MIT-WPU Dr. Vishwanath Karad MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

CET4001B Big Data Technologies

School of Computer Engineering and Technology

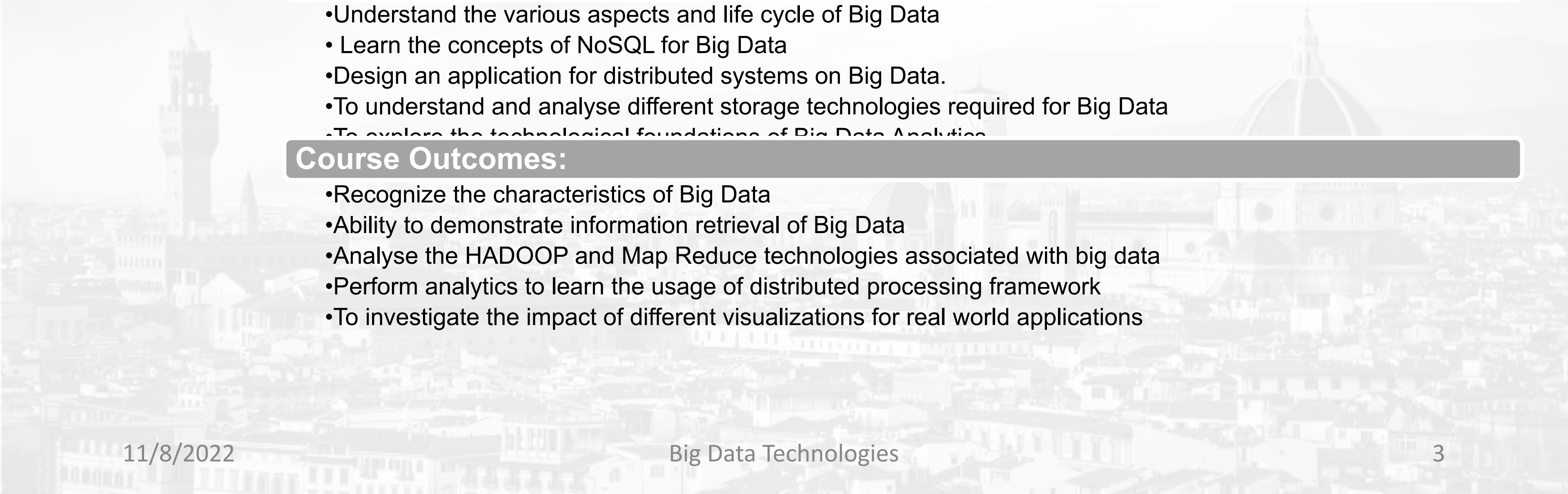
CET4001B Big Data Technologies

Dr. Vishwanath Karad MIT WORLD PEACE UNIVERSITY | PUNE

TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

(Computer Engineering and Technology)
(TYB.Tech)
UNIT IV

CET4001B Big Data Technologies



Teaching Scheme **Credits: 03 + 01**

Theory: 3 Hrs / Week **Practical:** 2Hrs/Week

Course Objectives:

- Understand the various aspects and life cycle of Big Data
- Learn the concepts of NoSQL for Big Data
- Design an application for distributed systems on Big Data.
- To understand and analyse different storage technologies required for Big Data
- To explore the technological foundations of Big Data Analytics

Course Outcomes:

- Recognize the characteristics of Big Data
- Ability to demonstrate information retrieval of Big Data
- Analyse the HADOOP and Map Reduce technologies associated with big data
- Perform analytics to learn the usage of distributed processing framework
- To investigate the impact of different visualizations for real world applications

Unit-IV
Technologies and tools for Big Data

- Importing Relational data with Sqoop
- Injecting stream data with flume.
- Pig:
  - Basic Concepts of Pig
  - Architecture of Pig
- Hive:
  - what is Hive?
  - Architecture of Hive
  - Hive Commands.
- Zookeeper
- Apache Spark ecosystem
  - Overview of Apache Spark Ecosystem,
  - Spark Architecture

Sqoop

- Apache Sqoop is a tool designed to transfer data between Apache Hadoop and relational databases.

- It is a popular tool for data warehousing and data analytics, as it allows users to easily move data from relational databases to Hadoop for processing and storage.

- Apache Sqoop is a tool designed to transfer data between Apache Hadoop and relational databases. It uses a connector architecture to support a wide variety of relational databases, including MySQL, PostgreSQL, Oracle, and SQL Server.

Benefits of Sqoop

Sqoop offers a number of benefits for data transfer, including:

- **High performance:** Sqoop is able to transfer large amounts of data quickly and efficiently.
- **Scalability:** Sqoop can be scaled to handle large data sets and high-volume workloads.
- **Flexibility:** Sqoop supports a variety of relational databases and Hadoop distributions.
- **Ease of use:** Sqoop is easy to use and configure, even for users with limited Hadoop experience.

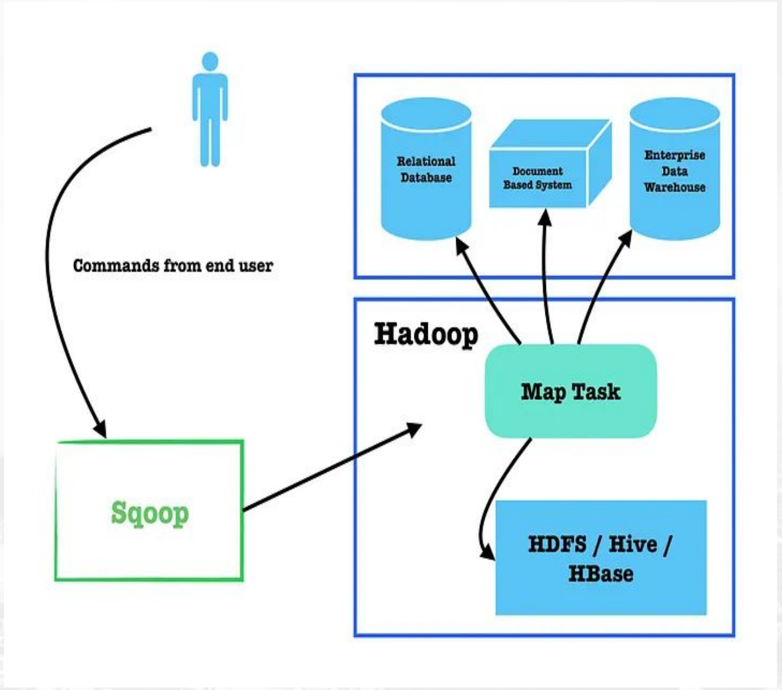
Sqoop Components

APACHE SQOOP

- **Sqoop client:** The Sqoop client is a Java application that runs on the Hadoop client node. It is responsible for initiating and managing data transfer requests.
- **Sqoop server:** The Sqoop server is a Java application that runs on the database server. It is responsible for executing data transfer requests from the Sqoop client.
- **Sqoop Connectors:** Sqoop connectors are responsible for establishing communication with the relational database and transferring data between the database and Hadoop. Each connector is implemented as a Java class and provides support for a specific relational database.

The Sqoop data transfer process works as follows

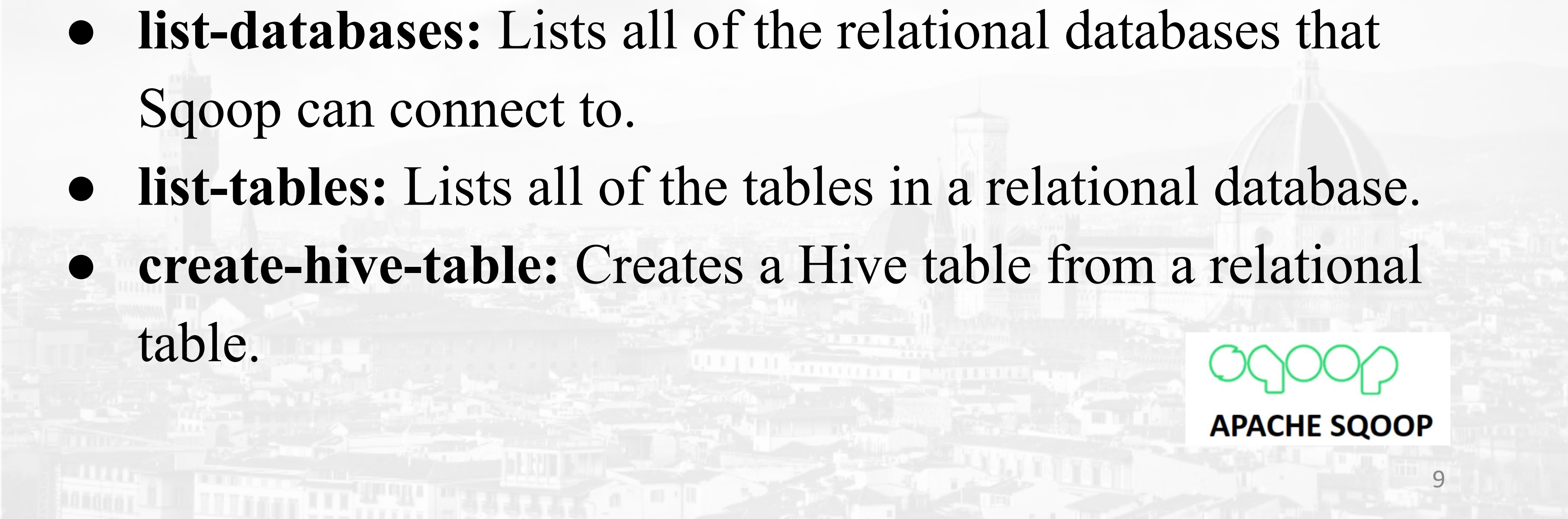
1) The Sqoop client initiates a data transfer request by submitting a job to the Hadoop cluster.
2) The Hadoop cluster assigns the job to a mapper task.
3) The mapper task connects to the relational database using the Sqoop connector.
4) The mapper task reads data from the relational database and writes it to HDFS.
5) The mapper task completes and the Hadoop cluster reduces the data to produce a final output file.
6) The Sqoop client retrieves the final output file from HDFS and stores it in the target location.



APACHE SQOOP

Sqoop provides a number of commands for transferring data between Hadoop and relational databases. The most common commands are:

- **import:** Imports data from a relational database to Hadoop.
- **export:** Exports data from Hadoop to a relational database.
- **list-databases:** Lists all of the relational databases that Sqoop can connect to.
- **list-tables:** Lists all of the tables in a relational database.
- **create-hive-table:** Creates a Hive table from a relational table.



To import data from MySQL to Hadoop in Cloudera using Sqoop, follow these steps:

1) **Install Sqoop.** If Sqoop is not already installed on your Cloudera cluster, you can install it using the following command:

```
sudo yum install sqoop
```

2) **Configure Sqoop.** You need to configure Sqoop to connect to your MySQL database and Hadoop cluster. To do this, edit the sqoop.properties file, which is located in the /etc/sqoop/conf directory.

   In the sqoop.properties file, you need to set the following properties:

   - sqoop.connector.mysql.connectString: The JDBC connection string for your MySQL database.
   - sqoop.connector.mysql.username: The username for your MySQL database.
   - sqoop.connector.mysql.password: The password for your MySQL database.
   - sqoop.target.dir: The HDFS directory where you want to import the data.

3) **Import the data.** To import the data, run the following command:

```
sqoop import \
-Dsqoop.connector.mysql.connectString=localhost:3306/mydb \
-Dsqoop.connector.mysql.username=root \
-Dsqoop.connector.mysql.password=password \
-Dsqoop.table=mytable \
-Dsqoop.target.dir=/user/hive/warehouse/mytable
```

This command will import the `mytable` table from the MySQL database `mydb` to the Hadoop directory `/user/hive/warehouse/mytable`

4) Verify the import. Once the import is complete, you can verify that the data was imported successfully by running the following command:

```
hadoop fs -ls /user/hive/warehouse/mytable
```

This command will list the contents of the Hadoop directory where the data was imported.

You can also use Hive to inspect the imported data by running the following command:

```
hive -e "SELECT * FROM mytable"
```

This command will display all of the rows in the mytable table.

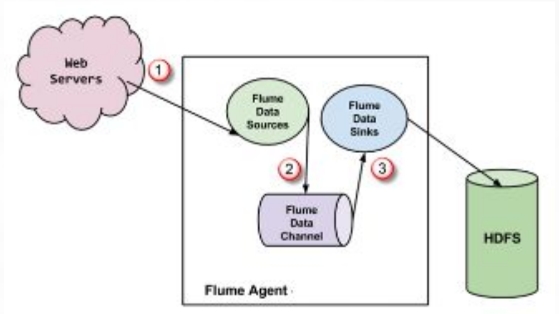
APACHE SQOOP

Flume

- Apache Flume is a distributed, reliable, and scalable service for collecting, aggregating, and moving large amounts of streaming data to a central data store.

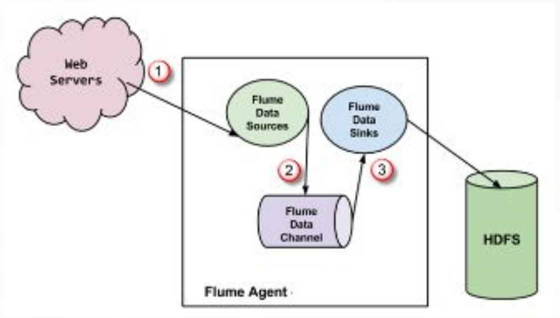
- It is a popular choice for collecting and processing data from a variety of sources, including web servers, application servers, databases, and sensors.

Flume Architecture



- **Sources:** Sources are responsible for collecting data from different sources. There are many different types of sources available for Flume, including file sources, network sources, and database sources.
- **Channels:** Channels are responsible for buffering and transporting data between sources and sinks. Flume provides a variety of different channel types, including memory channels, disk channels, and network channels.
- **Sinks:** Sinks are responsible for storing data in a destination, such as a Hadoop Distributed File System (HDFS), HBase, or a relational database

Data Flow in Flume



Data flows through Flume in a three-step process:

1. **Sources collect data from different sources and send it to channels.**
2. **Channels buffer and transport data to sinks.**
3. **Sinks store data in a destination.**

Flume Use Cases

Flume can be used for a variety of purposes, including:

- **Log collection:** Flume can be used to collect log data from different servers and send it to a central repository for analysis.
- **Real-time analytics:** Flume can be used to collect and process data from different sources in real time to generate insights.
- **Data ingestion:** Flume can be used to ingest data into a data warehouse or data lake for further analysis and processing.

Stepwise process for injecting stream data with Flume

1. **Identify the source of the stream data.** Flume can collect data from a variety of sources, including web servers, application servers, databases, and sensors.
2. **Choose a source type.** Flume provides a variety of different source types, such as file sources, network sources, and database sources.
3. **Configure the source.** The source configuration should specify the source type, the location of the data, and any other relevant parameters.
4. **Start the source.** Once the source is configured, it can be started to collect data.
5. **Choose a channel type.** Flume provides a variety of different channel types, including memory channels, disk channels, and network channels.
6. **Configure the channel.** The channel configuration should specify the channel type, the buffer size, and any other relevant parameters.
7. **Start the channel.** Once the channel is configured, it can be started to buffer and transport data.
8. **Choose a sink type.** Flume provides a variety of different sink types, such as HDFS sinks, HBase sinks, and relational database sinks.
9. **Configure the sink.** The sink configuration should specify the sink type, the destination location, and any other relevant parameters.
10. **Start the sink.** Once the sink is configured, it can be started to store data in the destination.
11. **Test the Flume agent.** Once the source, channel, and sink are configured and started, the Flume agent can be tested to ensure that it is collecting and processing data correctly.

The following example shows how to configure a Flume agent to collect data from a file and store it in HDFS:

```
# Source configuration
agent1.sources = r1
r1.type = exec
r1.command = tail -f /var/log/nginx/access.log

# Channel configuration
agent1.channels = c1
c1.type = memory
c1.capacity = 10000
c1.transactionCapacity = 100
```

Cont..

```
# Sink configuration
agent1.sinks = k1
k1.type = hdfs
k1.hdfs.path = hdfs://localhost:9000/flume/logs
k1.hdfs.fileType = SequenceFile
# Bind the source and sink to the channel
agent1.sources.r1.channels = c1
agent1.sinks.k1.channel = c1
```

Once this configuration is saved, the Flume agent can be started with the following command:

```
flume-ng agent --conf conf --name agent1
```

The Flume agent will start collecting data from the file /var/log/nginx/access.log and store it in HDFS.

Pig

- Pig technology is a high-level programming language that is used to process and analyze large datasets stored in Hadoop.
- It is a powerful tool for data scientists and engineers to extract valuable insights from their data.
- Pig technology is closely related to Hadoop.
- Pig is built on top of Hadoop, and it uses Hadoop to distribute and process data in parallel. This makes Pig a powerful tool for analyzing large datasets

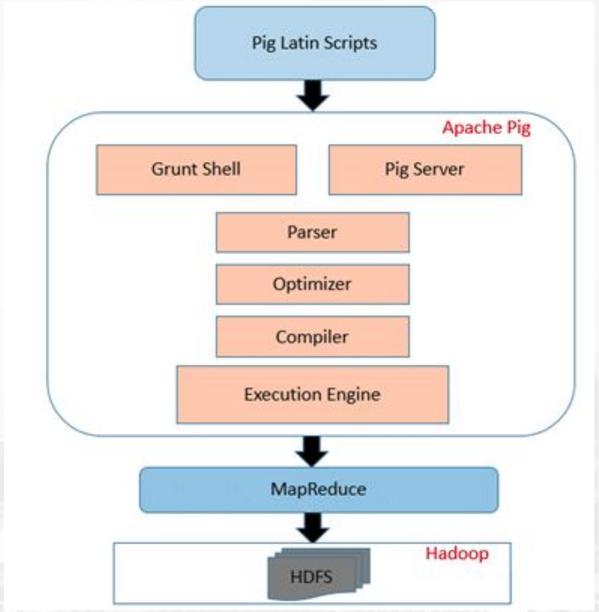
Here are some examples of how Pig technology can be used with Hadoop:

- **Loading data into Hadoop:** Pig can be used to load data into Hadoop from a variety of sources, such as relational databases, CSV files, and XML files.
- **Processing data in Hadoop:** Pig can be used to process data stored in Hadoop using a variety of operations, such as filtering, sorting, and aggregating.
- **Storing data in Hadoop:** Pig can be used to store data in Hadoop in a variety of formats, such as HDFS files and HBase tables.

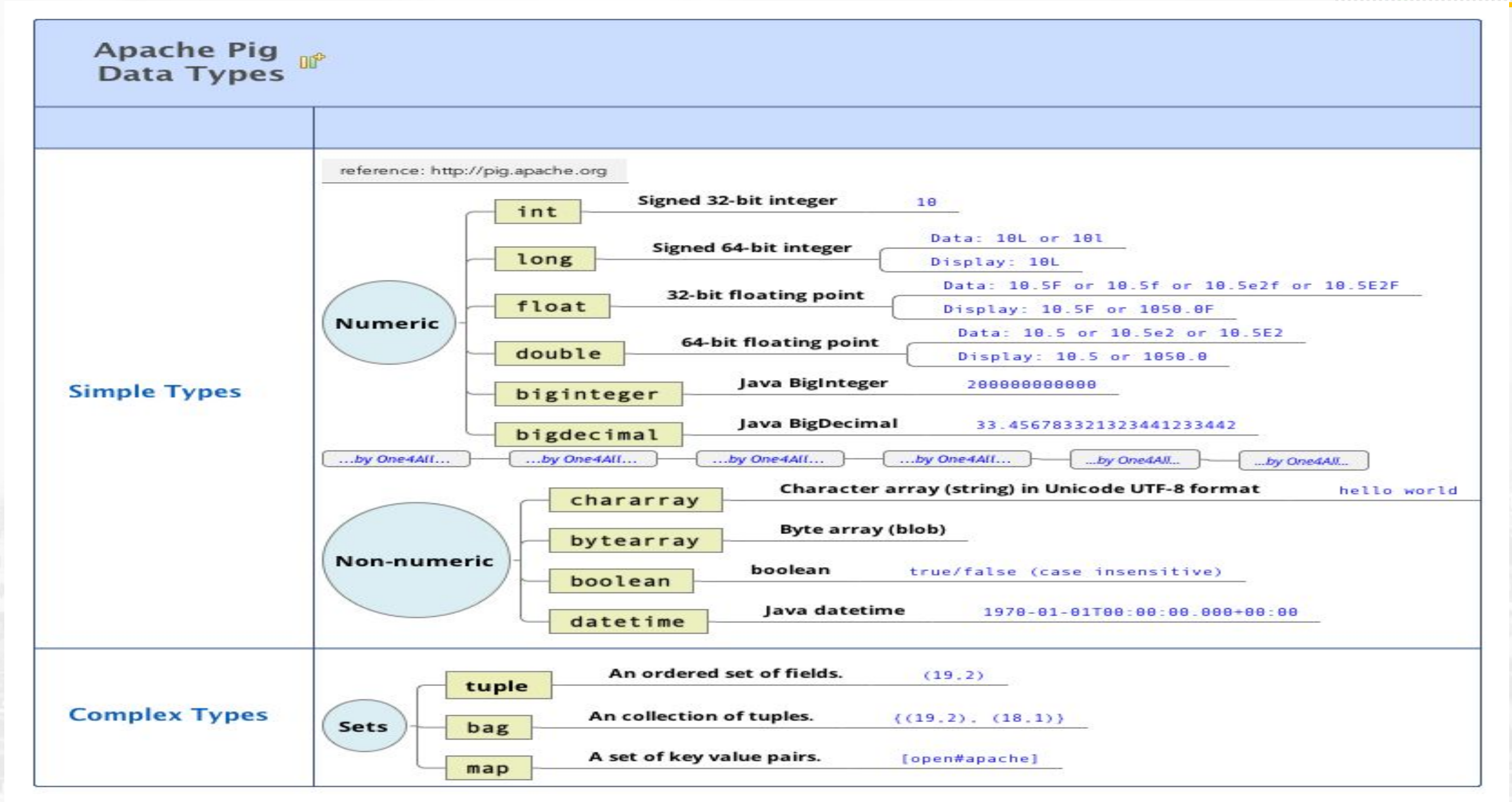
Pig Architecture

The Pig architecture consists of the following components:

- **Pig parser:** The Pig parser parses the Pig Latin program into an abstract syntax tree (AST).
- **Pig optimizer:** The Pig optimizer optimizes the AST to improve performance.
- **Pig compiler:** The Pig compiler compiles the AST into a MapReduce program.
- **Pig runtime:** The Pig runtime executes the MapReduce program on a Hadoop cluster to process the data.



Pig Data Types



Apache Pig Data Types

reference: http://pig.apache.org

Simple Types	Numeric	int	Signed 32-bit integer	10
		long	Signed 64-bit integer	Data: 10L or 10l Display: 10L
		float	32-bit floating point	Data: 10.5F or 10.5f or 10.5e2f or 10.5E2F Display: 10.5F or 1050.0F
		double	64-bit floating point	Data: 10.5 or 10.5e2 or 10.5E2 Display: 10.5 or 1050.0
		biginteger	Java BigInteger	200000000000
		bigdecimal	Java BigDecimal	33.456783321323441233442
	Non-numeric	chararray	Character array (string) in Unicode UTF-8 format	hello world
		bytearray	Byte array (blob)	
		boolean	boolean	true/false (case insensitive)
		datetime	Java datetime	1970-01-01T00:00:00.000+00:00
Complex Types	Sets	tuple	An ordered set of fields.	(19,2)
		bag	An collection of tuples.	{(19,2), (18,1)}
		map	A set of key value pairs.	[open#apache]

...by One4All... ...by One4All... ...by One4All... ...by One4All... ...by One4All... ...by One4All...

Pig Functions

Pig also provides a number of functions for manipulating data, including:

- **Bag functions:** FUNCTIONS for manipulating bags, such as FILTER, FOREACH, LIMIT, ORDER, and SUM.
- **Tuple functions:** FUNCTIONS for manipulating tuples, such as FIRST, LAST, SIZE, and TOBAG.
- **Math functions:** FUNCTIONS for performing mathematical operations, such as ABS, ADD, DIVIDE, MAX, MIN, and MOD.
- **String functions:** FUNCTIONS for manipulating strings, such as CONCAT, LENGTH, LOWER, REGEX_EXTRACT, and SUBSTRING.
- **Date and time functions:** FUNCTIONS for manipulating dates and times, such as CURRENT_DATE, CURRENT_TIME, and PARSE_DATETIME.

Pig functions can be used to perform a variety of data processing tasks, such as:

- **Filtering:** Filtering out data that does not meet certain criteria.
- **Sorting:** Sorting data in a specific order.
- **Joining:** Joining two or more data sets together.
- **Aggregating:** Aggregating data to produce summary statistics, such as counts, sums, and averages.

Pig Program : Example-1

```
registered_users = LOAD 'registered_users.txt' USING PigStorage(',');
filtered_users = FILTER registered_users BY age > 18;
STORE filtered_users INTO 'filtered_users.txt' USING PigStorage(',');
```

- This above program will load the data from the file registered_users.txt into a Pig bag called registered_users.
- It will then filter the bag to only include users who are over 18 years old.
- The filtered data will then be stored in the file filtered_users.txt.

Pig Program : Example-2

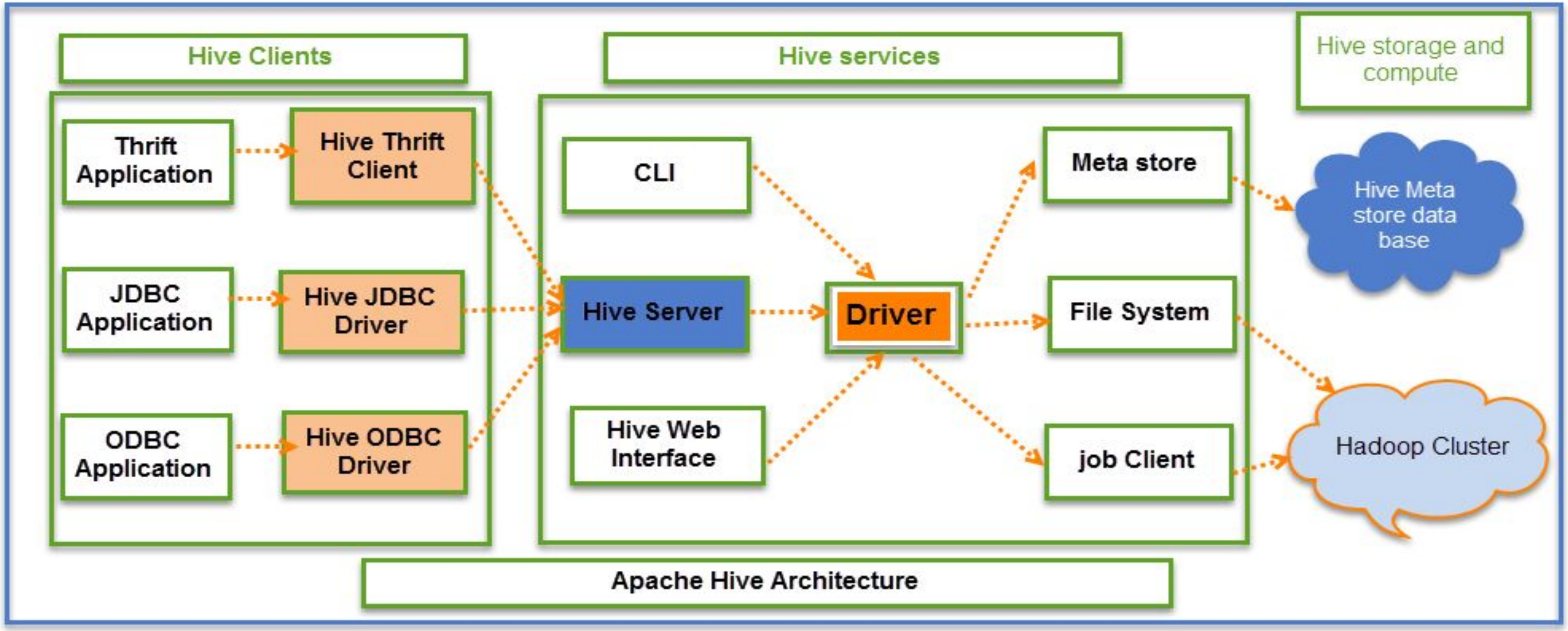
```
registered_users = LOAD 'registered_users.txt' USING PigStorage(',');
order_history = LOAD 'order_history.txt' USING PigStorage(',');
JOINED_DATA = JOIN registered_users BY user_id, order_history BY user_id;
STORE joined_data INTO 'joined_data.txt' USING PigStorage(',');
```

- This program will load the data from the files registered_users.txt and order_history.txt into Pig bags called REGISTERED_USERS and ORDER_HISTORY, respectively.
- It will then join the two bags together on the user_id field.
- The joined data will then be stored in the file joined_data.txt.

Hive

- Hive: A Data Warehouse Infrastructure Built on Top of Hadoop
- Apache Hive is a data warehouse software project built on top of Apache Hadoop for providing data query and analysis.
- Hive gives an SQL-like interface to query data stored in various databases and file systems that integrate with Hadoop.

Hive Architecture



Hive Architecture (cont..)

Hive architecture consists of the following components:

- **Hive clients:** Hive clients are used to interact with Hive. There are multiple types of Hive clients, including:
  - **Hive command-line interface (CLI)**: The Hive CLI is a text-based client that allows users to submit HiveQL queries and interact with Hive.
  - **Hive Web UI:** The Hive Web UI is a graphical user interface (GUI) that allows users to submit HiveQL queries, view metadata, and manage Hive tables and databases.
  - **Hive JDBC/ODBC drivers:** The Hive JDBC/ODBC drivers allow users to connect to Hive from other applications, such as BI tools and programming languages.
- **Hive Server:** The Hive Server is a service that accepts HiveQL queries from clients and executes them on the Hadoop cluster.
- **Hive Driver:** The Hive Driver is responsible for parsing HiveQL queries and converting them into MapReduce jobs.
- **Hive Compiler:** The Hive Compiler compiles HiveQL queries into MapReduce jobs.
- **Hive Metastore:** The Hive Metastore is a database that stores metadata about Hive tables, databases, and columns.
- **Hive Execution Engine:** The Hive Execution Engine is responsible for executing MapReduce jobs on the Hadoop cluster.

Hive Architecture (cont..)

When a user submits a HiveQL query to the Hive Server, the following steps occur:

1. The Hive Server parses the HiveQL query and converts it into a logical plan.
2. The Hive Compiler converts the logical plan into a DAG of MapReduce jobs.
3. The Hive Execution Engine executes the MapReduce jobs on the Hadoop cluster.
4. The results of the MapReduce jobs are returned to the Hive Server.
5. The Hive Server returns the results of the HiveQL query to the client.

Hive Commands

<u>Some of the most common Hive commands:</u>

- **CREATE DATABASE:** Creates a new database in Hive.
- **SHOW DATABASES:** Lists all of the databases in Hive.
- **USE DATABASE:** Sets the current database to the specified database.
- **CREATE TABLE:** Creates a new table in Hive.
- **SHOW TABLES:** Lists all of the tables in the current database.
- **DESCRIBE TABLE:** Displays information about a table, such as its schema and column names.
- **LOAD DATA INTO TABLE:** Loads data into a Hive table.
- **SELECT:** Queries data from Hive tables.
- **INSERT OVERWRITE TABLE:** Inserts data into a Hive table, overwriting any existing data.
- **UPDATE TABLE:** Updates data in a Hive table.
- **DELETE FROM TABLE:** Deletes data from a Hive table.

Hive Commands Example

```
CREATE DATABASE my_database;

USE DATABASE my_database;

CREATE TABLE my_table (

  id INT,

  name STRING

);

SHOW TABLES;

DESCRIBE my_table;

LOAD DATA INTO TABLE my_table FROM '/path/to/data/file';

SELECT * FROM my_table;

INSERT OVERWRITE TABLE my_table SELECT * FROM another_table;

UPDATE my_table SET name = 'New Name' WHERE id = 1;

DELETE FROM my_table WHERE id = 2;
```

Zookeeper

- ZooKeeper is a distributed coordination service for distributed systems. It is used to maintain configuration information, naming, providing distributed synchronization, and providing group services.
- ZooKeeper is a centralized service, but it is highly available and fault-tolerant. It is designed to be able to handle node failures and network partitions. ZooKeeper is also very scalable and can handle large numbers of clients and requests.

Here are some of the key features of ZooKeeper:

- Highly available and fault-tolerant: ZooKeeper is designed to be able to handle node failures and network partitions. It replicates data across multiple nodes and uses a consensus algorithm to ensure that the data is consistent.
- Scalable: ZooKeeper can handle large numbers of clients and requests. It is designed to be able to scale to meet the needs of large distributed systems.
- Fast: ZooKeeper is designed to be fast and efficient. It uses a variety of techniques to optimize performance, such as caching and in-memory data structures.
- Secure: ZooKeeper supports authentication and authorization to protect data from unauthorized access.

Features of Zookeeper

- **Leader election:** ZooKeeper can be used to elect a leader for a distributed cluster. This is useful for ensuring that there is only one node in the cluster that is responsible for certain tasks, such as coordinating transactions or processing requests.
- **Configuration management:** ZooKeeper can be used to store and manage configuration information for a distributed cluster. This allows all of the nodes in the cluster to have access to the same configuration information.
- **Distributed synchronization:** ZooKeeper can be used to provide distributed synchronization for a distributed cluster. This allows nodes in the cluster to coordinate their actions and avoid conflicts.
- **Group services:** ZooKeeper can be used to provide group services for a distributed cluster. This allows nodes in the cluster to form groups and coordinate their activities.

How Zookeeper is used in Hadoop

- **HDFS:** HDFS uses ZooKeeper to elect a leader node, which is responsible for managing the HDFS namespace. ZooKeeper is also used to coordinate the activities of the HDFS DataNodes.
- **HBase:** HBase uses ZooKeeper to elect a leader node, which is responsible for managing the HBase metadata. ZooKeeper is also used to coordinate the activities of the HBase RegionServers.
- **Hive:** Hive uses ZooKeeper to store the Hive metastore, which contains information about the Hive tables, databases, and columns. ZooKeeper is also used to coordinate the activities of the Hive Metastore Servers.

Apache Zookeeper

Apache Spark

- Apache Spark is a unified analytics engine for large-scale data processing.
- It provides high-level APIs in Java, Scala, Python, and R, and an optimized engine that supports general execution graphs.
- It also supports a wide range of workloads, including batch processing, stream processing, machine learning, and graph processing.
- Spark is designed to be fast and scalable.
- It can process large datasets in memory, and it can be scaled up or down to meet the needs of your workload.
- Spark is also fault-tolerant, so it can continue to operate even if some of its nodes fail.

Features of Apache Spark

- **Unified analytics engine:** Spark can be used for a variety of data processing tasks, including batch processing, stream processing, machine learning, and graph processing.

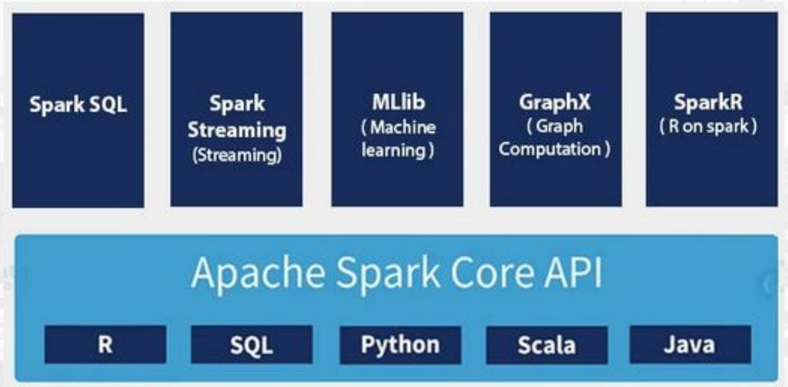
- **High-level APIs:** Spark provides high-level APIs in Java, Scala, Python, and R, making it easy to use Spark for a variety of data processing tasks.

- **Optimized engine:** Spark has an optimized engine that supports general execution graphs. This allows Spark to efficiently process large datasets

- **Scalable and fault-tolerant:** Spark is designed to be scalable and fault-tolerant. It can be scaled up or down to meet the needs of your workload, and it can continue to operate even if some of its nodes fail.

Apache Spark Ecosystem

- The Apache Spark ecosystem is a collection of libraries and tools that extend the capabilities of Apache Spark.
- The Spark ecosystem includes libraries for machine learning, graph processing, stream processing, and more.

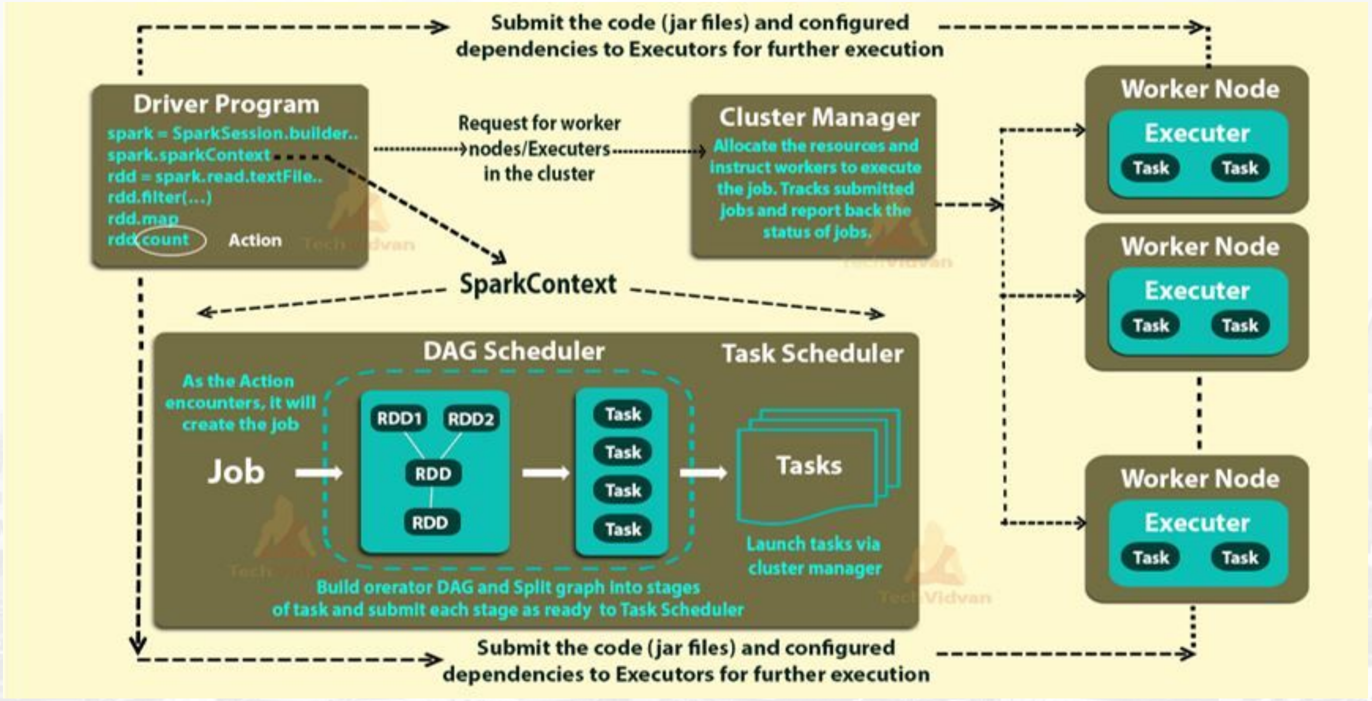

Spark SQL
Spark Streaming (Streaming)
MLlib (Machine learning)
GraphX (Graph Computation)
SparkR (R on spark)
Apache Spark Core API
R
SQL
Python
Scala
Java

Spark Ecosystem Components

- **MLlib:** MLlib is a machine learning library for Spark. It provides a variety of machine learning algorithms, including classification, regression, clustering, and recommendation.
- **GraphX:** GraphX is a graph processing library for Spark. It provides a variety of graph processing algorithms, such as shortest path, page rank, and triangle counting.
- **Spark Streaming:** Spark Streaming is a stream processing library for Spark. It allows you to process data streams in real time.
- **Structured Streaming:** Structured Streaming is a library that enables high throughput and fault-tolerant stream processing on Spark.
- **Spark SQL:** Spark SQL is a SQL query engine for Spark. It allows you to query Spark DataFrames and DataSets using SQL.
- **SparkR:** SparkR is an R package for Spark. It allows you to use Spark from within R.

The Spark ecosystem is constantly evolving, and new components are being added all the time. The Spark ecosystem provides a wide range of tools that can be used to solve a variety of data processing problems.

Spark Architecture


Submit the code (jar files) and configured dependencies to Executors for further execution
Driver Program
spark = SparkSession.builder..
spark.sparkContext
rdd = spark.read.textFile..
rdd.filter(...)
rdd.map
rdd.count
Action
Request for worker nodes/Executers in the cluster
Cluster Manager
Allocate the resources and instruct workers to execute the job. Tracks submitted jobs and report back the status of jobs.
SparkContext
DAG Scheduler
Task Scheduler
As the Action encounters, it will create the job
Job
RDD1
RDD2
RDD
RDD
Task
Task
Task
Task
Tasks
Launch tasks via cluster manager
Build orerator DAG and Split graph into stages of task and submit each stage as ready to Task Scheduler
Worker Node
Executer
Task
Task
Worker Node
Executer
Task
Task
Worker Node
Executer
Task
Task
Submit the code (jar files) and configured dependencies to Executors for further execution

Spark Architecture

Spark architecture is based on two main abstractions: Resilient Distributed Dataset (RDD) and Directed Acyclic Graph (DAG).

1) **Resilient Distributed Dataset (RDD)**: RDD is a fundamental data structure in Spark. It is an immutable, distributed collection of data elements that can be partitioned across multiple nodes and processed in parallel.
2) **Directed Acyclic Graph (DAG)**: DAG is a logical representation of a Spark application. It represents the flow of data through the application and the dependencies between the different tasks.

- Spark uses a cluster manager to manage the resources in the cluster. The cluster manager can be either Spark's own standalone cluster manager, Apache Mesos, or YARN.
- Spark applications are typically submitted to the cluster manager using a Spark shell or a client program. The cluster manager then allocates resources to the application and starts the Spark driver.
- The Spark driver is responsible for coordinating the execution of the application. It breaks the application down into tasks and schedules the tasks to be executed on the worker nodes.
- The worker nodes are responsible for executing the tasks that are scheduled by the driver. The worker nodes also store the data that is being processed by the application.

Spark Program Example

```
import pyspark

# Create a SparkSession

spark = pyspark.sql.SparkSession.builder.getOrCreate()

# Read the data from a file

data = spark.read.csv("data.csv")

# Sort the data by the "age" column

sorted_data = data.sort("age")

# Print the sorted data to the console

sorted_data.show()

# Stop the SparkSession

spark.stop()
```

This python program will read the data from a CSV file called data.csv and sort the data by the age column. It will then print the sorted data to the console.

Spark Use Cases

Spark is used by a wide range of companies for a variety of data processing tasks. Here are a few examples:

- **Netflix:** Netflix uses Spark to analyze its user data and to recommend movies and TV shows to its users.
- **Amazon:** Amazon uses Spark to power its search engine and to recommend products to its customers.
- **Yahoo:** Yahoo uses Spark to process its search engine data and to power its advertising platform.
- **Twitter:** Twitter uses Spark to process its tweet stream in real time and to detect trends and anomalies.
- **Spotify:** Spotify uses Spark to query its music catalog and to generate personalized playlists for its users.