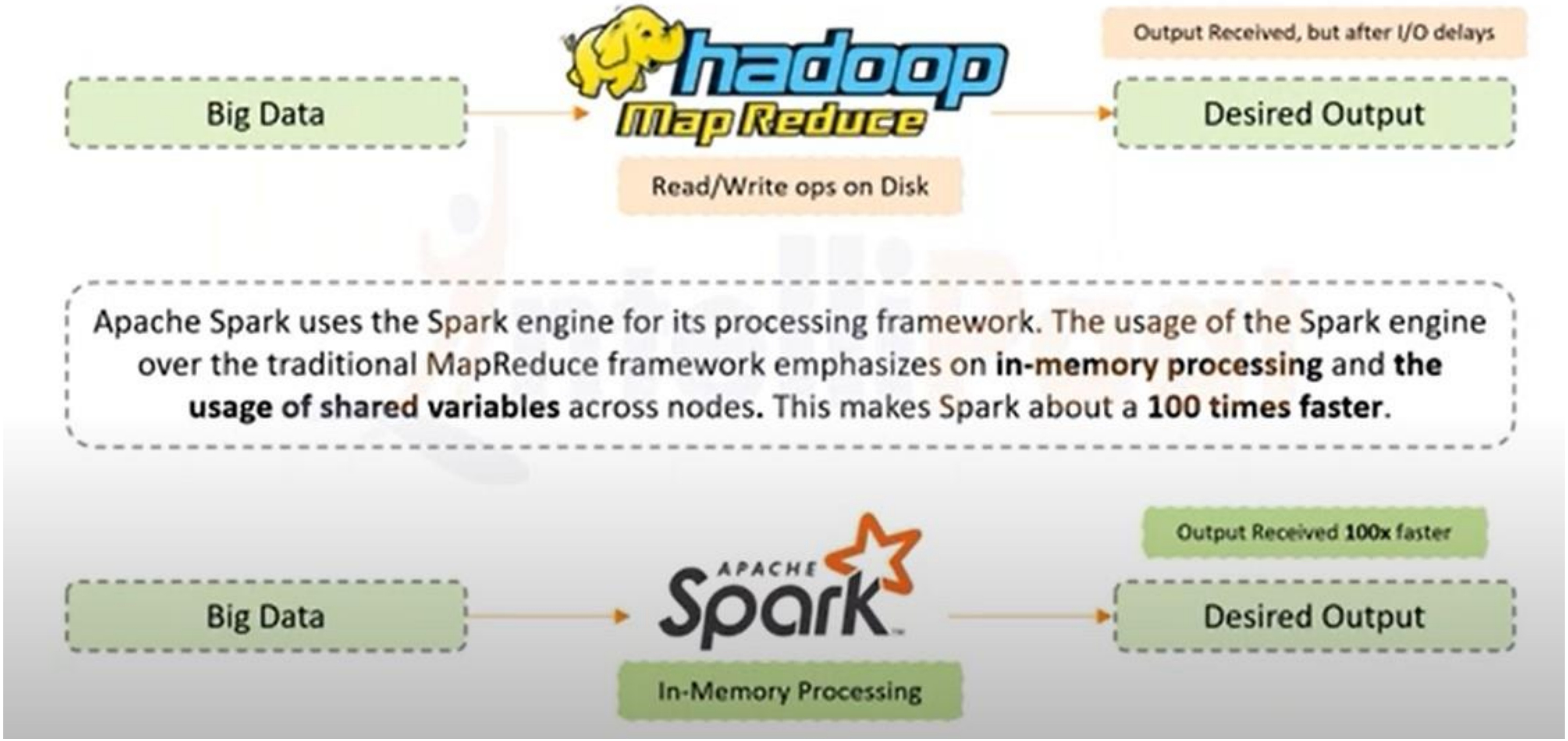
Spark

- Its used for processing
- You are free to use any File system (Storage) and resource manager
- Previously we used Map Reduce for processing.
- Its in-memory computation
- Needs high end machines in cluster

- --
- HDD
- SSD



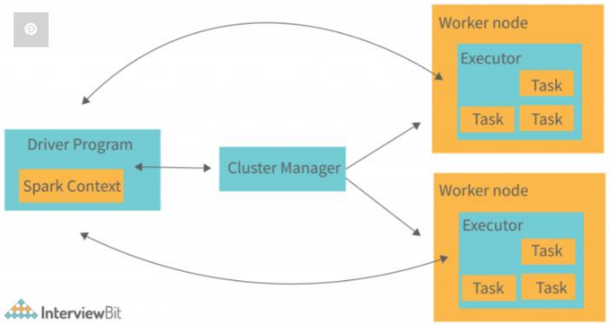
Apache Spark uses the Spark engine for its processing framework. The usage of the Spark engine over the traditional MapReduce framework emphasizes on **in-memory processing** and **the usage of shared variables** across nodes. This makes Spark about a **100 times faster**.

Difference between Hadoop and Spark

- This can not be compared whereas we compare spark and MapReduce

MapReduce	SPARK
Mainly Restricted For Java Developers	Java , Scala, python, R, SQL closure
Boiler Plate Coding	Conciseness
No Interactive Shell	REPL(Read evaluate print loop)
Disk Based, Performance is Slow	Memory based only
Only For Batch Processing	Batch as well as interactive processing
Not Optimized For Iterative Algorithm	Best for iterative algorithms, No Graph
No Graph Processing	Graph processing is supported

Spark Architecture



- Spark works on master slave
- Every application has driver program
- Slaves nodes has executors. One slave can have many executors
- Application master and driver program uses same executor (container).

- Spark supports resource managers like
  - YARN
  - Mesos
  - Standalone
  - Kubernetes

Lineage

- Transformations are not evaluated immediately, they are lazy.
- T will always give you new RDD
- Child RDD has a pointer to parent RDD.
- Like this a graph gets created and its called as Lineage graph
- E.g : read a file → line to line check → filter

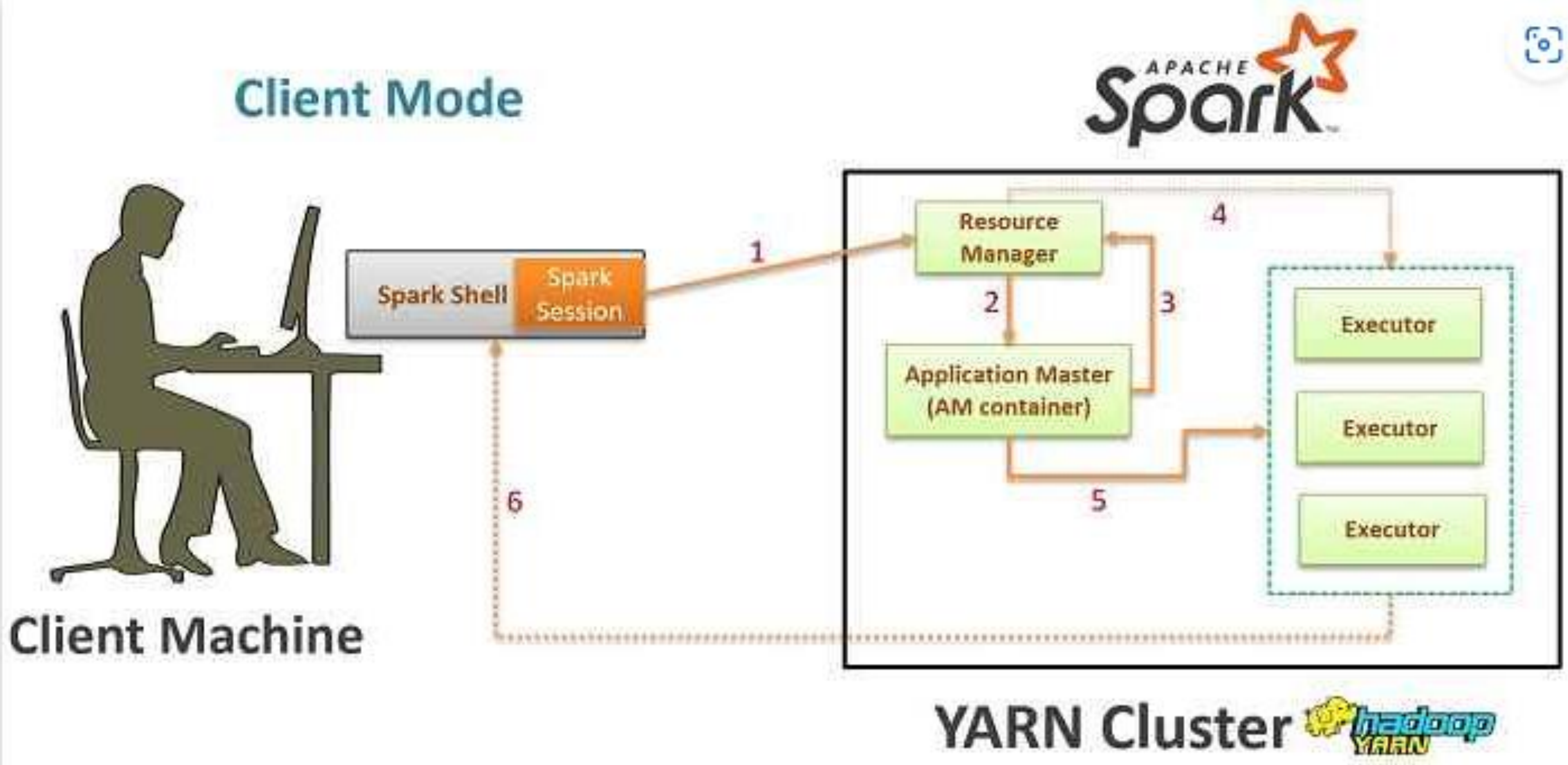
DAG – Directed Acyclic Graph

- It’s a combination of vertices and edges.
- Vertices is RDD and edges represents operations

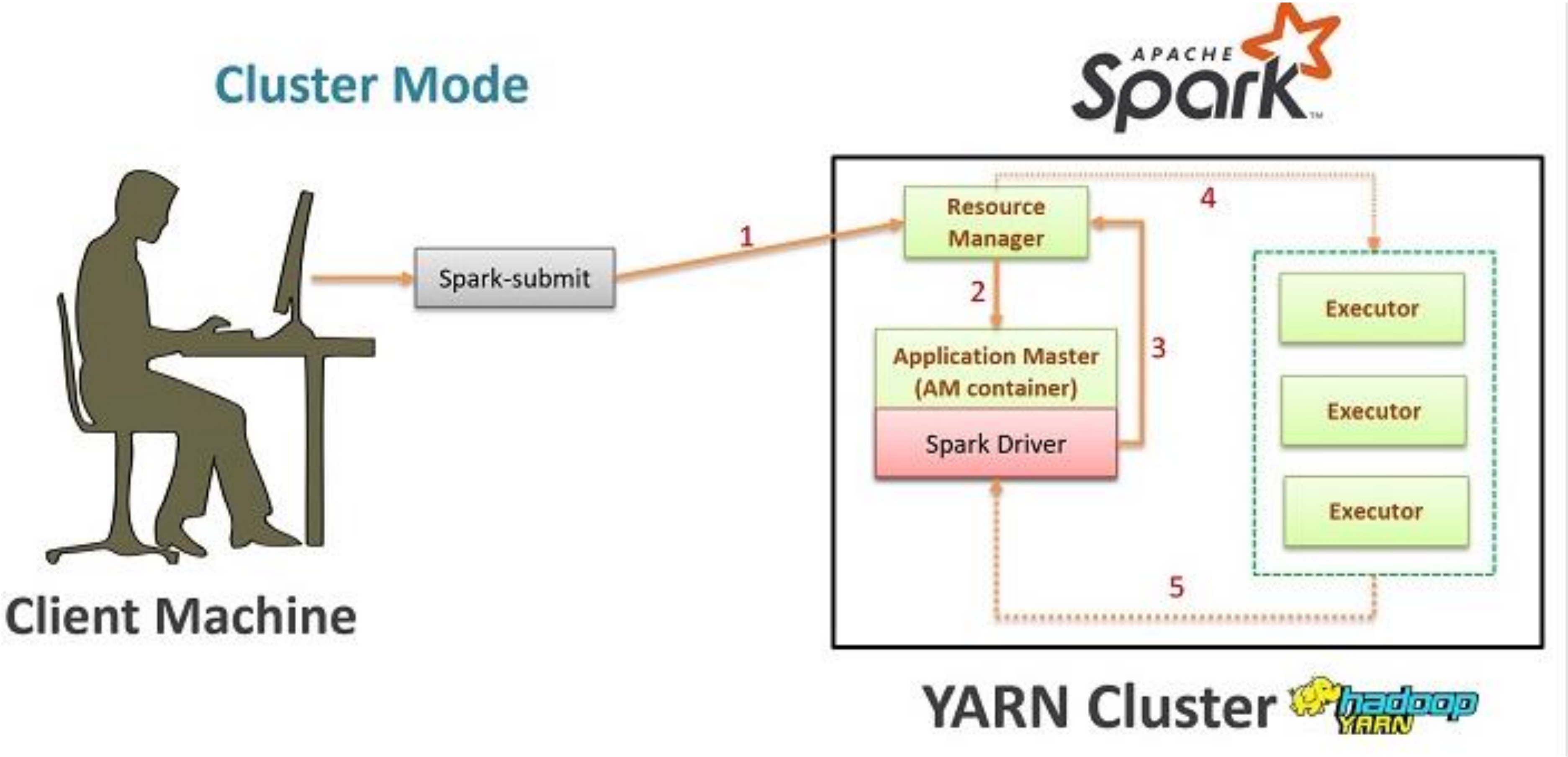
- RDD : Low level API
- DataFrame : High Level API
- Dataset : High Level API
- Spark SQL :High level API

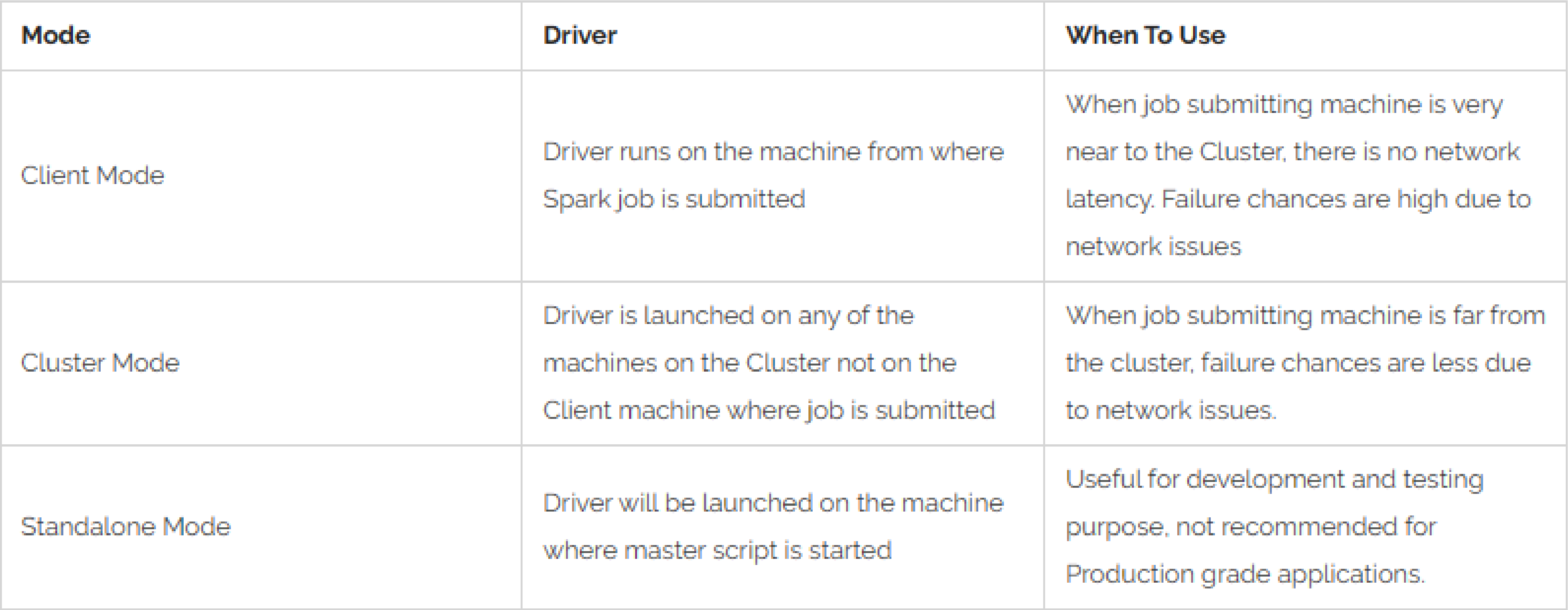
Client Modes

- Driver program runs on the client machine.
- Client machines needs to be running till you get the output



Cluster Mode

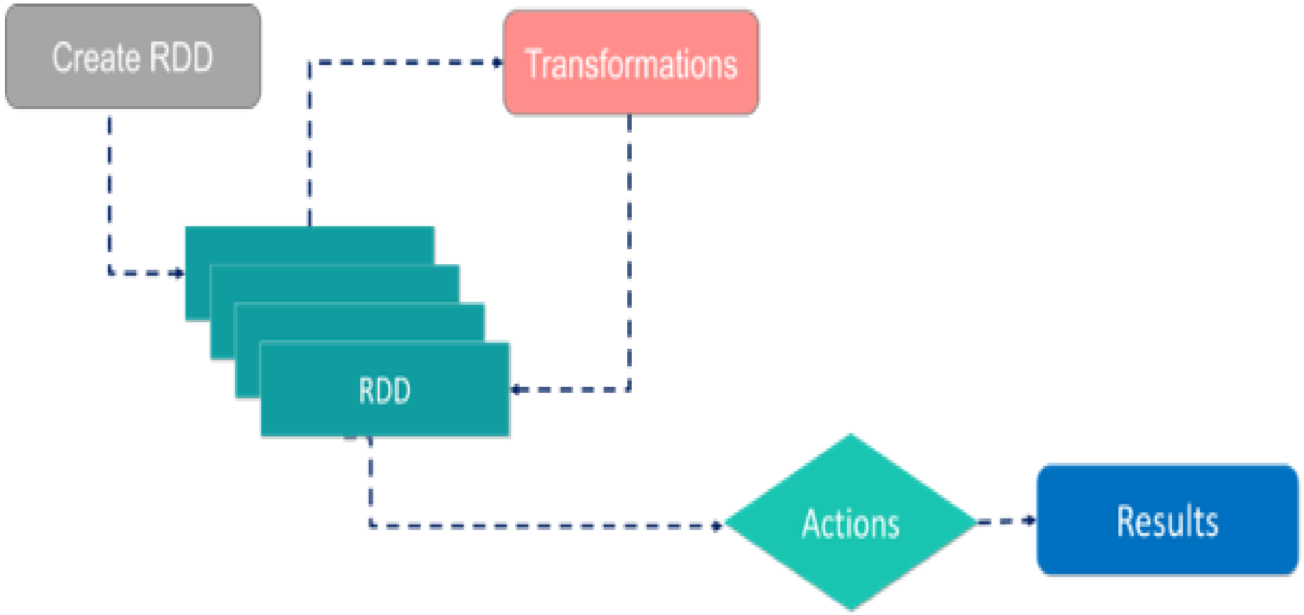




Mode	Driver	When To Use
Client Mode	Driver runs on the machine from where Spark job is submitted	When job submitting machine is very near to the Cluster, there is no network latency. Failure chances are high due to network issues
Cluster Mode	Driver is launched on any of the machines on the Cluster not on the Client machine where job is submitted	When job submitting machine is far from the cluster, failure chances are less due to network issues.
Standalone Mode	Driver will be launched on the machine where master script is started	Useful for development and testing purpose, not recommended for Production grade applications.

Types of function in spark

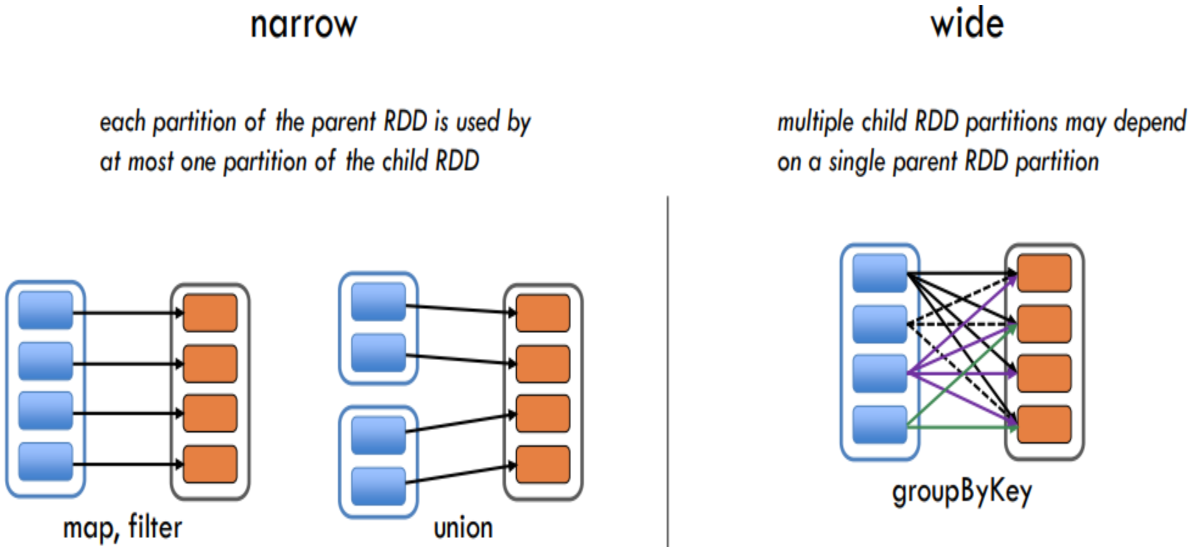
- Transformation :
    - lazy.
    - If a function returns you a new RDD from existing RDD
- Actions :
    - Eager.
    - It returns data in any form, other than RDD.
    - Output data is brought to driver node and result is given as output.



Workflow of RDD

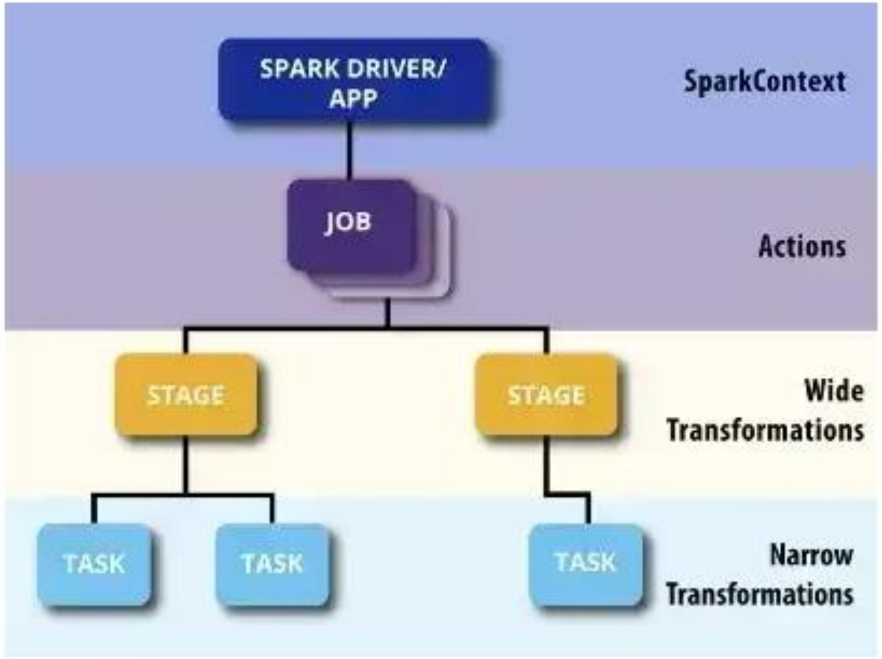
Types of transformations in spark

- Narrow :
    - On same node. NO shuffling involved.
    - One parent(input) and one child (output)
- Wide :
    - Shuffling is involved because the data is distributed and needs to be collected at one executor.
    - One child partition is dependent on multiple parent


narrow
wide
each partition of the parent RDD is used by at most one partition of the child RDD
multiple child RDD partitions may depend on a single parent RDD partition
map, filter
union
groupByKey

- Run Spark program
  - Action → creates new job
  - Wide transformation → creates new stage inside a job
  - Narrow transformation → task inside a stage

  - Program → Jobs → stages → tasks



Dataframe

Repartition	**Coalesce**
It can increase or decrease the number of partitions	It can only decrease the number of partitions
Output file size will be equal	Output file size will be unequal
Invokes Full shuffling	No shuffling – minimum shuffling
slow	fast
Creates new partitions	Uses the existing partition

SparkContext	SparkSession
Low level api - RDD	High level api – Dataframe, Dataset, spark sql
Is available since v1	It was introduced in V2
Part of spark core	Part of spark-sql
Every time you have to create new object SQLContext, HiveContext	It has all contexts inbuilt

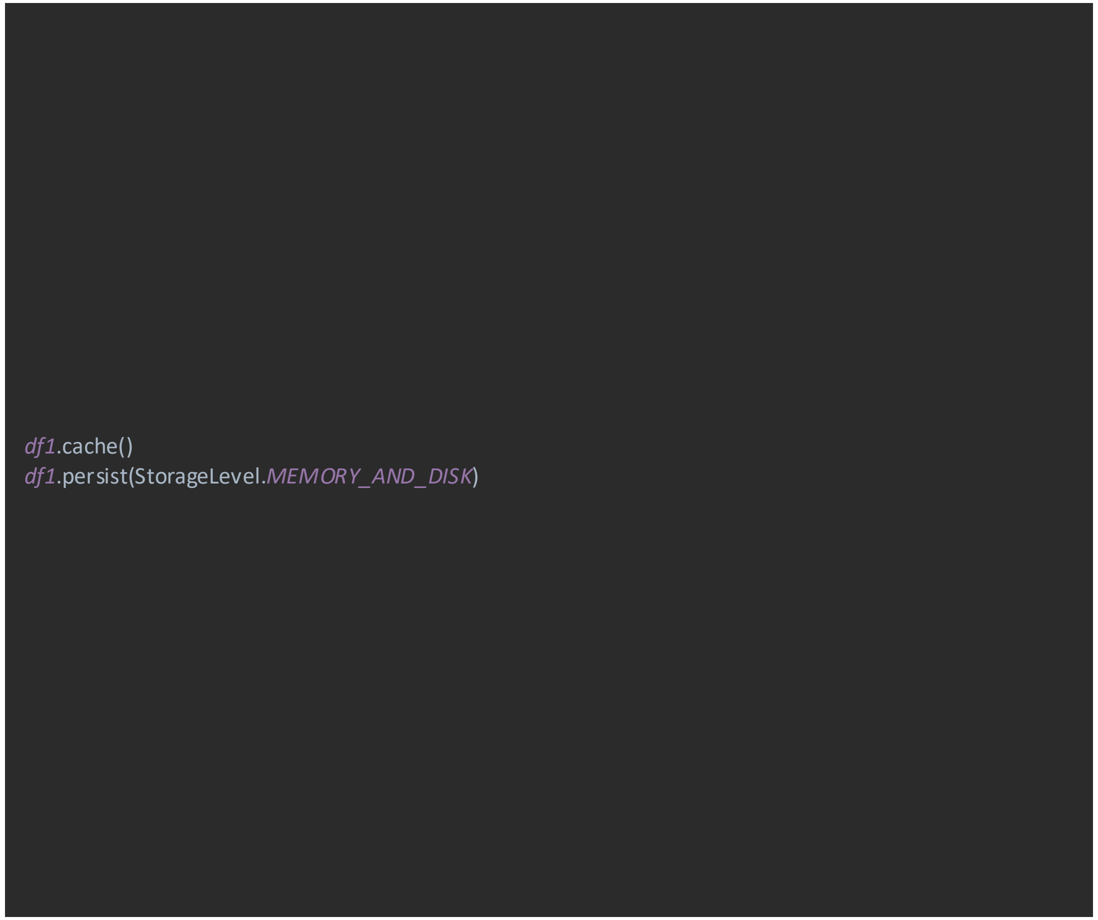
- Design pattern: proven solutions for common problems
- Singleton
- Builder

- Big data specific design pattern
- ETL
- ELT

Dataframe	Dataset
Untyped	Typed
If wrong column name used then you will get error at runtiime	If wrong column name used then you will get error at compiletime
Checks schema at later stage	Checks schema at early stage
Used in scala , python	Only in scala
	Its for developers benefit

- 1. Try word count program using dataframe
- 2. SQL challenge queries 1 to 6 with spark dataframe.
- 3. Understand ranking code shared on chat.
- // Define window specification
- val windowSpec = Window.partitionBy("custid").orderBy("orderdate")
- // Calculate cumulative sum of orders for each customer
- val dfWithCumulativeOrders = ordersdf.withColumn("cumulative_orders", sum("Amount").over(windowSpec))
- // Calculate rank of each order within each customer's order history
- val dfWithOrderRank = dfWithCumulativeOrders.withColumn("order_rank", rank().over(windowSpec))
- dfWithOrderRank.show()

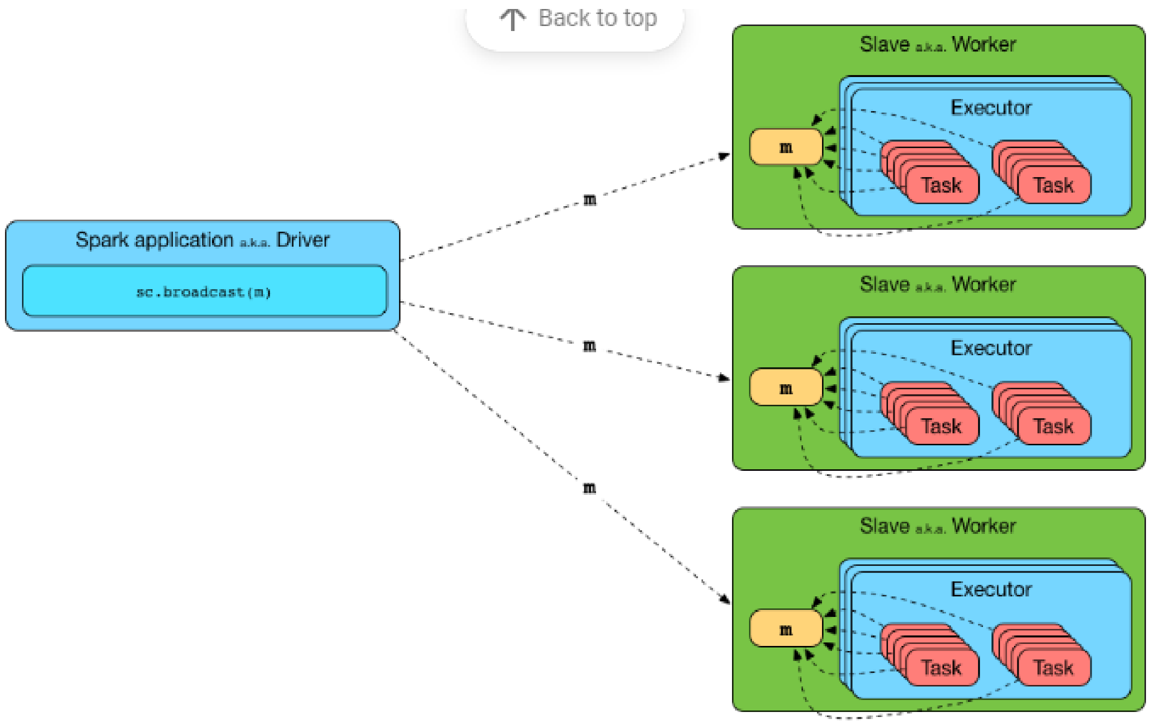
Cache vs persist



```
df1.cache()
df1.persist(StorageLevel.MEMORY_AND_DISK)
```

- Cache hit ratio
- Cache miss ratio

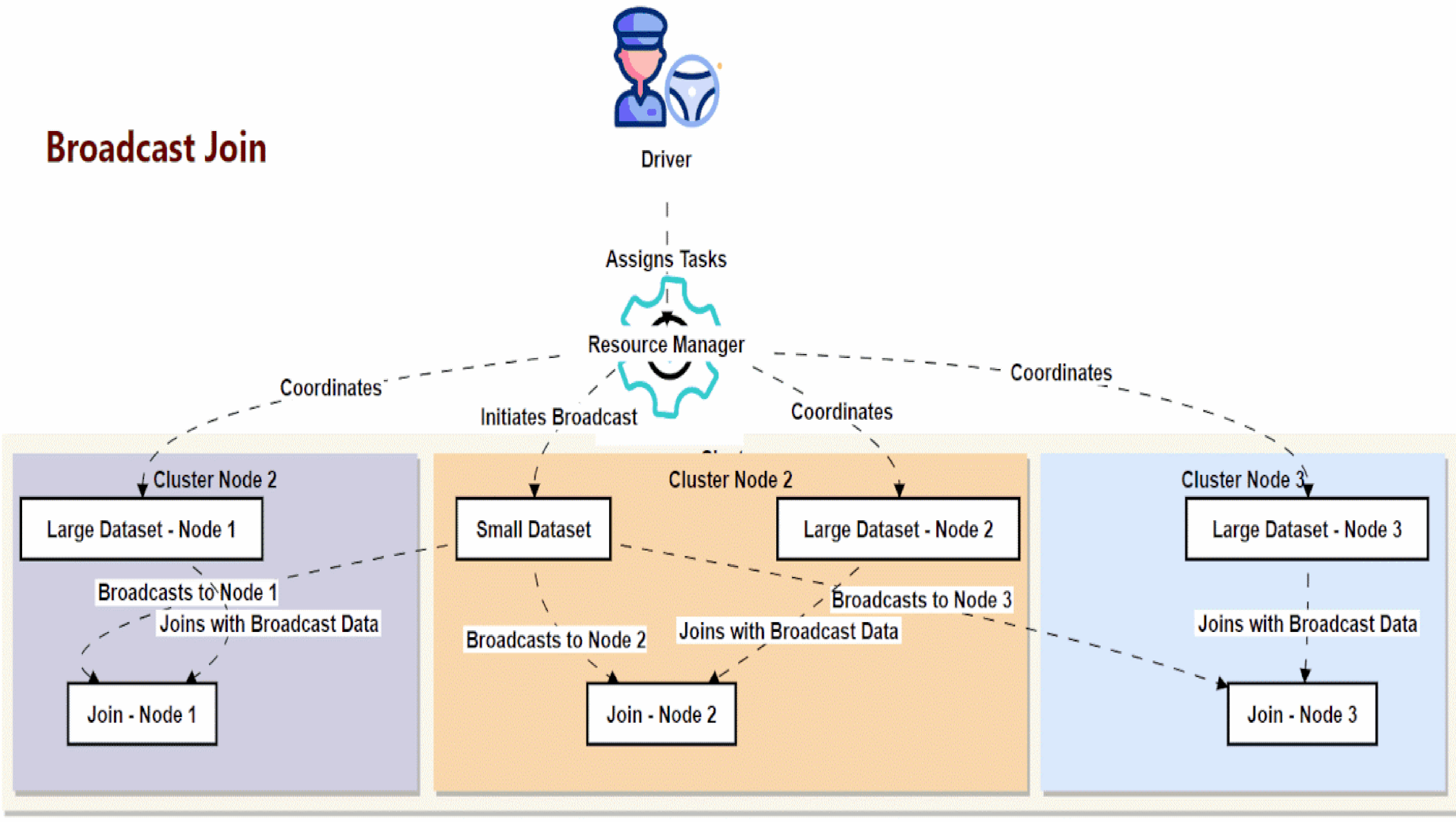
Broadcast join



- One Small and one big table
- sql.autoBroadcastJoinThreshold = 10 mb , 8GB
- spark.conf.set("spark.sql.autoBroadcastJoinThreshold", value_in_bytes)

Normal

Broadcast Join



- 1. 7 to 10 - sql challenge - spark sql
- 2. try to accept two parameters for for input path and second for outputpath for WC program and run on cluster.

-
  ```
  spark-submit --class org.itc.com.Main --master yarn mar2024Demo.jar UKUSMarHDFS/rupali/xyz1.txt
  ```