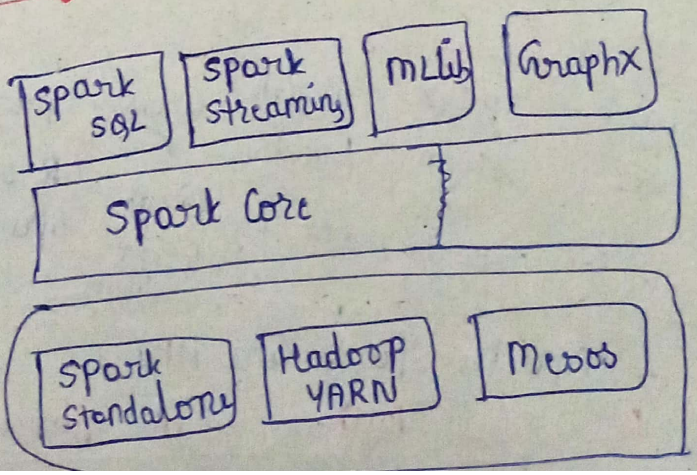
Apache Spark Ecosystem

1) Apache Spark
2) Ecosystem of Apache Spark
3) Components of Spark Ecosystem
   (i) Spark Core
   (ii) Spark SQL
   (iii) Spark Streaming
   (iv) Spark MLlib
   (v) Spark GraphX
   (vi) SparkR.
4) Features of Apache Spark Ecosystem.

Spark Features :-

1) Spark is written in Scala Programming language and runs in JVM
2) API - Scala, Java, Python, R
3) Interactive shell - Scala & Python
4) Data sources; SQL, NoSQL, S3, HDFS, Local file system
5) Good fit for iterative tasks like Machine Learning algos.

Components of Spark :-



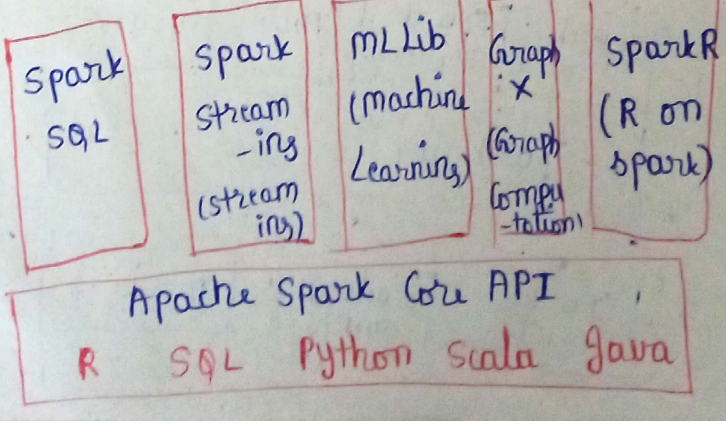
Data from various sources is collected and made to run in spark clusters (group of machines).

* Data Engineering
  - Data software storage
    - cheap storage
    - costly storage
      - cost of storage components is low
  - Data processing depends on (i) RAM (ii) storage (iii) processor (iv) n/w speed (v) Block speed (vi) OS (vii) hardware components (viii) SSD (ix) CPU

Apache Spark

* Spark is used for Dataprocessing
* Apache Spark is general purpose cluster computing system.
  - group of systems (virtual) working together
* It provides high-level API in Java, Scala, R, Python
* It has high-level tools for structured data processing, machine learning, graph processing and streaming
* Spark can run alone (or) on existing cluster manager

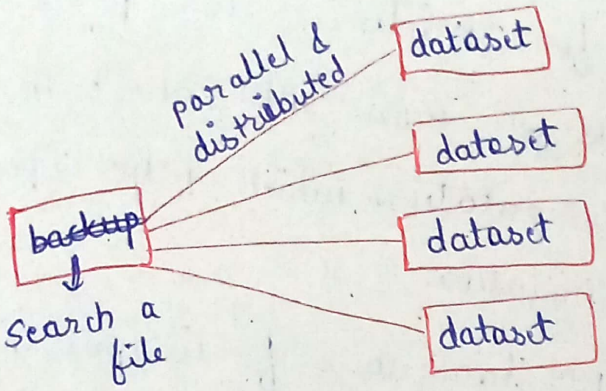
Introduction to Apache Spark Ecosystem Components



1) Apache Spark Core

* All functionalities provided by Apache Spark are built on top of Spark Core
* It delivers speed by providing in-memory computation capability
* Spark core is the foundation of parallel and distributed processing of huge dataset.

Stores data in RAM of servers which allows quick access and increases speed.



Features of spark core :-

1) Control essential I/O functionalities
2) significant / important in programming
3) task dispatching (dividing tasks & assigning)
4) Fault recovery
5) overcomes MapReduce by using in-memory computation
6) observes role of spark cluster.

RDD

* Spark Core is embedded with special collection called RDD (Resilient distributed dataset)
* RDD is among the abstractions of Spark
* RDD handles partitioning data across all the nodes in a cluster.

* Memory is held in the memory pool of cluster as single unit

Two operations

1) Transformation produces new RDDs from the existing RDDs

2) Action Action is used when we want to work with actual data set

Apache Spark SQL :-

* distributed framework for structured data processing
* Using Spark SQL, Spark gets more info about the structure of data & computation which helps Spark to perform extra optimization.
* doesn't depend on API/ language for computation but uses same execution engine while computing o/p.
* Spark SQL works to access structured & semi-structured info
* It enables powerful, interactive, analytical appln across both streaming & historical data.
* It is a spark module for structured data processing
* acts as distributed SQL query engine

Features

1) Cost based optimizer
2) Mid query fault tolerance

3) Full compability with existing Hive data

4) Dataframes and SQL provide a way to access a variety of data sources like Hive, Avro, ORC, JSON, Parquet, JDBC

5) allows to carry structured data inside spark pgrms using SQL or other.

Apache Spark Streaming

* It allows scalable, high-throughput, fault-tolerant stream processing of live data streams.
* Spark can access data from sources like kakfa, Flume, Kinesis or TCP socket.
* It can operate using various algos.
* Data received is given to file streaming, databases
* Spark uses Micro batching for real-time streaming
  - ↓
  - allows a process or task to treat a stream as sequence of small batches of data.
* Hence, spark streaming groups live data into small batches.
* Now sent for processing.

Phases of Spark Streaming :-

(a) Gathering

built-in stream sources
- Basic Sources ⇒ available in streaming Context API
  - Ex:- file systems
- Advanced sources
  - Ex:- kafka, Flume, kinesis

* Spark access data from different sources like kafka, flume, kinesis or TCP sockets.

b) Processing

gathered data is processed using complex algos

Ex :- map, reduce, join and window

c) Data storage

processed data is pushed out to file systems, databases

Dstream :- High level abstraction provided by spark Streaming is Dstream (or) Discretized stream.

→ signifies continuous stream of data

→ Formation of Dstream

- by high-level operations on other Dstreams.
- Sources like kafka, Flume & kinesis

* Dstream ⇒ Sequence of RDDs.

④ Apache Spark MLLib (Machine Learning Library)

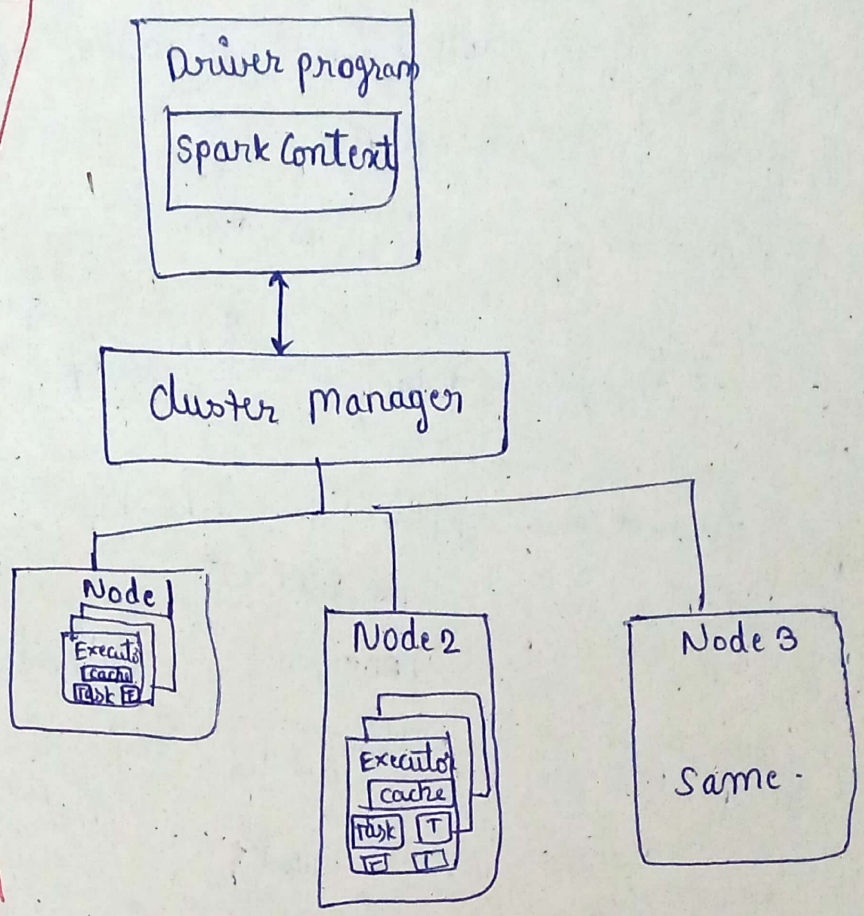
MLLib → scalable ML library → high-quality algo, high speed.

* Main aim → make ML scalable & easy
* Contains many ML libraries → has various algos like clustering, regression and also lowel level algos are present.
* In 2.0 version, Dataframe based API is primary ML API for Spark (bcoz more user friendly)
* Some more benefits → Data sources, SQL DF queries, uniform APIs.

~~Architecture~~ Components (Architecture)

→ ML Lib uses linear algebra package

Breeze
⇓
collection of libraries for numerical computations & ML



Master :- The machine on which driver program runs

Slave :- The machine on which executor program runs

Executor :- The process responsible for executing a task

DAG :- Directed Acyclic Graph

Job :- piece of code which reads some i/p from local, performs some computation on data & writes o/p

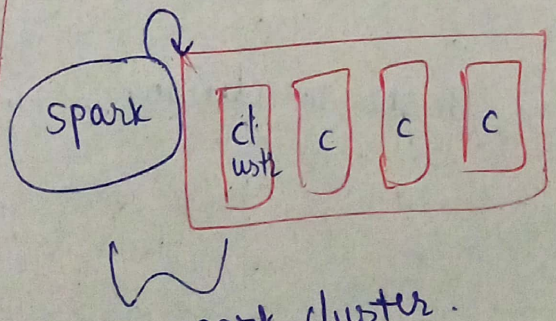
~~Task~~ Stages :- Jobs are divided into stages based on computational boundaries

cluster → group of virtual machines

Jobs :-

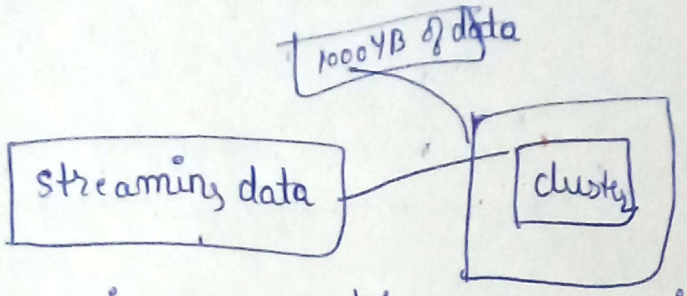
tasks :- Each stage has some tasks.

* one task is executed on one partition of data on one executor.

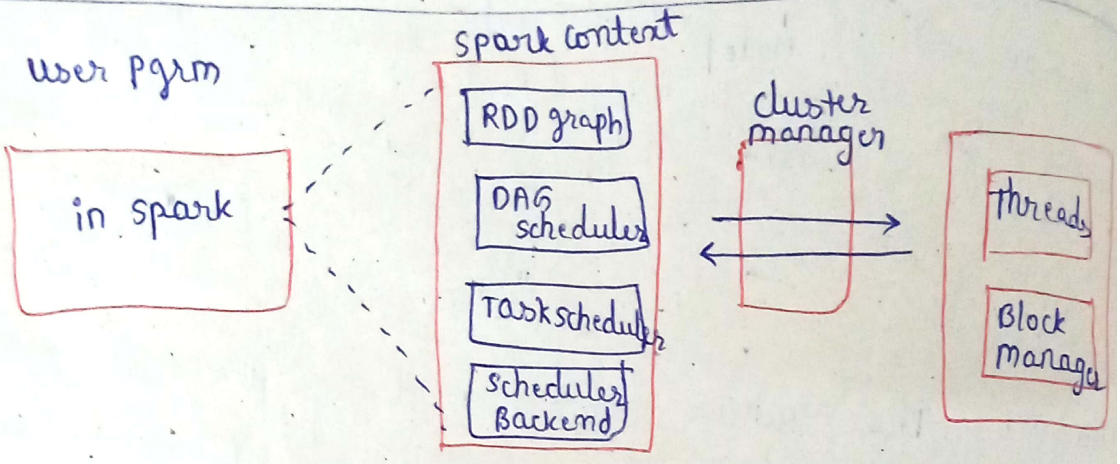


spark cluster.

* Spark SQL is preferred over SQL bcoz spark SQL deals with unstructured data, faster processing
* Pyspark => spark + python
*



* Used in various applications like banking



Spark Context :-

* represents connection to spark cluster
* used to create RDDs, accumulators
* broadcast variables on that cluster.

DAG Scheduler :- (collects tasks together, organized)

* Computes a DAG of stages for each job and submits them to task Scheduler.
* TS determines preferred locations for tasks and finds minimum schedule to run jobs

Task Scheduler :-

* responsible for sending tasks to cluster, running them, retrying them if there are failures.

Stand alone cluster → spark is installed in our pc.

Yarn → Yet another resource navigator

Mesos →

Block Manager Scheduler backend → backend interface for scheduling systems that allows plugging in different implementations.

How Spark Works ?

* Spark has small code base and system is divided into various layers.
* Each layer has some responsibilities and these layers are independent of each other.

Layer - 1 :- Interpreter

Spark uses Scala interpreter by making some modifications

↓

Enter code in spark console (like creating RDD's, apply operators)

↓

Spark creates operator graph.

↓

When user runs action, graph submitted to DAG scheduler

↓

DAG scheduler divides o/p graph into (map & reduce) Stages.

Stage :- Comprised of tasks based on partitions of i/p data

DAG scheduler pipelines operators together to optimize graph.

↓

*

↓

Final result of DAG scheduler is set of stages.

↓

stages are passed to Task scheduler

↓

TS launches tasks via cluster manager (spark standalone / Yarn / mesos)

↓

Worker executes the tasks.