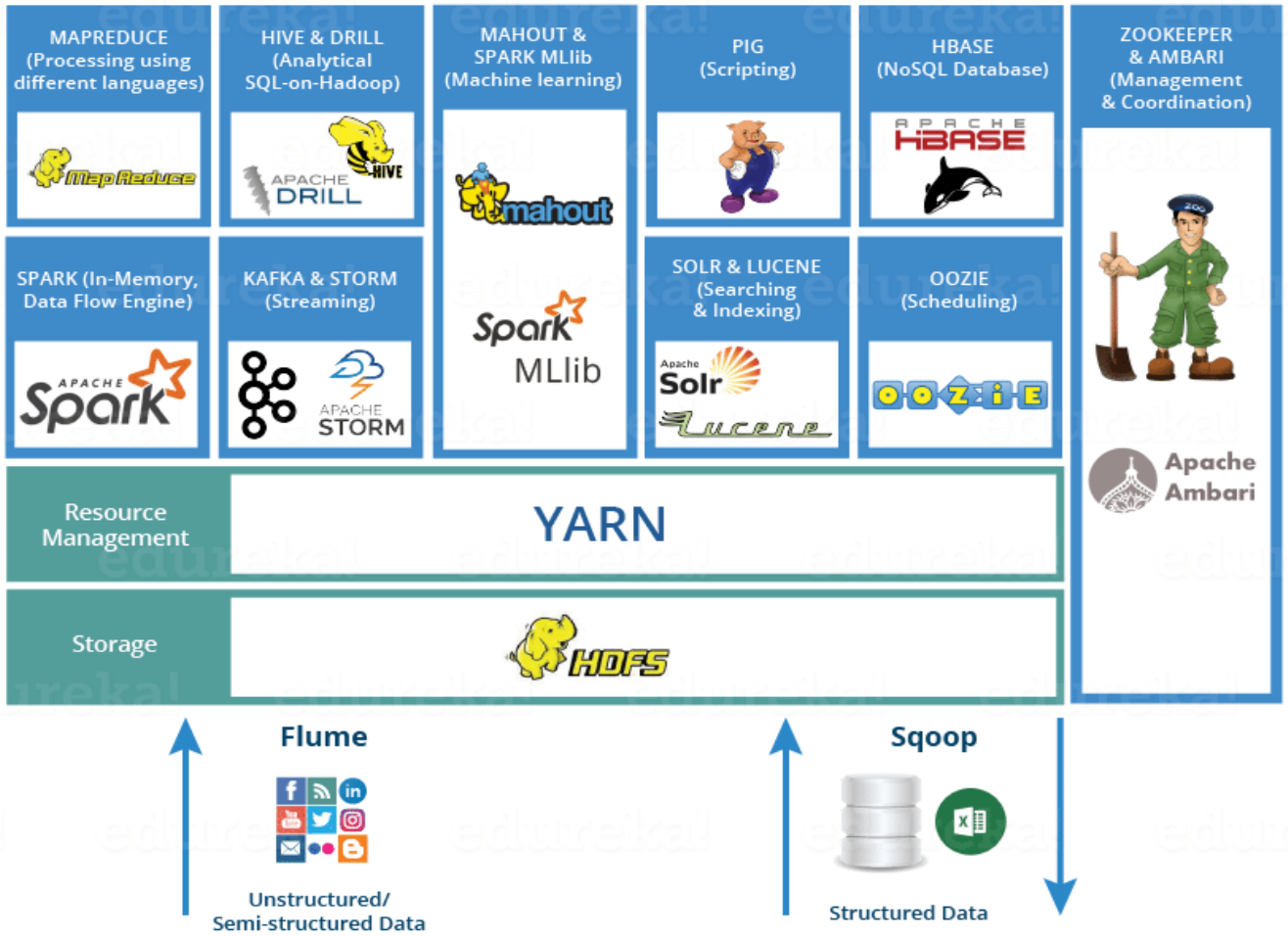
Introduction to Basics of Big Data

Hadoop Ecosystem



HDFS -> Hadoop Distributed File System
YARN -> Yet Another Resource Negotiator
MapReduce -> Data processing using programming
Spark -> In-memory Data Processing
PIG, HIVE-> Data Processing Services using Query (SQL-like)
HBase -> NoSQL Database

Mahout, Spark MLlib -> Machine Learning
Apache Drill -> SQL on Hadoop
Zookeeper -> Managing Cluster
Oozie -> Job Scheduling
Flume, Sqoop -> Data Ingesting Services
Solr & Lucene -> Searching & Indexing
Ambari -> Provision, Monitor and Maintain cluster

Introduction to Map Reduce

What is MapReduce?

- MapReduce is a programming model Google has used successfully is processing its “big-data” sets (~ 20000 peta bytes per day)
  - Users specify the computation in terms of a map and a reduce function,
  - Underlying runtime system automatically parallelizes the computation across large-scale clusters of machines, clusters of commodity hardware in a reliable manner.
  - A processing technique and a program model for distributed computing based on java.
  - Underlying system also handles machine failures, efficient communications, and performance issues.

-- Reference: Dean, J. and Ghemawat, S. 2008. **MapReduce: simplified data processing on large clusters.** *Communication of ACM* 51, 1 (Jan. 2008), 107-113.

How MapReduce Works?

- The whole process goes through four phases of execution namely, splitting, mapping, shuffling, and reducing.
- Consider you have following input data for your Map Reduce Program

Input Data

Welcome to Hadoop Class
Hadoop is good
Hadoop is bad

The final output of the MapReduce task is

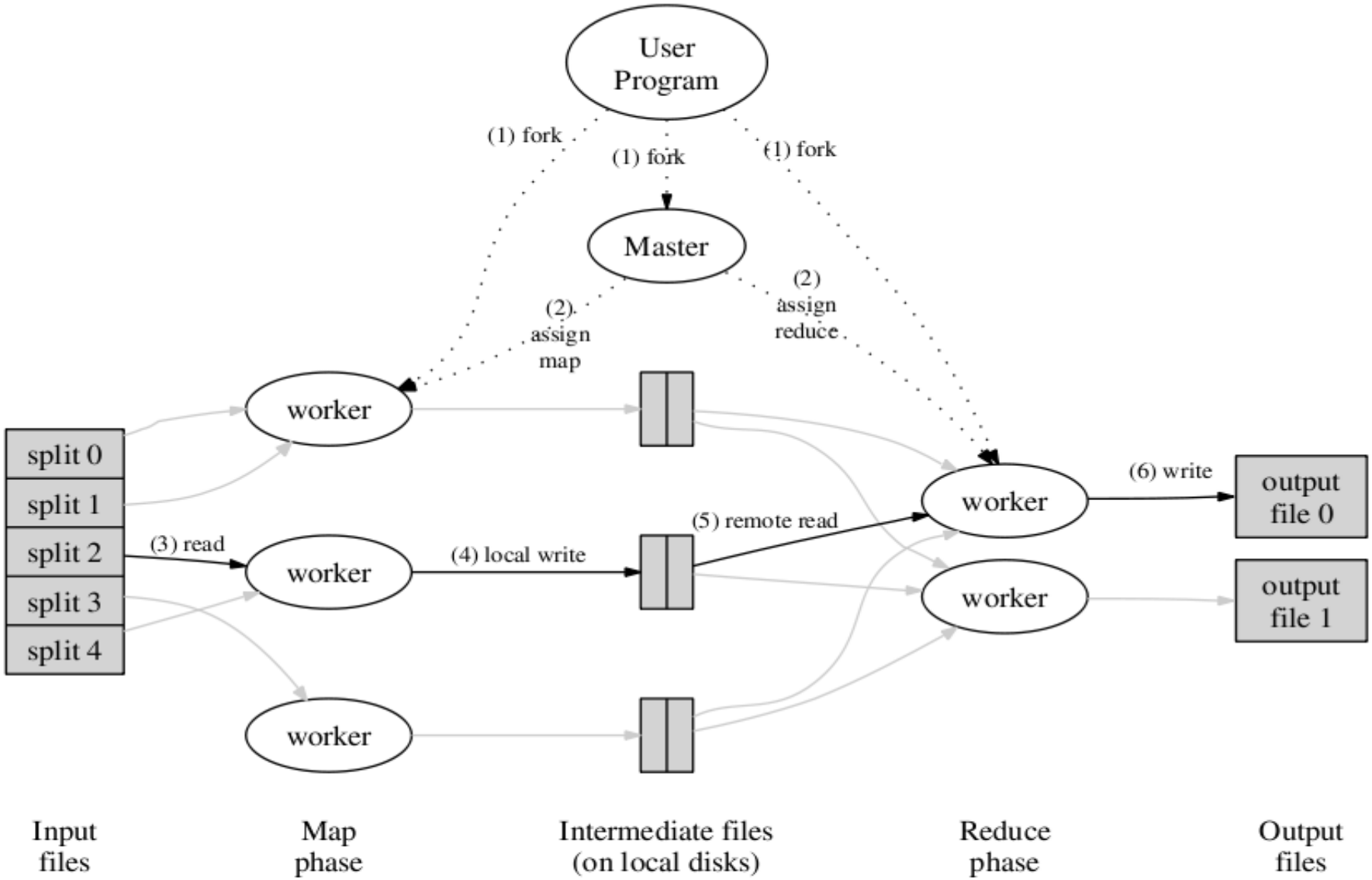
bad	1
Class	1
good	1
Hadoop	3
is	2
to	1
Welcome	1

MapReduce Architecture

- One map task (map function) is created for each split.
- It is always beneficial to have multiple splits because the time taken to process a split is small as compared to the time taken for processing of the whole input.
- When the splits are smaller, the processing is better to load balanced since we are processing the splits in parallel.
- However, it is also not desirable to have splits too small in size. When splits are too small, the overload of managing the splits and map task creation begins to dominate the total job execution time.
- For most jobs, it is better to make a split size equal to the size of an HDFS block (which is 64 MB, by default).

MapReduce Characteristics

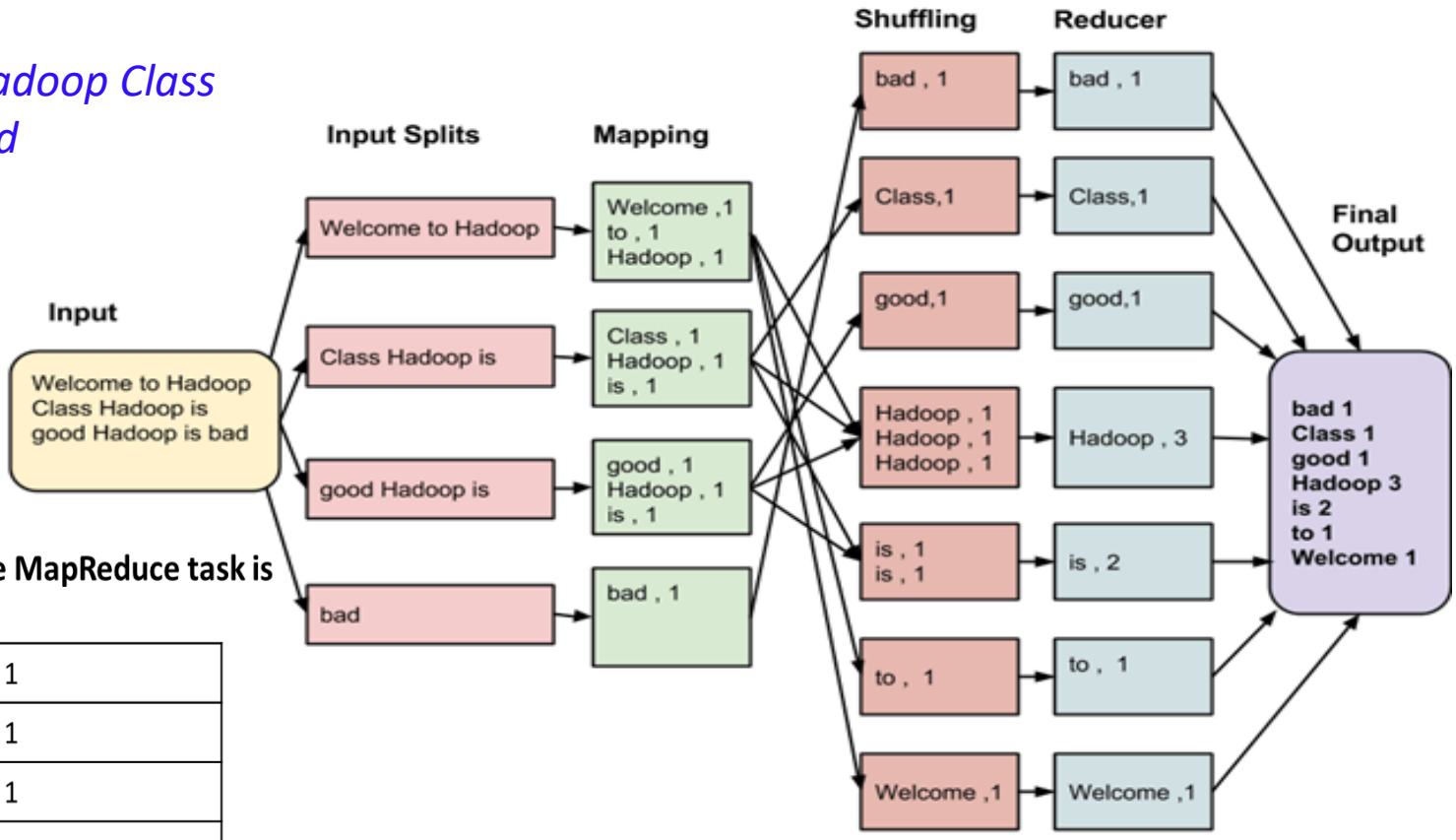
- Very large scale data: peta, exa bytes
- Write once and read many data: allows for parallelism without mutexes
- All the map should be completed before reduce operation starts.
- Map and reduce operations are typically performed by the same physical processor.
- Number of map tasks and reduce tasks are configurable.
- Commodity hardware and storage.
- Runtime takes care of splitting and moving data for operations.
- Special distributed file system. Example: Hadoop Distributed File System and Hadoop Runtime.


User Program
(1) fork
(1) fork
(1) fork
Master
(2) assign map
(2) assign reduce
worker
split 0
split 1
split 2
split 3
split 4
(3) read
worker
(4) local write
worker
(5) remote read
worker
worker
(6) write
output file 0
output file 1
Input files
Map phase
Intermediate files (on local disks)
Reduce phase
Output files

MapReduce Architecture

Input Data

Welcome to Hadoop Class
Hadoop is good
Hadoop is bad

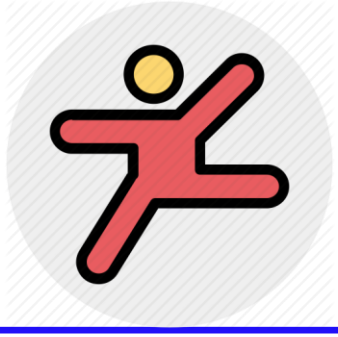


The final output of the MapReduce task is

bad	1
Class	1
good	1
Hadoop	3
is	2
to	1
Welcome	1

MapReduce Phases

- **Input Splits**
  - Job is divided into fixed-size pieces called input splits
- **Mapping**
  - In this phase data in each split is passed to a mapping function to produce output values. In our example, a job of mapping phase is to count a number of occurrences of each word from input splits
- **Shuffling**
  - This phase consumes the output of Mapping phase. Its task is to consolidate the relevant records from Mapping phase output. In our example, the same words are clubed together along with their respective frequency.
- **Reducing**
  - In this phase, output values from the Shuffling phase are aggregated. This phase combines values from Shuffling phase and returns a single output value. In short, this phase summarizes the complete dataset.



Hands-on Exercise

Working with Hadoop

Map Reduce Programing

1. Word count application
2. Youtube Video Rating Application
3. Average Salary of Male and Female Employees
4. Sales Review

Introduction to Hadoop Distributed file system (HDFS)

DFS or Distributed File System

- Distributed File System talks about managing data, i.e. files or folders across multiple computers or servers.
- In other words, DFS is a file system that allows us to store data over multiple nodes or machines in a cluster and allows multiple users to access data.
- The only difference is that, in case of Distributed File System, you store data in multiple machines rather than single machine.
- Even though the files are stored across the network, DFS organizes, and displays data in such a manner that a user sitting on a machine will feel like all the data is stored in that very machine.

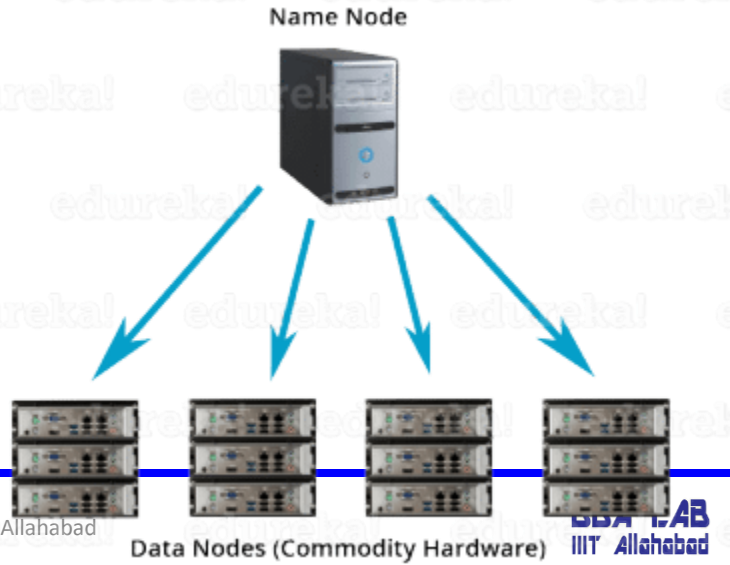
What is HDFS

- **Hadoop Distributed File System** or HDFS is a Java based distributed file system that allows you to store large data across multiple nodes in a Hadoop cluster.
- So, if you install Hadoop, you get HDFS as an underlying storage system for storing the data in the distributed environment.
- Example:
  - Imagine that you have ten machines or ten computers with a hard drive of 1 TB on each machine.
  - Now, HDFS says that if you install Hadoop as a platform on top of these ten machines, you will get HDFS as a storage service.
  - Hadoop Distributed File System is distributed in such a way that every machine contributes their individual storage for storing any kind of data.

Advantages of HDFS

Distributed Storage:

When you access Hadoop Distributed file system from any of the ten machines in the Hadoop cluster, you will feel as if you have logged into a single large machine which has a storage capacity of 10 TB (total storage over ten machines). What does it mean? It means that you can store a single large file of 10 TB which will be distributed over the ten machines (1 TB each). So, it is **not limited to the physical boundaries** of each individual machine.



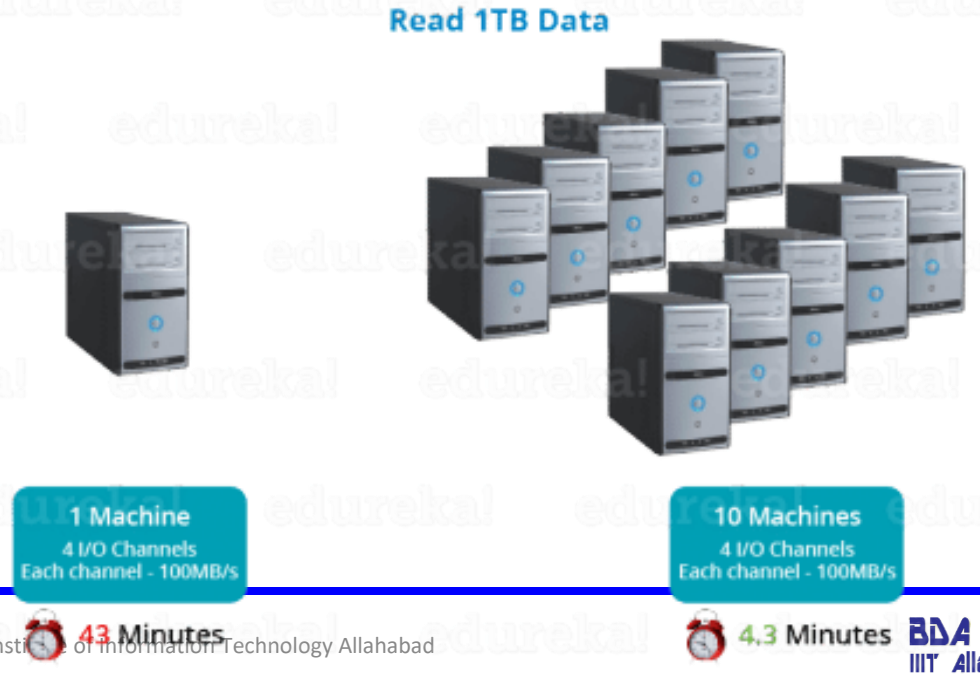
Advantages of HDFS

Distributed & Parallel Computation

- Because the data is divided across the machines, it allows us to take advantage of **Distributed and Parallel Computation**.
- Suppose, it takes 43 minutes to process 1 TB file on a single machine.
- How much time will it take to process the same 1 TB file when you have 10 machines in a Hadoop cluster with similar configuration –

  43 minutes or 4.3 minutes?

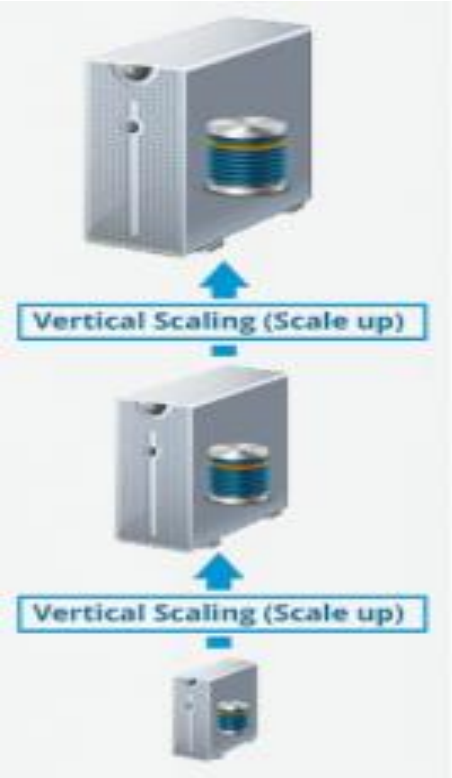
  4.3 minutes



Advantages of HDFS

Scalability

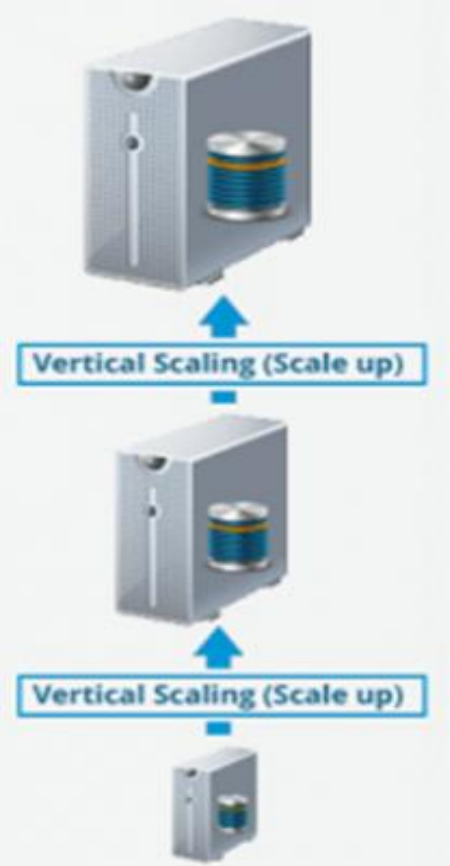
- There are two types of scaling: **vertical** and **horizontal.**
- In **vertical scaling (scale up),** you increase the hardware capacity of your system. In other words, you procure more RAM or CPU and add it to your existing system to make it more robust and powerful. But there are challenges associated with vertical scaling or scaling up:

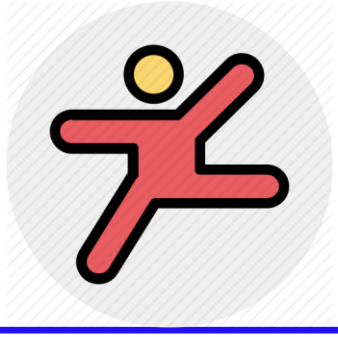


There is always a limit to which you can increase your hardware capacity. So, you can't keep on increasing the RAM or CPU of the machine.

Advantages of HDFS

- In case of **horizontal scaling (scale out),** you add more nodes to existing cluster instead of increasing the hardware capacity of individual machines. And most importantly, you can **add more machines on the go** i.e. Without stopping the system**.**





Hands-on Exercise

Working with HDFS

1.User Creation

2.Use of SSh Application in cluster

3.How to access HDFS

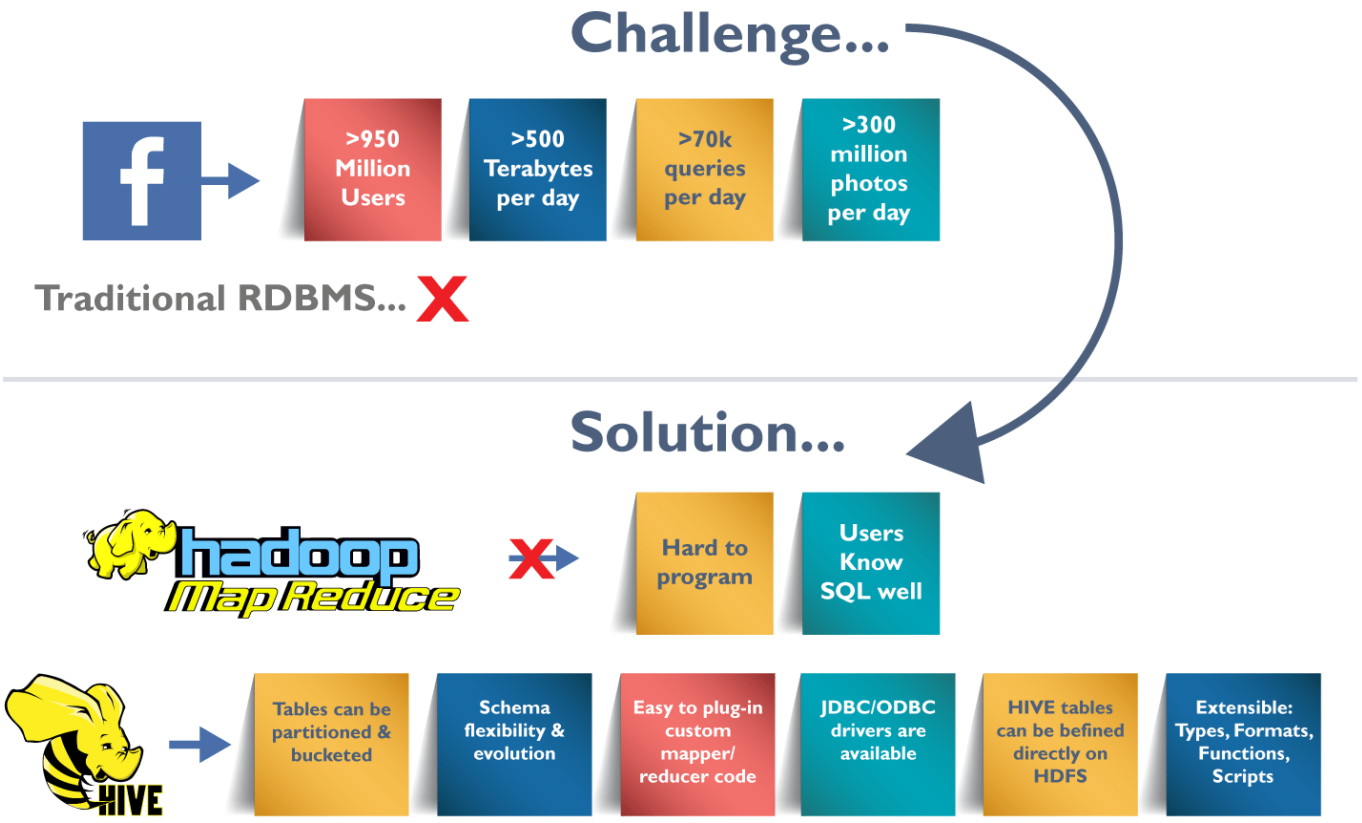
4.Read and create HDFS file

Introduction to Apache HIVE

Apache HIVE

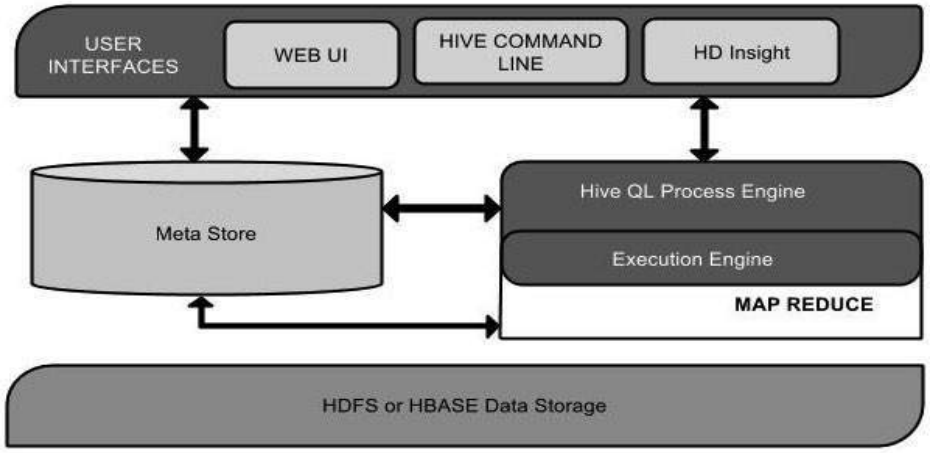
- It is a data warehouse and an ETL tool which provides an SQL-like interface between the user and the Hadoop distributed file system (HDFS) which integrates Hadoop.
- It is built on top of Hadoop.
- It is a software project that provides data query and analysis.
- It facilitates reading, writing and handling wide datasets that stored in distributed storage and queried by Structure Query Language (SQL) syntax.
- It is frequently used for data warehousing tasks like data encapsulation, Ad-hoc Queries, and analysis of huge datasets.
- It is designed to enhance scalability, extensibility, performance, fault-tolerance and loose-coupling with its input formats.

Story of Hive – from Facebook to Apache



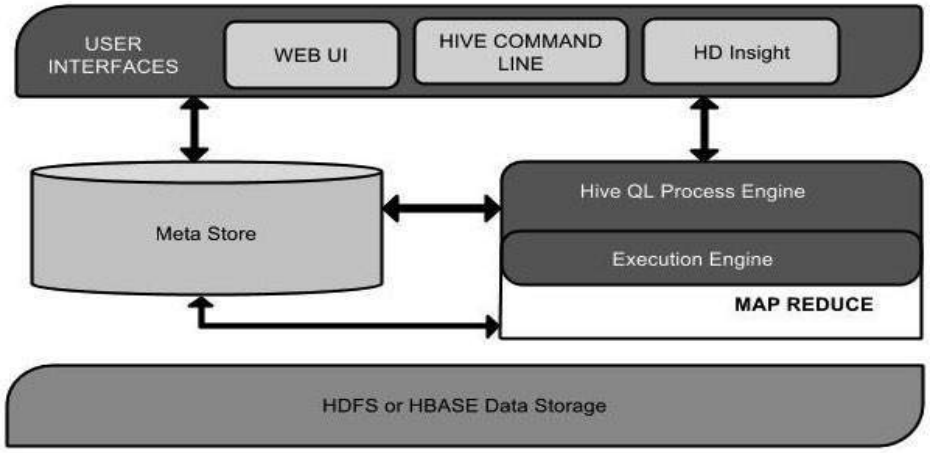
Apache HIVE Architecture

Unit Name	Operation
User Interface	Hive is a data warehouse infrastructure software that can create interaction between user and HDFS. The user interfaces that Hive supports are Hive Web UI, Hive command line, and Hive HD Insight (In Windows server).
Meta Store	Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping.



Apache HIVE Architecture

HiveQL Process Engine	HiveQL is similar to SQL for querying on schema info on 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 conjunction part of HiveQL process Engine and MapReduce is Hive Execution Engine. Execution engine processes the query and generates results as same as MapReduce results. It uses the flavor of MapReduce.
HDFS or HBASE	Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.



Modes of Hive

- **Local Mode –**
  - It is used, when the Hadoop is built under pseudo mode which have only one data node, when the data size is smaller in term of restricted to single local machine, and when processing will be faster on smaller datasets existing in the local machine.

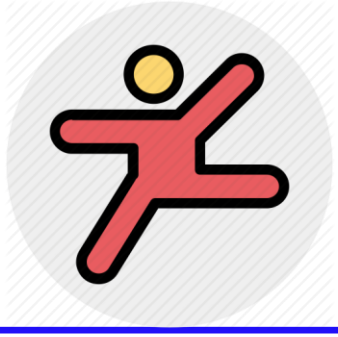
- **Map Reduce Mode –**
  - It is used, when Hadoop is built with multiple data nodes and data is divided across various nodes, it will function on huge datasets and query is executed parallelly, and to achieve enhanced performance in processing large datasets.

Benefits of Apache Hive

- **Ease of use** — Querying data is easy to learn with its SQL-like language.
- **Accelerated initial insertion of data —** Data does not have to be read, parsed, and serialized to a disk in the database's internal format, since Apache Hive reads the schema without checking the table type or schema definition. Compare this to a traditional database where data must be verified each time it is inserted.
- **Superior scalability, flexibility, and cost efficiency** — Apache Hive stores 100s of petabytes of data, since it stores data in the HDFS, making it a much more scalable solution than a traditional database. As a cloud-based Hadoop service, Apache Hive enables users to rapidly spin virtual servers up or down to accommodate fluctuating workloads.
- **Streamlined security** — Critical workloads can be replicated for disaster recovery.
- **Low overhead** — Insert-only tables have near-zero overhead. Since there is no renaming required, the solution is cloud friendly.
- **Exceptional working capacity** — Huge datasets support up to 100,000 queries/hour.

Limitations of Apache Hive

- Certain standard SQL functions, such as NOT IN, NOT LIKE, and NOT EQUAL, do not exist or require certain workarounds.
- Hive is not made for low-latency, real-time, or near-real-time querying.
- SQL queries translate to MapReduce, which means slower performance for certain queries compared to traditional RDBMS.



Hands-on Exercise

Woking with Hive

1.Create Table

2.Insert values into table

3.Other DDL and DML commands

Introduction to Flume

Apache Flume

- Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data.
- It has a simple and flexible architecture based on streaming data flows.
- It is robust and fault tolerant with tunable reliability mechanisms and many failover and recovery mechanisms.
- It uses a simple extensible data model that allows for online analytic application.

Apache Flume with HDFS

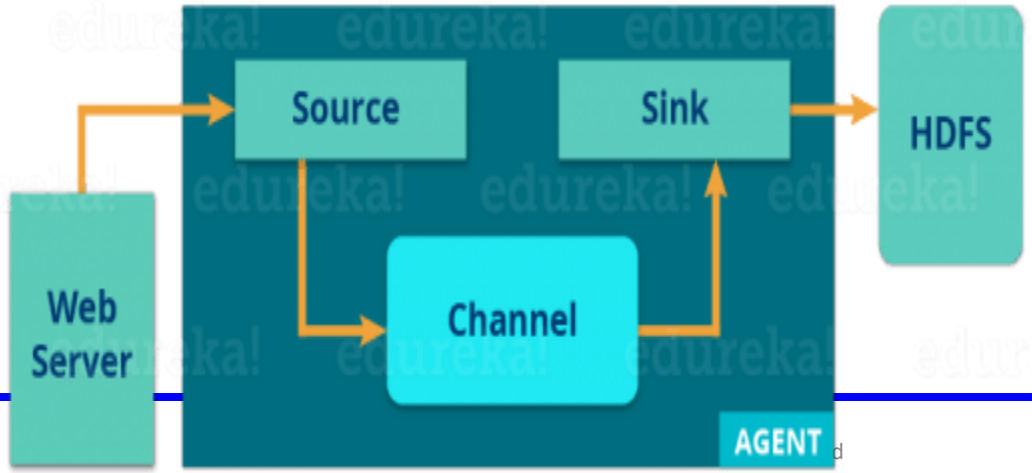
- Apache Flume is a tool for data ingestion in HDFS.
- It collects, aggregates and transports large amount of streaming data such as log files, events from various sources like network traffic, social media, email messages etc. to HDFS.
- The main idea behind the Flume's design is to capture streaming data from various web servers to HDFS. It has simple and flexible architecture based on streaming data flows.

Apache Flume with HDFS

- Apache Flume is a tool for data ingestion in HDFS.
- It collects, aggregates and transports large amount of streaming data such as log files, events from various sources like network traffic, social media, email messages etc. to HDFS.
- The main idea behind the Flume's design is to capture streaming data from various web servers to HDFS. It has simple and flexible architecture based on streaming data flows.

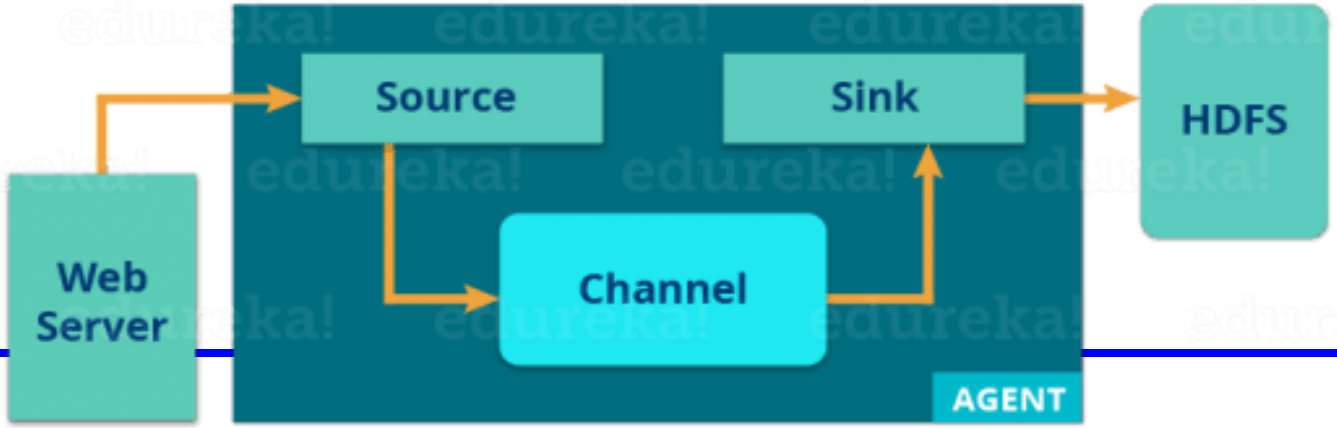
Flume Architecture

- There is a Flume agent which ingests the streaming data from various data sources to HDFS. From the diagram, you can easily understand that the web server indicates the data source. Twitter is among one of the famous sources for streaming data.
- The flume agent has 3 components: source, sink and channel.



Flume Architecture

- **Source:** It accepts the data from the incoming streamline and stores the data in the channel.
- **Channel**: In general, the reading speed is faster than the writing speed. Thus, we need some buffer to match the read & write speed difference. Basically, the buffer acts as a intermediary storage that stores the data being transferred temporarily and therefore prevents data loss. Similarly, channel acts as the local storage or a temporary storage between the source of data and persistent data in the HDFS.
- **Sink:** Then, our last component i.e. Sink, collects the data from the channel and commits or writes the data in the HDFS permanently.

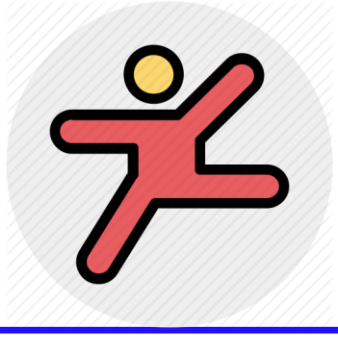


Advantages of Apache Flume

- Flume is scalable, reliable, fault tolerant and customizable for different sources and sinks.
- Apache Flume can store data in centralized stores (i.e data is supplied from a single store) like HBase & HDFS.
- Flume is horizontally scalable.
- If the read rate exceeds the write rate, Flume provides a steady flow of data between read and write operations.
- Flume provides reliable message delivery.
- The transactions in Flume are channel-based where two transactions (one sender & one receiver) are maintained for each message.

Advantages of Apache Flume

- Using Flume, we can ingest data from multiple servers into Hadoop.
- It gives us a solution which is reliable and distributed and helps us in collecting, aggregating and moving large amount of data sets like Facebook, Twitter and e-commerce websites.
- It helps us to ingest online streaming data from various sources like network traffic, social media, email messages, log files etc. in HDFS.
- It supports a large set of sources and destinations types.
- The architecture is one which is empowering Apache Flume with these benefits. Now, as we know the advantages of Apache Flume, lets move ahead and understand Apache Flume architecture.



Hands-on Exercise

Woking with Flume

- **Installation of flume**
- **Demonstration of telnet communication via console**
- **Application of flume on HDFS**

Introduction to Apache Mahout

Apache Mahout

- Apache Mahout is a machine-learning and data mining library. It provides three core features forprocessing large data sets.
  - **Clustering** is the ability to identify related documents to each other based on the content of each document.
  - **Classification** is the ability to categorize a document into an existing category based on the content of the document.
  - **Collaborative filtering** is the ability to provide filters that are based on the similarities and differences of tastes between users.

Mahout Use Cases

- Yahoo: Spam Detection
- Foursquare: Recommendations
- SpeedDate.com: Recommendations
- Adobe: User Targetting
- Amazon: Personalization Platform

Use case Example

- Predict what the user likes based on
  - His/Her historical behavior
  - Aggregate behavior of people similar to him

Customers Who Bought This Item Also Bought

Pattern Recognition and Machine Learning... by Christopher M. Bishop

(50)

$76.10

The Elements of Statistical Learning: Data Minin... by Trevor Hastie

(38)

$71.96

Pattern Classification (2nd Edition) by Richard O. Duda

(29)

$88.52

Features of Apache Mahout

- The algorithms of Mahout are written on top of Hadoop, so it works well in distributed environment. Mahout uses the Apache Hadoop library to scale effectively in the cloud.
- Mahout offers the coder a ready-to-use framework for doing data mining tasks on large volumes of data.
- Mahout lets applications to analyze large sets of data effectively and in quick time.
- Includes several MapReduce enabled clustering implementations such as kmeans, fuzzy k-means, Canopy, Dirichlet, and Mean-Shift.
- Supports Distributed Naive Bayes and Complementary Naive Bayes classification implementations.
- Comes with distributed fitness function capabilities for evolutionary programming.
- Includes matrix and vector libraries

Applications of Mahout

- Companies such as Adobe, Facebook, LinkedIn, Foursquare, Twitter, and Yahoo use Mahout internally.
- Foursquare helps you in finding out places, food, and entertainment available in a particular area. It uses the recommender engine of Mahout.
- Twitter uses Mahout for user interest modelling.
- Yahoo! uses Mahout for pattern mining.

Introduction to Elasticsearch

Basics of Elasticsearch

elasticsearch

Elasticsearch

- In today's IT world, a voluminous amount of data sizing approx 2.5 Quintillion bytes is generated every day.
- This data majorly comes from different sources, for example, social media sites, video sharing sites, and medium to large-scale organizations.
- This data is referred as data ocean or in more general terms called the Big Data.
- A considerable part of this data is insignificant, unstructured and scattered when it's alone.
- To make sense out of it you need analytic tools.

Elasticsearch

- Elasticsearch is a highly scalable open-source full-text search and analytics engine.
- It allows you to store, search, and analyze big volumes of data quickly and in near real time.
- It is generally used as the underlying engine/technology that powers applications that have complex search features and requirements.
- It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents.

Advantages of Elasticsearch

- **Scalability:** Elasticsearch is very easy to scale and reliable as well. It is a very important feature which helps to simplify the complex architectures and save time during the implementation of projects.
- **Speed:** Elasticsearch uses distributed inverted indices to find the best matches for your full-text searches. This makes it really fast even when searching from very large data sets.
- **Easy to use API**: Elasticsearch provides simple RESTful APIs and uses schema-free JSON documents which makes indexing, searching, and querying the data really easy.
- **Multilingual:** One of the most distinct features Elasticsearch has is, it is multilingual. It supports a wide variety of documents written in different languages like Arabic, Brazilian, Chinese, English, French, Hindi, Korean etc.

Advantages of Elasticsearch

- **Document-Oriented:** Elasticsearch stores real-world complex entities as structured JSON documents and indexes all fields by default to make the data searchable. Since there are no rows and columns of data, you can perform complex full-text search easily.
- **Auto-completion:** Elasticsearch also provides autocompletion functionality. By predicting the word using very few characters, autocompletion speeds up human-computer interaction.
- **Schema-Free:** Elasticsearch is schema-free as it accepts JSON documents. It tries to detect the data structure, index the data and thus makes the data searchable

Introduction to Kibana

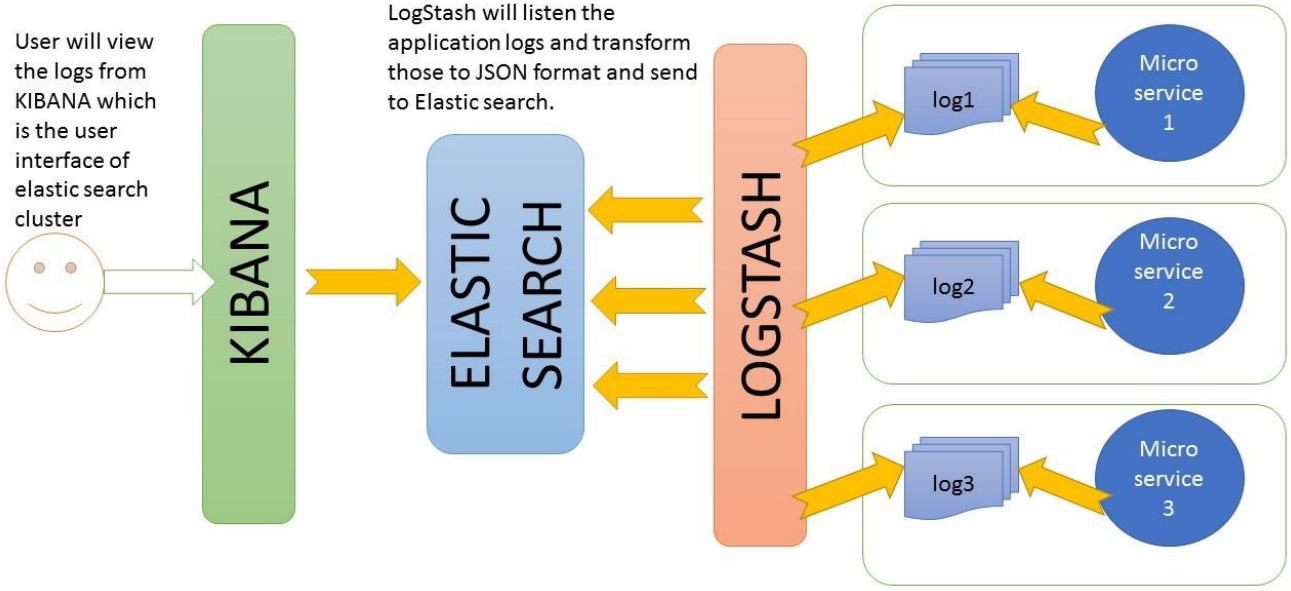
Basics of Kibana

Kibana

- Kibana is an open source analytics and visualization platform designed to work with Elasticsearch.
- Kibana can be used to search, view, and interact with data stored in Elasticsearch indices.
- We can easily perform advanced data analysis and visualize our data in a variety of charts, tables, and maps.
- Kibana makes it easy to understand large volumes of data. Its simple, browser-based interface enables you to quickly create and share dynamic dashboards that display changes to Elasticsearch queries in real time.
- Setting up Kibana is a snap. You can install Kibana and start exploring your Elasticsearch indices in minutes — no code, no additional infrastructure required.

Kibana

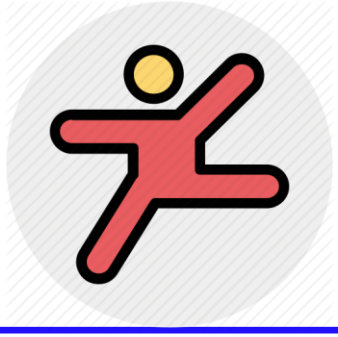
- Kibana also provides a presentation tool, referred to as Canvas, that allows users to create slide decks that pull live data directly from Elasticsearch.
- The combination of Elasticsearch, Logstash, and Kibana, referred to as the "Elastic Stack" (formerly the "ELK stack"), is available as a produ



ELK stack interaction with different applications based on Log file

Kibana Benefits

- **INTERACTIVE CHARTS**
  - Kibana offers intuitive charts and reports that you can use to interactively navigate through large amounts of log data. You can dynamically drag time windows, zoom in and out of specific data subsets, and drill down on reports to extract actionable insights from your data.
- **MAPPING SUPPORT**
  - Kibana comes with powerful geospatial capabilities so you can seamlessly layer in geographical information on top of your data and visualize results on maps.
- **PRE-BUILT AGGREGATIONS AND FILTERS**
  - Using Kibana's pre-built aggregations and filters, you can run a variety of analytics like histograms, top-N queries, and trends with just a few clicks.
- **EASILY ACCESSIBLE DASHBOARDS**
  - You can easily set up dashboards and reports and share them with others. All you need is a browser to view and explore the data.



Hands-on Exercise

1. Request to Elasticsearch and response to Kibana
2. Use of post and get methods
3. Search in database
4. Update
5. Bulk insertion

Introduction to Apache Kafka

Introduction to Apache Kafka

- Kafka® is used for building real-time data pipelines and streaming apps.
- It is horizontally scalable, fault-tolerant, wicked fast, and runs in production in thousands of companies.
- A streaming platform has three key capabilities:
  - Publish and subscribe to streams of records, similar to a message queue or enterprise messaging system.
  - Store streams of records in a fault-tolerant durable way.
  - Process streams of records as they occur

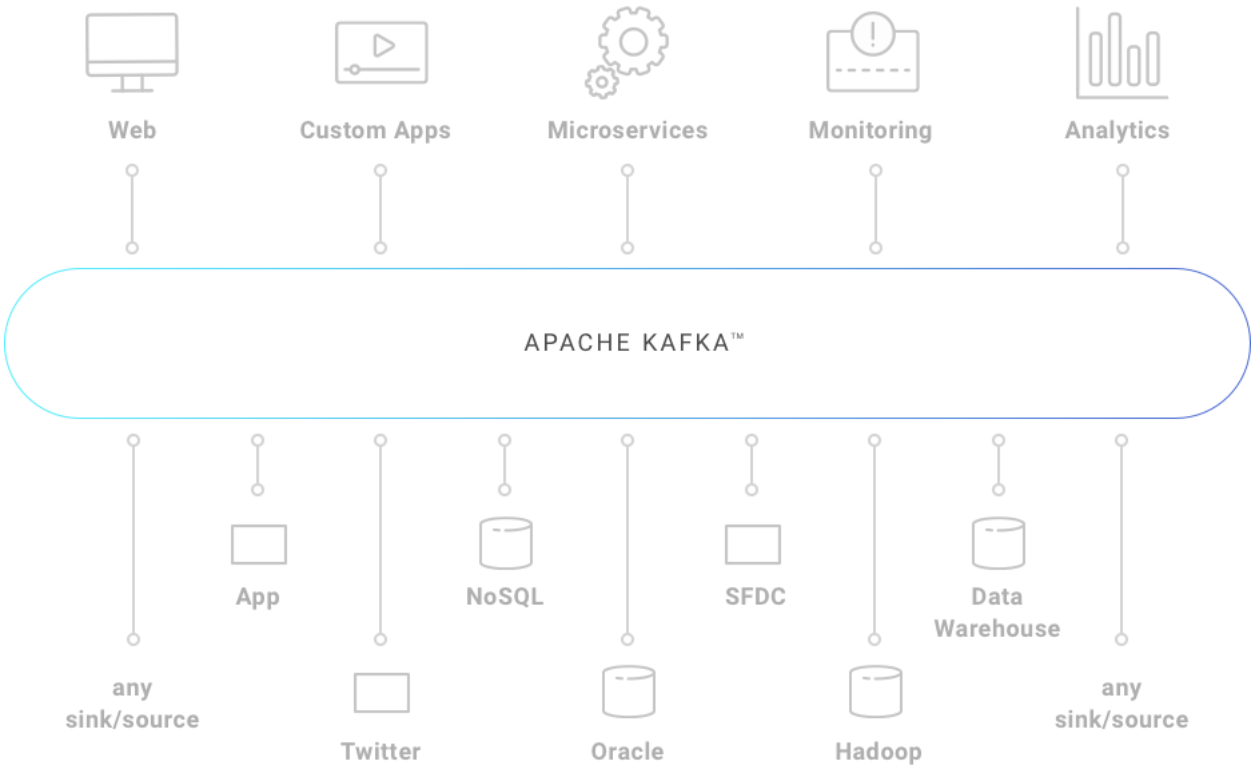
Introduction to Apache Kafka

- It is an apache project initially developed at LinkedIn
  - Distributed publish-subscribe messaging system
  - Designed for processing of real time activity stream data e.g. logs, metrics collections
  - Written in Scala
- Features
  - – Persistent messaging
  - – High-throughput
  - – Supports both queue and topic semantics
  - – Uses Zookeeper for forming a cluster of nodes (producer/consumer/broker)
  - and many more…
  - http://kafka.apache.org/

Introduction to Apache Kafka

- Kafka is generally used for two broad classes of applications:
  - Building real-time streaming data pipelines that reliably get data between systems or applications
  - Building real-time streaming applications that transform or react to the streams of data
- First a few concepts:
  - Kafka is run as a cluster on one or more servers that can span multiple datacenters.
  - The Kafka cluster stores streams of *records* in categories called *topics*.
  - Each record consists of a key, a value, and a timestamp.

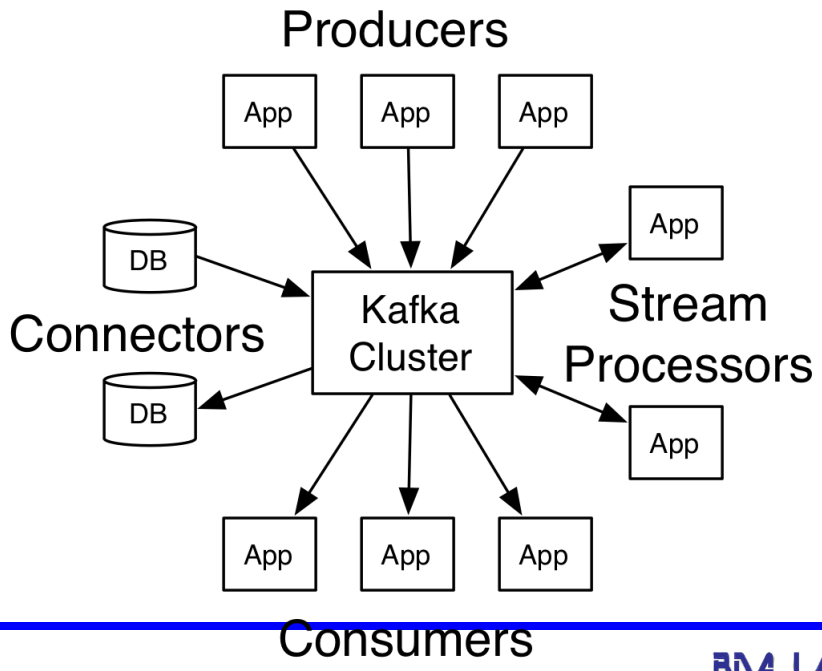
Where Apache Kafka Fits In



Introduction to Apache Kafka

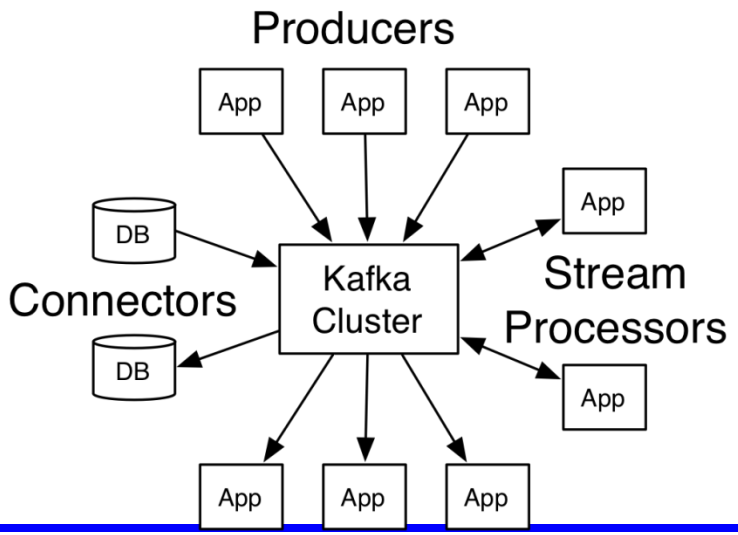
Kafka has four core APIs:

- **The Producer API** allows an application to publish a stream of records to one or more Kafka topics.
- **The Consumer API** allows an application to subscribe to one or more topics and process the stream of records produced to them.



Introduction to Apache Kafka

- **The Streams API** allows an application to act as a stream processor, consuming an input stream from one or more topics and producing an output stream to one or more output topics, effectively transforming the input streams to output streams.
- **The Connector API** allows building and running reusable producers or consumers that connect Kafka topics to existing applications or data systems. For example, a connector to a relational database might capture every change to a table.



How Can Apache Kafka Help You?

- **Publish + Subscribe**
  - Using its immutable commit log, you can subscribe to it, and publish data to any number of systems or real-time applications.
  - Unlike messaging queues, Kafka is a highly scalable, fault tolerant distributed system, allowing it to be deployed for applications like managing passenger and driver matching at Uber.
  - It is providing real-time analytics and predictive maintenance for British Gas' smart home, and performing numerous real-time services across all of LinkedIn.
  - This unique performance makes it perfect to scale from one app to company-wide use.

How Can Apache Kafka Help You?

Store

- An abstraction of a distributed commit log commonly found in distributed databases, Apache Kafka provides durable storage.
- Kafka can act as a ‘source of truth’, being able to distribute data across multiple nodes for a highly available deployment within a single data center or across multiple availability zones.

How Can Apache Kafka Help You?

Process

- An event streaming platform would not be complete without the ability to manipulate that data as it arrives.
- The Streams API within Apache Kafka is a powerful, lightweight library that allows for on-the-fly processing, letting you aggregate, create windowing parameters, perform joins of data within a stream, and more.
- Perhaps best of all, it is built as a Java application on top of Kafka, keeping your workflow intact with no extra clusters to maintain.
- Kafka is very fast, performs 2 million writes/sec.

Introduction to Apache Zookeeper

Apache ZooKeeper™

What is a Distributed System

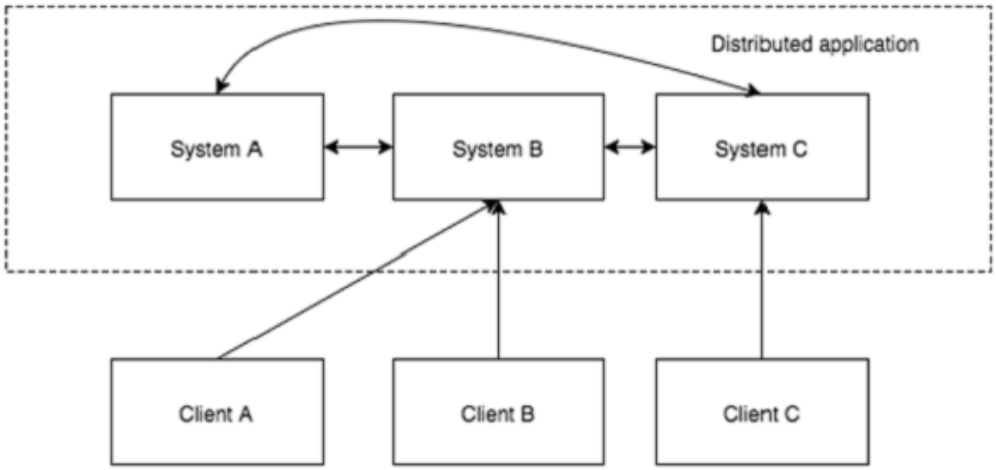
- “A Distributed system consists of multiple computers that communicate and coordinate their actions by passing messages. The components interact with each other in order to achieve a common goal. ”

  - Wikipedia

- Normally, complex and time-consuming tasks, which will take hours to complete by a non-distributed application (running in a single system) can be done in minutes by a distributed application by using computing capabilities of all the system involved.

What is a Distributed System

- A distributed application has two parts, Server and Client application. Server applications are actually distributed and have a common interface so that clients can connect to any server in the cluster and get the same result.
- Client applications are the tools to interact with a distributed application.



Benefits of Distributed Applications

- **Reliability** – Failure of a single or a few systems does not make the whole system to fail.
- **Scalability** – Performance can be increased as and when needed by adding more machines with minor change in the configuration of the application with no downtime.
- **Transparency** – Hides the complexity of the system and shows itself as a single entity / application.

Challenges of Distributed Applications

- **Race condition** – Two or more machines trying to perform a particular task, which actually needs to be done only by a single machine at any given time. For example, shared resources should only be modified by a single machine at any given time.
- **Deadlock** – Two or more operations waiting for each other to complete indefinitely.
- **Inconsistency** – Partial failure of data.

What is Apache ZooKeeper Meant For?

- Apache ZooKeeper is a service used by a cluster (group of nodes) to coordinate between themselves and maintain shared data with robust synchronization techniques.
- ZooKeeper is itself a distributed application providing services for writing a distributed application.
- The common services provided by ZooKeeper are as follows −
  - **Naming service –** Identifying the nodes in a cluster by name. It is similar to DNS, but for nodes.
  - **Configuration management –** Latest and up-to-date configuration information of the system for a joining node.
  - **Cluster management –** Joining / leaving of a node in a cluster and node status at real time.
  - **Leader election –** Electing a node as leader for coordination purpose.
  - **Locking and synchronization service –** Locking the data while modifying it. This mechanism helps you in automatic fail recovery while connecting other distributed applications like Apache HBase.

Benefits of ZooKeeper

- Simple distributed coordination process- Race condition and deadlock are handled using fail-safe synchronization approach. Another main drawback is inconsistency of data, which ZooKeeper resolves with atomicity.
- Synchronization − Mutual exclusion and co-operation between server processes. This process helps in Apache HBase for configuration management.
- Serialization − Encode the data according to specific rules. Ensure your application runs consistently. This approach can be used in MapReduce to coordinate queue to execute running threads.
- Atomicity − Data transfer either succeed or fail completely, but no transaction is partial. Inconsistency of data, can also be resolved using Zookeeper.

Why is Zookeeper necessary for Apache Kafka?

- **Controller election**
  - Whenever a node shuts down, a new controller can be elected and it can also be made sure that at any given time, there is only one controller and all the follower nodes have agreed on that.
- **Configuration of Topics**
  - The configuration regarding all the topics including the list of existing topics, the number of partitions for each topic, the location of all the replicas, list of configuration overrides for all topics and which node is the preferred leader, etc.
- **Access control lists**
  - Access control lists or ACLs for all the topics are also maintained within Zookeeper.
- **Membership of the cluster**
  - Zookeeper also maintains a list of all the brokers that are functioning at any given moment and are a part of the cluster.

References

- Official Websites of Apache Software

BDA TASK 1

Background

- **Primary data** refers to the original data collected directly from first-hand sources through methods such as surveys, experiments, observations, or interviews.
- This data is gathered with a specific purpose in mind and is tailored to address particular research questions or objectives.
- It is typically raw, unprocessed, and highly relevant to the researcher's needs, offering direct insights into the study topic.

Background

- In contrast, **secondary data** is derived from existing sources and involves the repurposing of data that has already been collected for different research or organizational purposes.
- This data can be sourced from research articles, public databases, government reports, or previously conducted studies. While secondary data is often more accessible and cost-effective than primary data, it may require substantial cleaning, transformation, and augmentation to align with new research goals.
- Together, **primary and secondary data** play crucial roles in research, providing a foundation for generating comprehensive and insightful analyses.

Objective

- The primary objective of this assignment is to equip students with a comprehensive understanding of the process of generating large, high-quality datasets.
- Students will learn to apply various **data augmentation** and **fusion techniques**, ensuring data integrity through meticulous cleaning and preprocessing.
- Additionally, students will perform preliminary data analysis to extract valuable insights.
- This assignment aims to provide practical experience in handling Big Data, enhancing students' skills in data manipulation, analysis, and real-world application, ultimately preparing them for advanced research and Big Data Analytics roles.

TASK 1: Primary Dataset Selection:

Specify the type of your dataset as one of the following:

- **Unimodal:** Contains a single type of data (e.g., Text, Image, Audio, or Video).
- **Multimodal:** Combines multiple types of data (e.g., a combination of Text, Image, Audio, and Video).
- **Note that more weightage will be given to multimodal datasets due to their complexity and the breadth of analysis they allow.**

TASK 1: Primary Dataset Selection:

Additionally, classify your dataset based on its structure:

- **Structured**: Data is highly organized and formatted in a fixed schema, such as in tables or databases (e.g., SQL databases, Excel spreadsheets).
- **Unstructured:** Data lacks a pre-defined format or structure, often consisting of text, images, or multimedia content (e.g., social media posts, video recordings).
- **Semi-Structured:** Data that does not have a rigid structure but contains tags or markers to separate elements, offering some level of organization (e.g., JSON files, XML documents).

TASK 2: Secondary Dataset Generation

- Use the primary dataset to generate a secondary dataset.

TASK 3: Data Cleaning and Preprocessing:

- Clean the generated dataset by handling missing values, duplicates, and outliers.
- Normalize or standardize the data if necessary.
- Document the data cleaning steps and justify the choices made.

TASK 4: Synthetic/Secondary Data Validation:

Data Quality Metrics:

- Use metrics such as coverage, completeness, and consistency to validate the quality of synthetic/Secondary data.

Statistical Comparison:

- Compare the statistical properties of synthetic/Secondary data with the original dataset using metrics like distribution similarity.

Model Performance Evaluation:

- Evaluate the performance of machine learning models on synthetic/Secondary data and compare it with performance on the original data.

TASK 5: Exploratory Data Analysis (EDA):

- Perform EDA on the secondary dataset.
- Use visualizations to showcase the distribution of key features, correlations, and any interesting patterns.
- Provide a summary of the key insights obtained from the EDA.

Task 5: Documentation:

- Create a detailed report that includes:
- Introduction and objective of the assignment.
- Description of the primary and secondary datasets.
- Data generation process and techniques used.
- Data cleaning and preprocessing steps.
- Results of the exploratory data analysis.
- Conclusion and potential future work.

TASK 6

Methods for Secondary Data Generation

Data Augmentation:

- Apply techniques such as data duplication with noise, synthetic data generation, or adding new calculated features.
- Ensure to use a variety of augmentation techniques relevant to your data type.

TASK 6

Data Augmentation Types:

Image Data Augmentation:

- Geometric Transformations: Rotation, Flipping, Cropping, Scaling, Translation, Shearing.
- Color and Brightness Adjustments: Brightness Adjustment, Contrast Adjustment, Saturation Adjustment, Hue Adjustment, Grayscale Conversion.
- Noise Injection: Gaussian Noise, Salt-and-Pepper Noise, Speckle Noise.
- Blur and Sharpen: Gaussian Blur, Median Blur, Sharpening Filters.
- Occlusion: Random Erasing, Cutout.
- Distortions: Elastic Transformations, Perspective Transformations.

TASK 6

Data Augmentation Types:

Text Data Augmentation:

- Synonym Replacement, Back Translation, Random Insertion, Random Deletion, Random Swap, Contextual Augmentation, Text Infilling.

Audio Data Augmentation:

- Time Stretching, Pitch Shifting, Adding Noise, Time Shifting, Volume Control, SpecAugment.

Time-Series Data Augmentation:

- Time Warping, Jittering, Permutation, Scaling, Window Slicing, Magnitude Warping, Window Warping.

TASK 6

Data Augmentation Types:

Tabular Data Augmentation:

- SMOTE (Synthetic Minority Over-sampling Technique), Random Noise Injection, Data Synthesis, Permutation, Feature Engineering, Bootstrap Sampling.

Multimodal Data Augmentation:

- Cross-Modal Augmentation, Fusion-Based Augmentation, Independent Augmentation.

TASK 6

Data Fusion:

- **Simple Techniques:**
- **Concatenation:** Combine data from multiple sources by appending them horizontally or vertically.
- **Aggregation**: Summarize and aggregate data from multiple sources based on common attributes.

- **Intermediate Techniques:**
- **Cross-Referencing**: Link datasets using common entities (e.g., customer IDs, geographic locations) to enrich the dataset with additional information.
- **Hierarchical Data Fusion:** Combine datasets at different levels of granularity (e.g., regional data with national data).
- **Metadata Integration:** Add metadata such as timestamps, geographic locations, or sensor information to the primary dataset.

Complex Data preparation Techniques:

- **Data Integration Frameworks:** Use frameworks like Apache NiFi or Talend for systematic data integration and transformation.
- **Canonical Correlation Analysis (CCA):** Find linear relationships between two datasets to combine them in a way that maximizes their correlation.
- **Dimensionality Reduction for Fusion:** Apply dimensionality reduction techniques (e.g., PCA, t-SNE) to each dataset before combining them to reduce complexity.
- **Entity Resolution:** Resolve and match entities (e.g., customers, products) across different datasets to integrate them accurately.
- **Time-Series Alignment:** Align time-series data from different sources based on timestamps to synchronize them.

Complex Data preparation Techniques:

1.Feature Engineering:

- **Embedding Representations:** Use embeddings (e.g., word embeddings, graph embeddings) to create dense feature representations.
- **Temporal Feature Engineering:** Extract temporal features like trend, seasonality, and cyclic patterns from time-series data.
- **Spatial Feature Engineering:** Create spatial features such as distance to important locations, spatial lag, or spatial autocorrelation metrics.

Complex Data preparation Techniques:

1.Data Normalization and Transformation:

- **Box-Cox Transformation:**Apply Box-Cox transformation to make data more normally distributed.
- **Power Transformations:** Use power transformations to stabilize variance and make data more normally distributed.
- **Rank Transformations:**Transform data to ranks to reduce the effect of outliers and non-normal distributions.

Complex Data preparation Techniques:

1.Clustering and Segmentation:

- **Spectral Clustering:** Apply spectral clustering to segment data based on the eigenvalues of a similarity matrix.
- **Affinity Propagation:** Use affinity propagation clustering to identify exemplars and clusters based on message passing between data points.
- **Time-Series Clustering:** Cluster time-series data based on similarity in temporal patterns using dynamic time warping (DTW) or other similarity measures.

Complex Data preparation Techniques:

Dimensionality Reduction:

- **Independent Component Analysis (ICA):** Use ICA to separate a multivariate signal into additive, independent components.
- **Factor Analysis:**Apply factor analysis to model variability among observed variables in terms of fewer unobserved variables called factors.
- **Non-Negative Matrix Factorization (NMF):**Use NMF to reduce dimensionality by factoring data into non-negative matrices.

Complex Data preparation Techniques:

Data Augmentation in NLP:

- **Back Translation:**Translate text to another language and back to the original language to generate paraphrases.
- **Text Infilling:**Randomly remove words from a sentence and use a language model to predict the missing words.
- **Contextual Augmentation:**Replace words with their contextual synonyms using language models like BERT.

Complex Data preparation Techniques:

Graph-Based Techniques:

- **Graph Augmentation:**Add or remove edges and nodes in a graph to create new graph instances.
- **Graph Embeddings:**Use graph embedding techniques to create feature vectors for nodes, edges, or entire graphs.
- **Graph Neural Networks (GNNs):** Apply GNNs to generate new graph structures based on learned representations.

Complex Data preparation Techniques:

Cross-Domain Data Generation:

- **Domain Adaptation:**
  - Use domain adaptation techniques to generate data that is useful for transfer learning between different domains.
- **Cross-Domain Translation:**
  - Translate data from one domain to another using models like CycleGAN.
- **Meta-Learning:**
  - Use meta-learning to generate data that can be used to train models that generalize well across different tasks.

Data Quality and Size Guidelines:

Quality Standards:

- Ensure that the generated data is free of errors, inconsistencies, and irrelevant information.
- Maintain a high level of accuracy in the synthetic data generation process.
- Perform thorough validation to ensure the synthetic data reflects the characteristics and distribution of the primary dataset.

Data Sizes:

- Minimum Size: The secondary dataset should have at least 10,000 samples for unimodal datasets and 5,000 samples per modality for multimodal datasets.
- Variety: Ensure a diverse representation of data types and categories within the dataset.
- Balance: Aim for balanced datasets, especially in classification tasks, to avoid biases.

Deliverables:

- The secondary dataset generated.
- A Jupyter notebook (or similar) containing the code for data generation, cleaning, and EDA.
- A detailed report in PDF format.

Evaluation Criteria:

- **Understanding of the Dataset:** Clear and concise description of the primary dataset and its relevance.
- **Creativity in Data Generation:** Innovative techniques used to create the secondary dataset.
- **Data Cleaning and Preprocessing:** Thoroughness and appropriateness of the cleaning process.
- **Exploratory Data Analysis:** Depth and clarity of the analysis and visualizations.
- **Documentation:** Quality and completeness of the report.
- **Data Quality:** Adherence to the quality standards and size guidelines provided.
- **Grading Scheme:** A+, A, B+, B, C, D, F

Example Topics for Primary Datasets:

- Social Media Data: (e.g., Twitter sentiment analysis)
- Financial Data: (e.g., stock prices, credit card transactions)
- Healthcare Data: (e.g., patient records, medical images)
- Environmental Data: (e.g., climate statistics, pollution levels)
- E-commerce Data: (e.g., sales transactions, customer reviews)

Assignment 1: Important Points

1. Dataset Selection:

- What type of dataset have you chosen for your primary dataset (Unimodal or Multimodal)? Please explain why you selected this type.
- How is your dataset structured (Structured, Semi-Structured, Unstructured)? Provide examples of the data it contains.

2. Data Augmentation Techniques:

- Which data augmentation techniques will you apply to your dataset? Explain how these techniques will enhance your data.
- Describe the methods you will use to ensure the augmented data maintains high quality and relevance.

Assignment 1: Important Points

3. Data Cleaning and Preprocessing:

- What steps will you take to clean your dataset? Describe how you will handle missing values, duplicates, and outliers.
- Will you normalize or standardize your data? Justify your choice.

4. Secondary Data Generation:

- Explain the process you will use to generate a secondary dataset from your primary dataset.
- How will you validate the quality of your secondary dataset? Discuss the metrics you will use.

5. Exploratory Data Analysis (EDA):

- What methods and tools will you use for EDA on your secondary dataset?
- Provide examples of the key features you will analyze and the types of visualizations you plan to create.

6. Documentation:

- What key elements will you include in your final report? Outline the structure of your report.
- How will you document the data cleaning and preprocessing steps? Provide examples.

Thank you