Module-2 Part-1

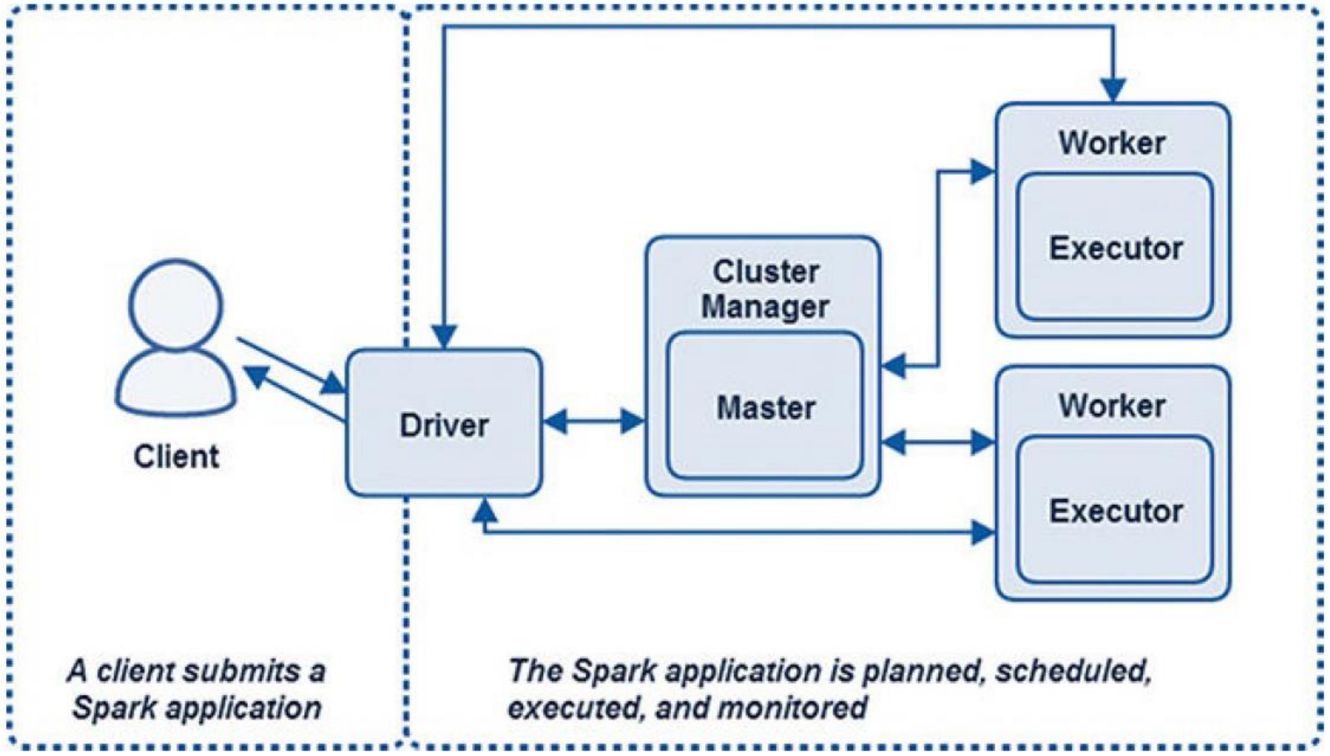
COMPILED BY,

DR. PRAKASH KALINGRAO AITHAL

Components of Spark Application

Driver, the *Master*, the *Cluster Manager*, and the *Executor(s)*, which run on worker nodes, or *Workers*.

Components of Spark Application


Worker
Executor
Cluster Manager
Master
Client
Driver
Worker
Executor
A client submits a Spark application
The Spark application is planned, scheduled, executed, and monitored

Components of Spark Application

All Spark components, including the Driver, Master, and Executor processes, run in Java virtual machines (JVMs). A JVM is a cross-platform runtime engine that can execute instructions compiled into Java bytecode. Scala, which Spark is written in, compiles into bytecode and runs on JVMs.

Components of Spark Application

Spark Driver: The life of a Spark application starts and finishes with the Spark Driver. The Driver is the process that clients use to submit applications in Spark. The Driver is also responsible for planning and coordinating the execution of the Spark program and returning status and/or results (data) to the client. The Driver can physically reside on a client or on a node in the cluster

Components of Spark Application

Spark Workers and Executors: Spark Executors are the processes on which Spark DAG tasks run. Executors reserve CPU and memory resources on slave nodes, or Workers, in a Spark cluster. An Executor is dedicated to a specific Spark application and terminated when the application completes. A Spark program normally consists of many Executors, often working in parallel.

Typically, a Worker node—which hosts the Executor process—has a finite or fixed number of Executors allocated at any point in time. Therefore, a cluster— being a known number of nodes—has a finite number of Executors available to run at any given time. If an application requires Executors in excess of the physical capacity of the cluster, they are scheduled to start as other Executors complete and release their resources.

Components of Spark Application

The Spark Master and Cluster Manager: The Master and the Cluster Manager are the central processes that monitor, reserve, and allocate the distributed cluster resources (or containers, in the case of YARN or Mesos) on which the Executors run. The Master and the Cluster Manager can be separate processes, or they can combine into one process, as is the case when running Spark in Standalone mode.

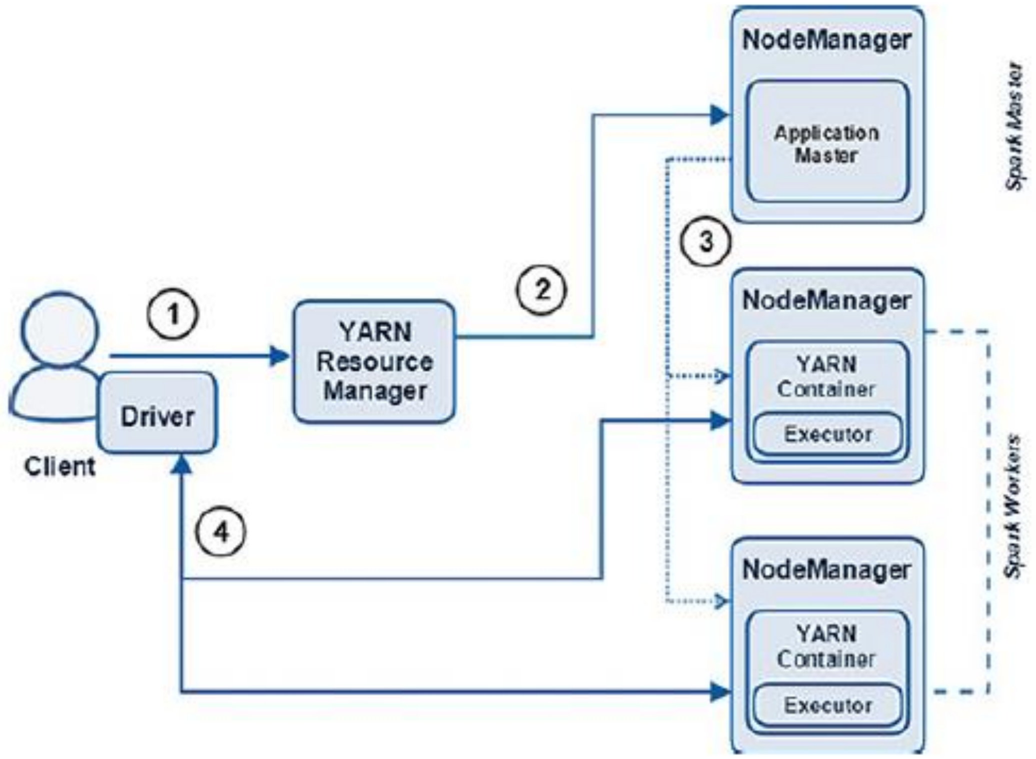
Deployment Modes of Spark Application

Client mode

Cluster mode

Client Mode

Client mode.



Client Mode

1. The client submits a Spark application to the Cluster Manager (the YARN ResourceManager). The Driver process, SparkSession, and SparkContext are created and run on the client.

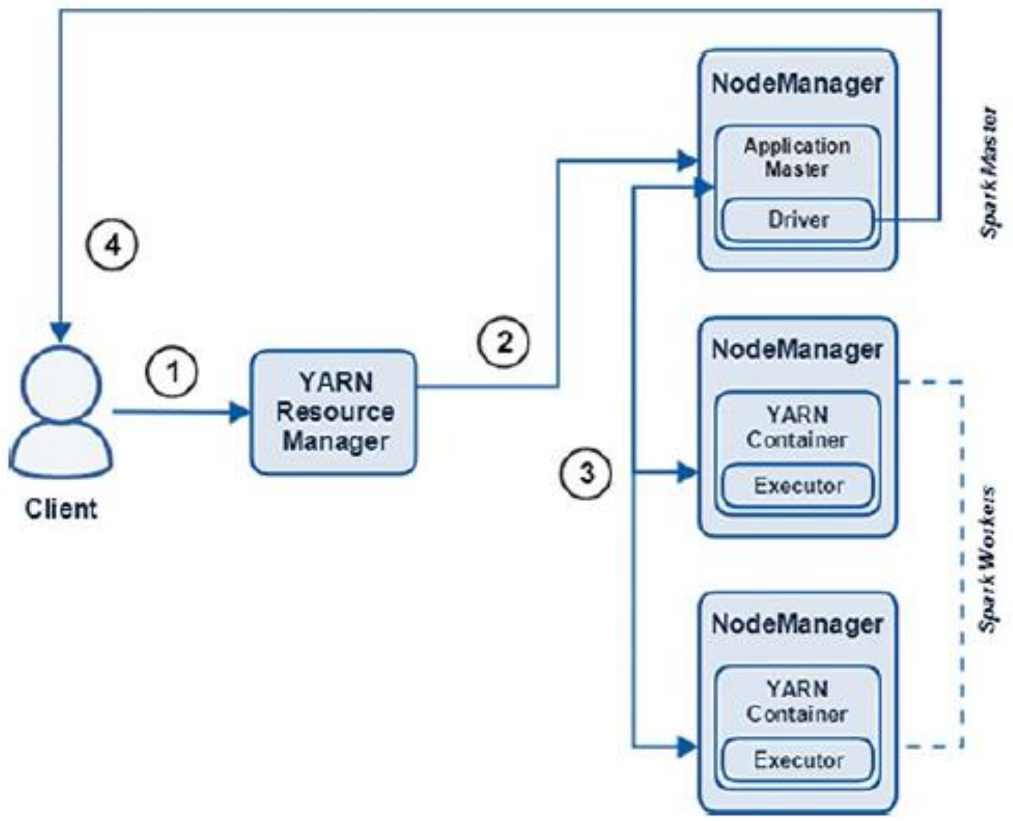
2. The ResourceManager assigns an ApplicationMaster (the Spark Master) for the application.

3. The ApplicationMaster requests containers to be used for Executors from the ResourceManager. With the containers assigned, the Executors spawn.

4. The Driver, located on the client, then communicates with the Executors to marshal processing of tasks and stages of the Spark program. The Driver returns the progress, results, and status to the client.

Cluster Mode

Cluster mode.



Cluster Mode

1. The client, a user process that invokes spark-submit, submits a Spark application to the Cluster Manager (the YARN ResourceManager).

2. The ResourceManager assigns an ApplicationMaster (the Spark Master) for the application. The Driver process is created on the same cluster node.

3. The ApplicationMaster requests containers for Executors from the ResourceManager. Executors are spawned within the containers allocated to the ApplicationMaster by the ResourceManager. The Driver then communicates with the Executors to marshal processing of tasks and stages of the Spark program.

4. The Driver, running on a node in the cluster, returns progress, results, and status to the client.

References

Jeffery Aven, *Data Analytics with Spark using Python*, Pearson, 2018