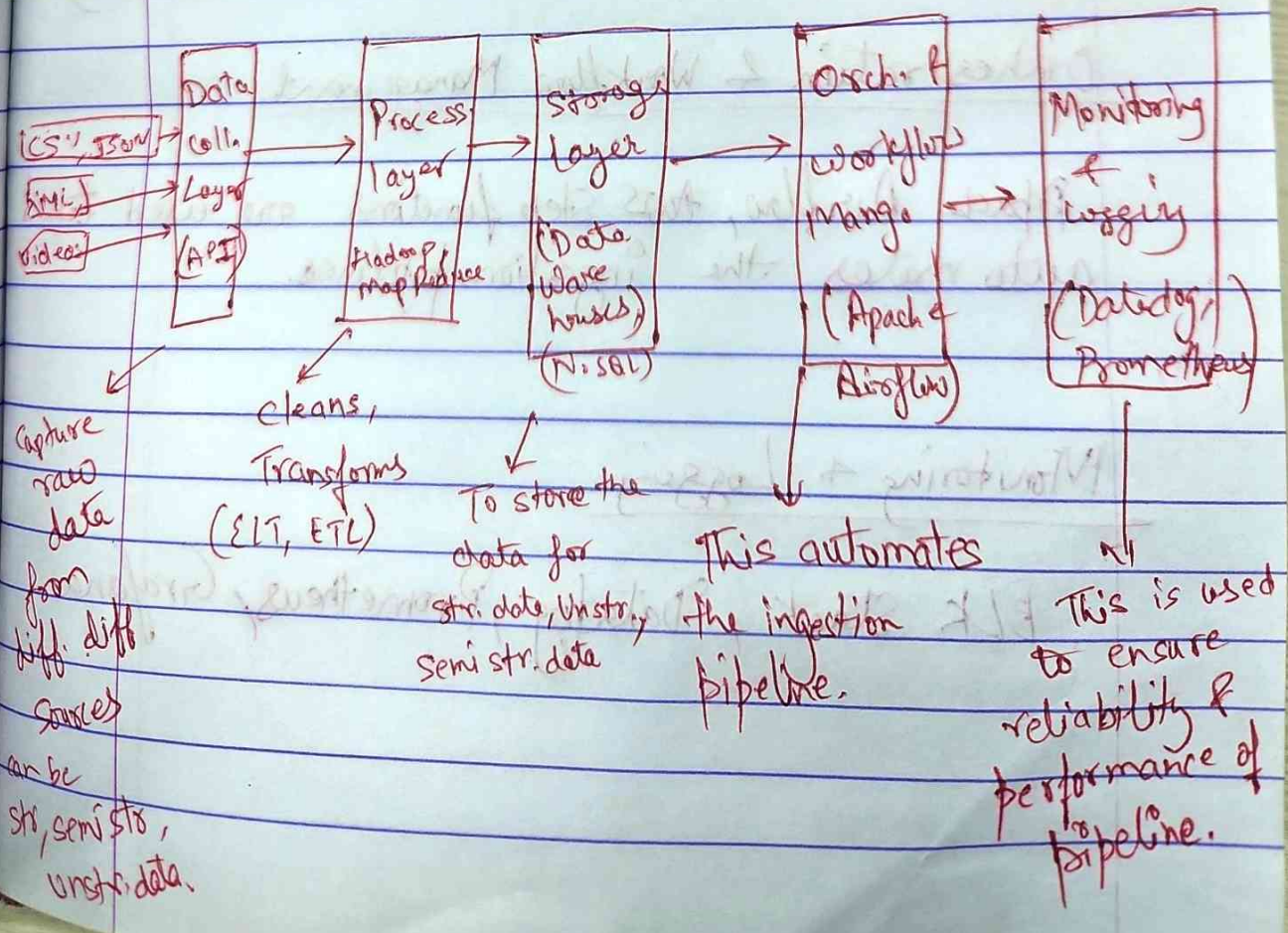
- → Data Ordering & Consistency
- → Scalability
- → Latency vs Accuracy Trade-off

Data Ingestion Pipeline

It is responsible for collecting, processing and storing large volumes of data efficiently

Key Components of a Data Ingestion Pipeline

1. Data Sources
2. Data collection layer
3. Processing Layer
4. Storage layer
5. Orchestration & Workflow Management
6. Monitoring & Logging



Page No.
Date: / /

Data Collection Examples :→ APIs, IoT sensors etc.

Processing Layer :→ Apache Spark, Hadoop MapReduce for Batch Tools.

Streaming Tools, Kafka Streams

Storage layer :-

Str. data → Data Warehouses like Google BigQuery.

Unstr. data → Data Lakes like (AWS S3, Azure data Lake) for raw, unstr data

Semi Str. data → NoSQL Databases (MongoDB, Cassandra)

Orchestration & Workflow Management :

Apache Airflow, AWS Step functions are used to automates the ingestion pipelines.

Monitoring & Logging

ELK Stack, Datadog, Prometheus, Grafana

Unit → 5

Page No.
Date : / /

Techniques for ingesting batch data from various sources.

Date: / /

Sqoop Arch.

Sqoop → Feature

1. Helps to connect the result from SQL queries into HDFS
2. Helps to load processed data into Hive or Hbase.
3. Can performs security operations of data (using kerberos).
4. can compress data.

Main operations performed in Sqoop

(1) Import
(2) Export

Sqoop used the command-line interface (CLI) to process command of user.
(Can also use APIs)

Sqoop will only do import and export of data based on user command (Nothing else! no aggregation of data)

Working

Import: Sqoop will first parses argument (from CLI) and send them to next stage where arguments are induced for Map only job. (Only Mapper is used no Reducer).

Page No.
Date: / /

Once the Map receives arguments it then gives command of release of multiple mappers depending upon the number defined by the user (default 4) at the of CLI)

Each mapper task is assigned with respective part of data that is imported on the basis of key which is defined by the user in CLI)

To increase the efficiency of process Sqoop uses parallel processing technique

↙

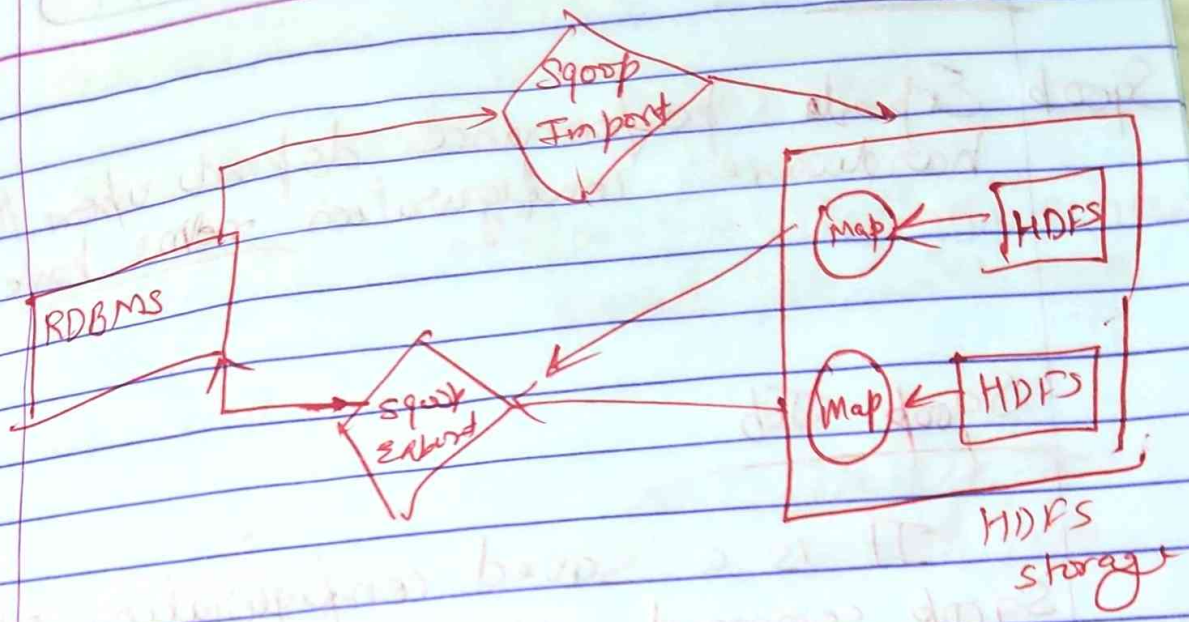
In this data is distributed equally among all mappers.

Then each mapper creates an individual connection with the database & then fetches individual part of data assigned by Sqoop.

Once the data is fetched then the data is been written in HDFS or HBase or Hive.

Export

Export is done in the same way, Sqoop export tool which available performs the operation by allowing set of files from the Hadoop system back to RDBMS.



Adv.

→ Can use variety of structured data stores like Oracle, Teradata etc.

→ Can perform ETL operation very fast & cost-effective way.

→ parallel processing of data leads to overall faster processes.

→ Supports fault Tolerance (multiple Map)

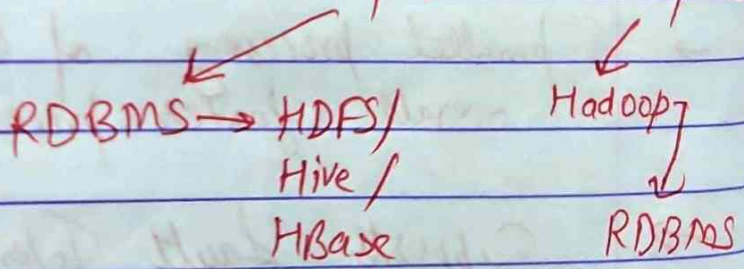
Sqoop Exports performance depends upon the hardware configuration rdbms have.

Sqoop Job

It is a saved configuration of a Sqoop command which can be use later to run or schedule without having to re-enter all the parameters each time.

It encapsulates all the necessary details for moving data between a relational database and a Hadoop ecosystem component.

It can be data imports or exports



It is reusable. It stores the parameters like database connection URL, credentials, table name, target directory in HDFS and can be other things.

It can be simply tigger the job whenever needed without recreating the

command line.

How to create a sqoop job.

--create <job-id> :→ Define a new saved job with the specified job-id

--delete <job-id> :→ Delete a saved job

--exec <job-id> :→ Given a job defined with --create, run the saved job.

--show <job-id> :→ Show the parameters for a saved job.

--list :→ List all saved Jobs.

--create my-job

```
$ sqoop job --create myjob -- import
--connect jdbc:mysql://example.com/
db --table mytable
```

To execute

```
$ sqoop job --exec myJob
```

Date: / /

Sqoop jobs simplifies repetitive tasks ensures consistency and is especially useful in scheduled or production environments where tasks need to be automated.

Incremental Import

The process of importing only the data that has changed since the last import operation.

Only newly added records needed to be updated. It will reduce the time to completely import the whole dataset.

There are two modes of incremental import

a) Append Mode
b) Last modified Mode

Append Mode :-

Sqoop imports only the rows where the value in a designated column is greater than the last value

imported. This mode works best when new rows are added to the Table.

Last modified Mode :→ Sqoop uses a timestamp column to track changes.

File Handling in incremental Imports

To handle we have to specify to sqoop where to write the imported in HDFS.

So, --target-dir option is used

For incremental imports, we must decide whether to write to the same target directory or to use a new directory for each run.

To write in same target directory we can use --append flag

↓

It means adding the new files alongside the other files.

Handling & generation of small files

As the volume of data from incremental changes can be relatively small, this could lead to many small files.

Large number of small files can lead to performance issues when processing or querying the data later.

↳ To solve this issue, periodical consolidation of these files is necessary. This can be done using file merging or compaction job.

When writing to the same target directory over multiple incremental runs, ensures that the file won't get overwritten accidentally.

In case of new target directory creation every time, we'll need a file handling process which merges or organizes these directories for processing.

Date : | |

Best Practices

1) Merge Small files
2) Logging & Monitoring
   ↳ monitor the target directories
3) Directory Organization
   ↳ Organize HDFS directories in a way that makes it easy to separate full imports from incremental imports.

Unit-6

Page No.
Date: / /

Introduction to message queuing systems

Message queues enable communication between various system components.

A message queue is a form of asynchronous service-to-service communication used in serverless & microservices.

Messages are stored on the queue until they are processed & deleted.

→ This is done in order to increase system reliability, scalability & flexibility.

This is achieved by adding a message broker (like RabbitMQ or LavinMQ) to pass messages or data b/w processes or systems.

Messages are sent by producer to the message Queue, where they are held until consumed by a consumer.

↓

This decouples the producer from the consumer, allowing them to operate independently.

↳ means data can be sent to a queue without waiting for immediate, real-time processing

Page No.
Date : / /

Message Queuing can be used for distributed computing, task scheduling & workflow management.

Basic features

(1) Decoupling Producers & Consumers

(2) Handling High Throughput & Scalability.

(3) Asynchronous Processing
↳ data ingestion often requires asynchronous handling.
Messages queues buffer the data, allowing downstream systems to process it at their own pace.

(4) ~~Reliability and Fault Tolerance~~ Load Balancing & Integration.
↳ Multiple consumers pulling data from a message queue, the workload can be distributed evenly, preventing any single processing node from being overwhelmed.

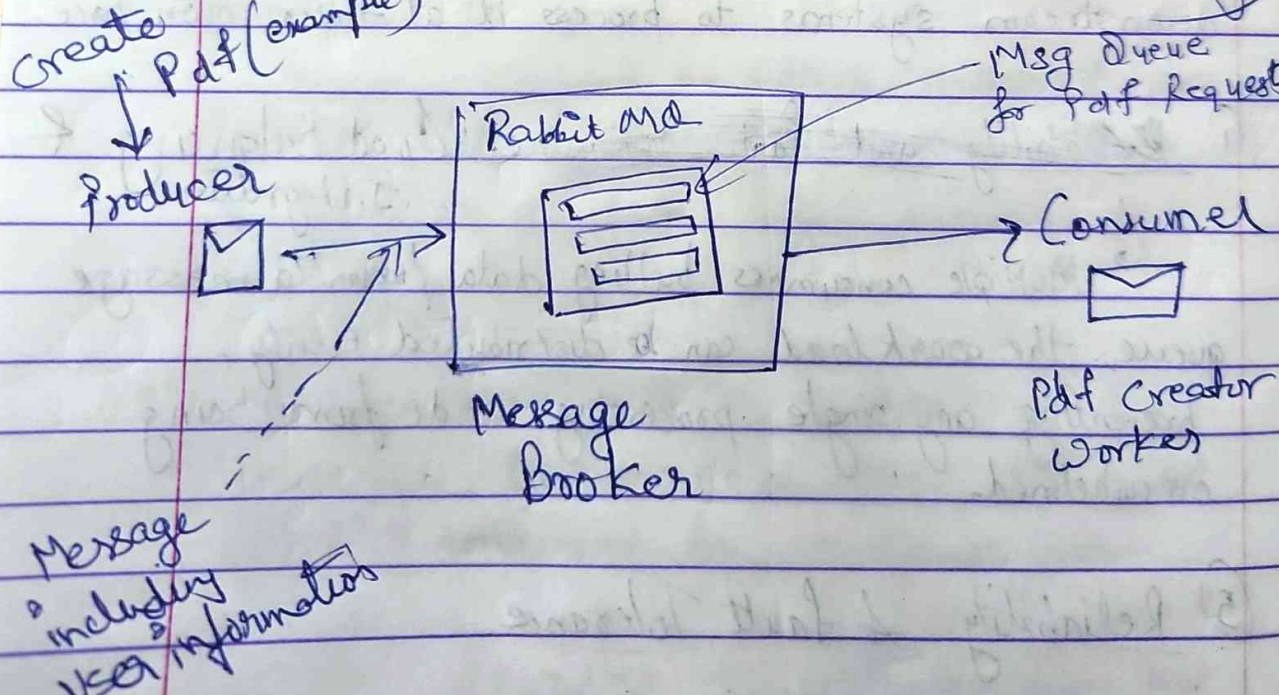
(5) Reliability & Fault Tolerance.

These systems typically offer features such as message persistence, automatic retries, & fault tolerance.

Page No.
Date : / /

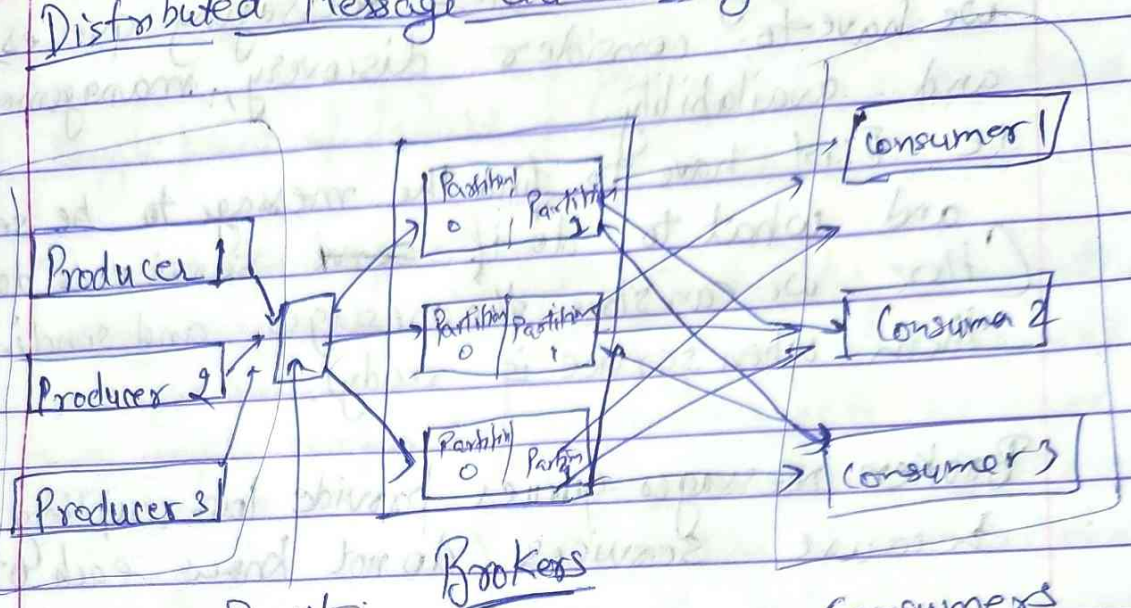
Arch. for Message Queuing Systems

1. Producer Layer
2. Messaging Layer (Queue or Broker)
3. Consumer Layer
4. Data Processing & Storage
5. Auxiliary Components & Considerations (like Load Balancing, Fault Tolerance, Monitoring)



Date : / /

Distributed Message Queue System



Broker v/s Broker Less messaging

Broker messaging arch. means we have a centralized broker acting like a mediator.
ex:→ RabbitMQ, IBM MQ

In Brokerless messaging arch., no broker exists and participants communicate directly with themselves.

ex:→ ZeroMQ, NanoMsg

In this seperate broker process does not exist.

Date: / /

When using Brokerless messaging queues, we have to consider discovery, management and availability.
i.e First have to find the message to be sent and what to do if ~~servi~~ service is down. (Here we can store the messages and sending them when service is ready).

Broker messages queues provide low coupling because services do not know each other.

Broker MQ require more network power than brokerless MQ.

Adv. of Broker	Adv. of Brokerless
(1) Loose coupling	(1) Simplicity
(2) Scalability → horizontal scaling possible	(2) Improved Performance: without the need to route messages through a broker, brokerless messaging can offer lower latency & higher throughput, especially for local comm. blw services.
(3) Reliability → message persistence & guaranteed delivery	
(4) Centralized Control	
(5) Message Transformation & routing :- Brokers have powerful routing capabilities.	

Date: / /

Disadv. of Broker	Disadv. of Brokerless
① Single point of failure	① Lack of centralized control.
② Increased ~~Cap~~ complexity :- setting & configuring a message broker can be more complex compared to brokerless.	② Scalability limitations :- may not scale as well as broker MQ because responsibilities for message distribution & load balancing lies with individual services
③ Potential Performance impact :- Additional overhead of routing messages	③ No built-in guaranteed delivery

Note:- **Message Routing**

It is the process of determining how messages are directed or forwarded from one point to another within a messaging system. (It is based on a set of rules, ~~etc~~ criteria or configurations)

Exchanges and Exchange Types

Exchange is a critical component responsible for routing messages to one or more queues based on defined rules

First producer sends a message to an exchange and then exchange decides which queue the message should be delivered to.
It is done utilizing attributes like routing keys and binding rules.

Routing Keys

A routing key is a string attached to a message by the producer when it is sent to an exchange.
It is like a label or tag that describes the message's content or category.

Binding Rules

Binding rules are configurations that associate a queue with an exchange using a binding key or a matching pattern.

When a queue is bound to an exchange, the binding rule specifies which messages the queue is interested in receiving based on the routing keys.

* ① Producer assigns a routing key to each message.
② The exchange uses this routing key to evaluate the binding rules set by queues.

(3) If the routing key of the message matches a queue's binding rule, the message is routed to that queue.

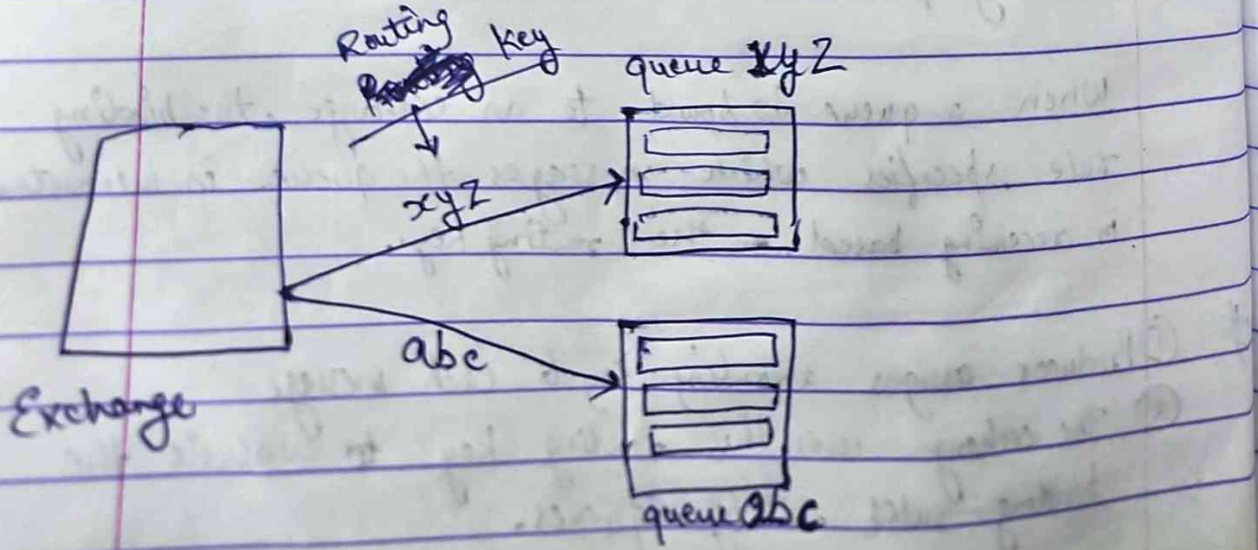
Types of Exchanges

1. Direct Exchange
2. Topic Exchange
3. Fanout Exchange
4. Headers Exchange.

(1) Direct Exchange

A message goes to the queues whose binding key exactly matches the routing key of the message.

when you need a simple one-to-one routing.
ex → sending routing tasks to specific workers.



Date : | |

If the message routing key does not match any binding key, the message is discarded.

If more than one queue is bound to exchange with same bounding key, the direct exchange will broadcast the message to all matching queues.

② Topic Exchange

It is similiar to direct exchange, but the routing is done acc. to the routing pattern.

It uses wildcards instead of using fixed routing keys. (* & #)

The routing key must consist of list of words delimited by a period ".".

"*" is used to match a word in a specific position of the routing key

ex data.*.*.list

→ only match routing keys where the first word is "data" & fourth word is "list".

"#" - it is used to match zero or more words

ex data.list.# → means all routing keys starting with "data.list."

Date: / /

③ Fanout Exchange

This routes a message to all queues that are bound to it regardless of routing keys or patterns.

It is useful when the same message needs to be sent to one or more queues with consumers who may process the same message in different ways.

④ Headers Exchange

It uses the message header attributes for routing.

A Header Exchange routes messages on arguments containing headers.

i.e Header contains some arguments like (key, value) pair, format, type, x-match.

x-match is a special argument added in the binding b/w exchange & queue.

It specifies if all headers must match or just one.

* match argument property can have two different values ① Any ② all.

Default value is 'All' → i.e all header pairs must pass, while value of 'any' means atleast one of the headers pairs must match.

Distributed publish-subscribe messaging System

It is an architectural pattern in which messages produced by publishers are distributed among multiple consumers (subscribers)

In this model, publishers (producers) post messages to a central topic or channel without knowing which subscribers will receive them, and subscribers express interest by subscribing to channels or topics

Publishers and subscribers operate independently i.e decoupling.
↳ It improves scalability & flexibility.

Topics and Channels
→ In many publish-subscribe systems, messages are organized by topics or channels.
A publisher sends a message to a specific topic, and all subscribers to that topic receive the messages.

These systems are designed to run across clusters of machine to distribute the load, manage network partitions & provide high throughput & fault tolerance
Components are horizontally scalable

Page No.
Date: / /

Modern distributed pub-sub systems often provide guarantees about message delivery and ordering.

ex → Apache Kafka, Apache Pulsar

Working

Pub-sub system has four key components

(1) Messages (2) Topics/channel

(3) Publisher (4) Subscribers

Messages

A msg is communication data sent from sender to receiver.

Msg datatypes can be anything from strings to complex objects representing text, video, audio -- etc.

Topics

The topic acts like an intermediary channel b/w senders and receivers.

It maintains a list of receivers who are interested in messages about that topic.

Topics serve as a logical grouping to organize messages by subject, type, or any

other categorical attribute.

Topic is a channel identified by a unique name.

Every message has a topic associated with it.

Subscribers

A subscriber is the message recipient. Subscribers have to register to topics of interest.

Publishers

The publisher is the component that sends messages. It creates messages about a topic & sends them once only to all subscribers of that topic.

The publisher doesn't need to know who is using the information it is broadcasting & the subscribers don't need to know where the message comes from.

Usage

Perform parallel asynchronous processing, Balance workloads, Deliver app^n & system alerts.

Kafka Broker

It is a server that handles the storage, retrieval & distribution of messages in an Apache Kafka cluster.

It is responsible for managing topics, partitions, producers (publisher) & consumers (subscribers).

A Kafka cluster typically consists of multiple brokers to ensure fault tolerance, scalability & high availability.

Apache Zookeeper manages the metadata, leader elections & broker coordination.

It is process where Apache Kafka dynamically selects a broker as the leader for a given partition.

The main task of the leader is to handle all read & write operations for that partition, while other brokers act as followers, maintaining replicas of the data.

If a leader broker fails, Zookeeper detects it & elects a new leader from the in-sync replicas (ISR)

Date: / /

Key functions of a Kafka Broker

(1) Message Storage :- Brokers store messages in topics, which are further divided into partitions

(2) Message Distribution :- distribute messages across multiple partitions for scalability & fault tolerance.

(3) Load Balancing :- Efficient distribution is done.

(4) Replication :- Supports replication of data

(5) Leader & follower Roles :- One broker acts as a leader other acts as followers.

(6) Handling Requests :- Brokers process read/write requests from producers & consumers

Producers & Consumers

Do it Yourself