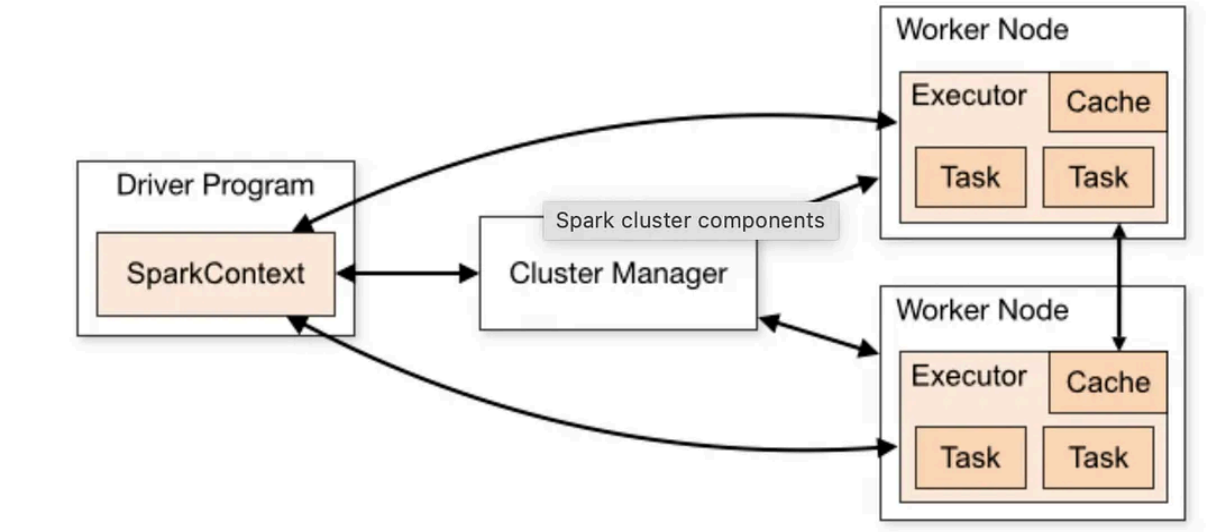
Unit - V Spark and Real-Time Analytics

5.1 Introduction to Big Data tool Spark : Main components of Spark Architecture, Features of Spark, Spark Software Stack

Spark is a super-fast engine for big data that can store, process, and analyze data efficiently. It's a powerful tool that lets you process large amounts of data across many computers at the same time.

Main components of Spark Architecture

Component	Description
Driver Program	Runs application, creates **Context** & schedules jobs.
Cluster Manager	Allocates resources across clusters
Spark Context	Communicates with cluster manager.
Executors	Run tasks of driver on nodes; store data in memory.
Tasks	Smallest unit of work executed by an executor.



Features of Spark

- **Speed:** Up to 100× faster than MapReduce. Controlled data partitioning.
- **Powerful Caching:** In-memory storage and disk persistence.

- **Deployment:** Can run on Mesos, YARN, or Spark's cluster manager.
- **Real-Time:** Supports real-time computation with low latency.
- **Polyglot:** Supports Java, Scala, Python, and R;
- **Scalable:** Handles large-scale data across clusters efficiently.

Spark Software Stack:

Layer	Description
Spark Core	Engine for distributed task scheduling, memory management, fault recovery.
Spark SQL	For structured data processing using **SQL queries**.
Spark Streaming	Real-time data stream processing.
MLlib	Machine learning library with algorithms for classification, regression, clustering, etc.
GraphX	Graph processing library for graph analytics.
SparkR / PySpark	APIs for R and Python developers.

5.2 Introduction to Data Analysis with Spark : Spark SQL

Spark SQL is a Spark module for working with structured and semi-structured data using SQL queries. It allows you to query data using SQL or DataFrame / Dataset APIs.

Features

1. **Unified API:** Work with SQL, DataFrames, and Datasets.
2. **Compatibility with Hive:** Can run Hive queries and access Hive tables.
3. **Optimized Execution:** Catalyst optimizer improves query performance.
4. **Supports multiple formats:** Parquet, ORC, JSON, JDBC, etc.
5. **Interactive Analysis:** Can run ad-hoc queries on large datasets.

5.3 Programming with RDDs and Machine learning with MLib

Programming with RDD (Resilient Distributed Datasets)

The core Spark abstraction – an immutable, distributed collection of objects that can be processed in parallel. Programming with RDDs lets you process large datasets efficiently, reliably, and in parallel while giving Spark the power to optimize computations automatically.

- **Immutable:** Once created, cannot be changed.
- **Distributed:** Stored across multiple nodes in the cluster.
- **Lazy Evaluation:** Transformations are computed only when an action is called.
- **Fault-Tolerant:** Tracks lineage to recompute lost partitions automatically.

RDD Operations

Type	Examples	Description
Transformation	map(), filter(), flatMap(), reduceByKey()	Create new RDD from existing one
Action	collect(), count(), take(), saveAsTextFile()	Trigger execution and return results

Machine learning with MLib

MLlib: Spark's **distributed machine learning library** for large-scale data.

Features

- Built for **scalable ML** on clusters.
- Provides **common algorithms**: classification, regression, clustering, collaborative filtering.
- Supports **feature extraction, transformation, and evaluation**.
- Works seamlessly with **RDDs and DataFrames**.

5.4 Data ETL (Extract, Transform and Load) Process: Composing Spark Program steps for ETL

Data ETL (Extract, Transform and Load)

The process of moving data from source systems, transforming it, and storing it for analysis. Spark makes ETL fast, scalable, and distributed.

Compose a Spark Program for ETL

1. Extract (Read Data)

- Load data from various sources: HDFS, S3, JDBC, JSON, CSV, Parquet.

2. Transform (Process Data)

- Clean, filter, aggregate data using RDD/DataFrame operations

3. Load (Save Data)

- Write processed data to HDFS, databases, or other storage systems.
- Supports formats: CSV, JSON, Parquet, ORC, JDBC.

5.5 Analytics, Reporting and Visualization

Analytics

- Examine data to find **patterns, trends, and insights**.
- Use **Spark Core, Spark SQL, or MLlib** for processing and predictive analysis.

Reporting

- Summarize and present data in **structured formats**.
- Supports **batch or real-time reports**.
- Output formats: **CSV, JSON, dashboards, BI tools**.

Visualization

- Graphically represent data for **easy understanding**.
- Tools: **Matplotlib, Seaborn, Plotly**, Tableau, Power BI.
- Enables **interactive charts and dashboards**.

5.6 Apache Spark Streaming Platform: Spark Streaming Architecture, Spark streaming vs Structured streaming, Internal Working of Spark Streaming

Spark Streaming Architecture

Spark Streaming is an extension of Apache Spark that enables scalable, high-throughput, fault-tolerant stream processing of live data streams.

Components

1 Input Data Sources

Spark Streaming can consume data from: Kafka, Flume, Kinesis, Socket streams, HDFS/S3 and other file systems

2 DStream (Discretized Stream)

The **core abstraction** in Spark Streaming. Represents a continuous stream of data. Internally, each DStream is a sequence of **RDDs**, each corresponding to one micro-batch.

3 Receivers

Responsible for ingesting data from data sources. Two types: 1 **Reliable receivers** → ensure data is acknowledged after being stored. 2 **Unreliable receivers** → no acknowledgment.

4 Batch Interval

Defines how often data is sliced into batches (e.g., 1 second, 5 seconds).

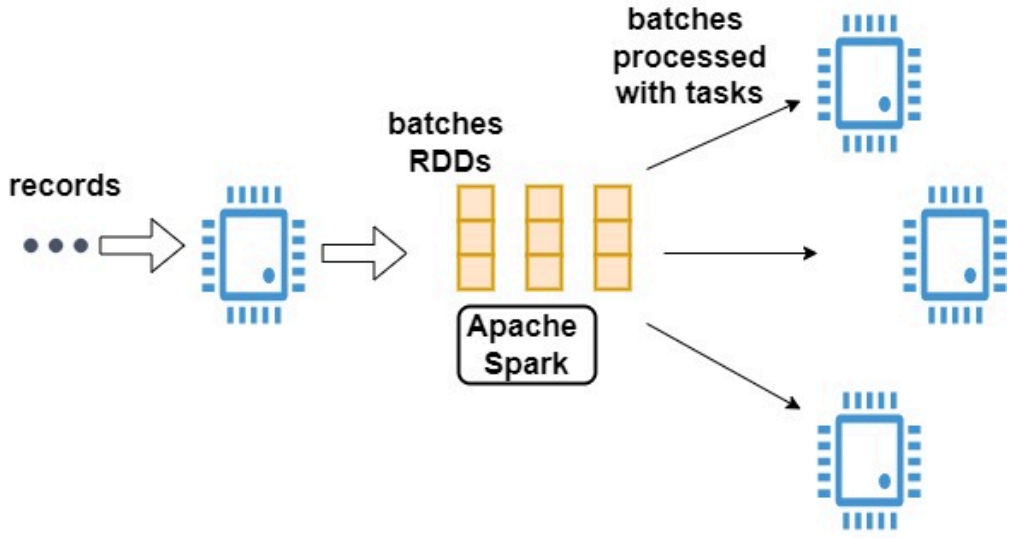
Each batch → an RDD.

5 Processing Layer

Uses the Spark engine and its components: RDD transformations, Shuffle, Executors, Driver program

6 Output Operations

Results can be pushed to: Databases (MySQL, Cassandra, HBase), Filesystems, Dashboards, Kafka topics



Records processed in batches with short tasks
Each batch is an RDD (partitioned dataset)

Spark Streaming vs. Structured Streaming

Feature	Spark Streaming (DStreams)	Structured Streaming
Processing Model	Micro-batch processing only	Micro-batch or continuous processing
API Level	RDD-based, low-level	Dataset/DataFrame API, high-level
Fault Tolerance	Achieved via RDD lineage	Uses **checkpointing + logs**, stronger guarantees

Event-time Support	Limited	Full support (event-time, watermarks)
Ease of Use	More complex	Much simpler, SQL-like
Performance	Slower for complex operations	Faster and optimized by Catalyst engine

Internal Working of Spark Streaming

Spark Streaming works on the principle of **micro-batching**, where incoming data is divided into small batches.

Step-by-Step Internal Workflow

1. Data Ingestion

A receiver (or Kafka source) continuously collects data. Data is stored in **Spark executors' memory** as blocks.

2. Batch Construction

At every batch interval (e.g., every 2 seconds), Spark creates a new batch. All data collected in that interval → forms an **RDD**.

3. Job Generation (Driver Program)

For each batch, the Spark Streaming scheduler generates a **DAG** of RDD transformations. Creates a Spark job per batch.

4. Job Execution

Spark's DAG Scheduler organizes tasks into stages. Tasks are shipped to executors and processed.

5. State Management

Stateful transformations maintain data across batches: 1 updateStateByKey, 2 mapWithState

Spark uses checkpointing to store: 1 Metadata (streaming graph) 2 RDD lineage. This helps in recovery after a failure.

6. Fault Tolerance

Spark Streaming ensures: **Receiver fault tolerance:** Data is replicated if WAL (Write Ahead Log) is enabled. **Executor/driver failure recovery:** Uses checkpointing. **RDD lineage recomputation:** Lost partitions can be recomputed.

7. Output

The final results of each batch are stored or pushed to external systems.

Questions:

1. Explain Real-time analytics
2. State the use of Apache Spark.
3. Define RDD.
4. Write a code for building spark SQL application to count the total number of WARN lines in the logs.txt file (assume appropriate log file)
5. Difference between Spark and Hadoop map reduce.
6. Describe Apache Spark Architecture, Explain Apache Spark Architecture.
7. State spark shell.
8. Explain spark core RDD operations.
9. Explain spark real time use case for data analytics project architecture.
10. Implement code for building SPARK SQL application with SBT, Write a code for building spark SQL application with SBT.
11. Explain Spark Real Time use case
12. State RDD. Explain RDD operation & creating a RDD.