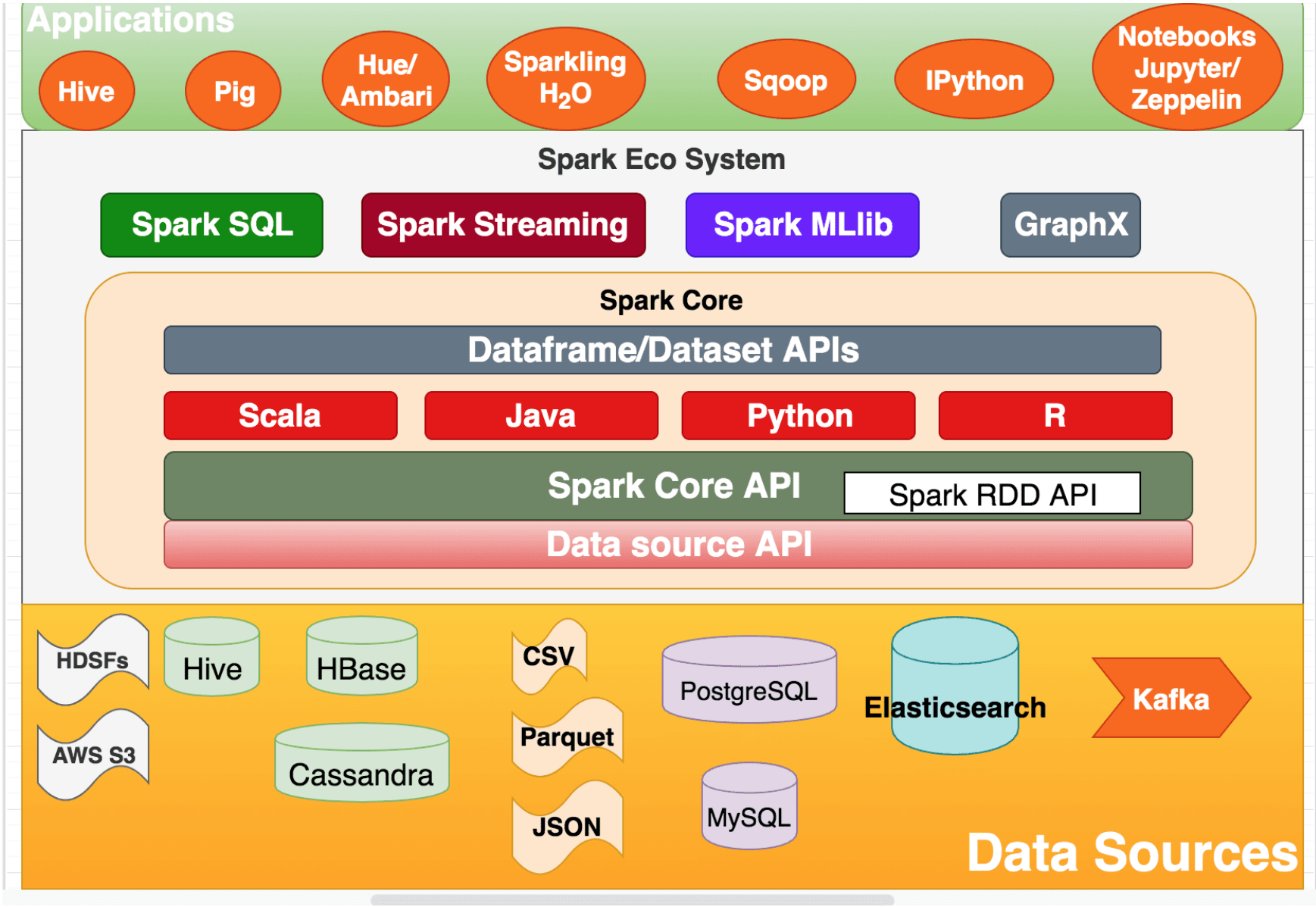
00: Apache Spark eco system & anatomy interview questions and answers

Q01. Can you summarise the Spark eco system?

A01. Apache Spark is a general purpose **cluster computing** system. It provides high-level API in Java, Scala, Python, and R. It has **6 components** Core, Spark SQL, Spark Streaming, Spark MLlib, Spark GraphX, and SparkR. All the functionalities being provided by Apache Spark are built on the top of **Spark Core**. Spark Core is the foundation of in-memory parallel and distributed processing of huge dataset with fault-tolerance & recovery.


Applications
Hive
Pig
Hue/
Ambari
Sparkling
H_2O
Sqoop
IPython
Notebooks
Jupyter/
Zeppelin
Spark Eco System
Spark SQL
Spark Streaming
Spark MLlib
GraphX
Spark Core
Dataframe/Dataset APIs
Scala
Java
Python
R
Spark Core API
Spark RDD API
Data source API
HDSFs
Hive
HBase
CSV
PostgreSQL
Elasticsearch
Kafka
AWS S3
Cassandra
Parquet
MySQL
JSON
Data Sources

Spark Eco System

The **Spark SQL** component is a distributed framework for structured data processing. **Spark Streaming** is an add on API, which allows scalable, high-throughput, fault-tolerant stream processing of live data streams. Spark can access data from sources like **Kafka**, **Flume**, **Amazon Kinesis** or **TCP socket**. **MLlib** in Spark is a scalable Machine learning library. **GraphX** in Spark is API for graphs. The key component of **SparkR** is SparkR DataFrame.

Q02. What are the key execution components of Apache Spark?
A02. Apache Spark uses **master-slave** architecture, where there will be one master process & multiple slave (aka worker) processes. This master slave architecture is applied at the

1) Cluster Management Level: Application master is the **master** and the Node managers are the slaves. Application master is responsible for coordinating the node mangers to allocate resources like memory & CPU cores.

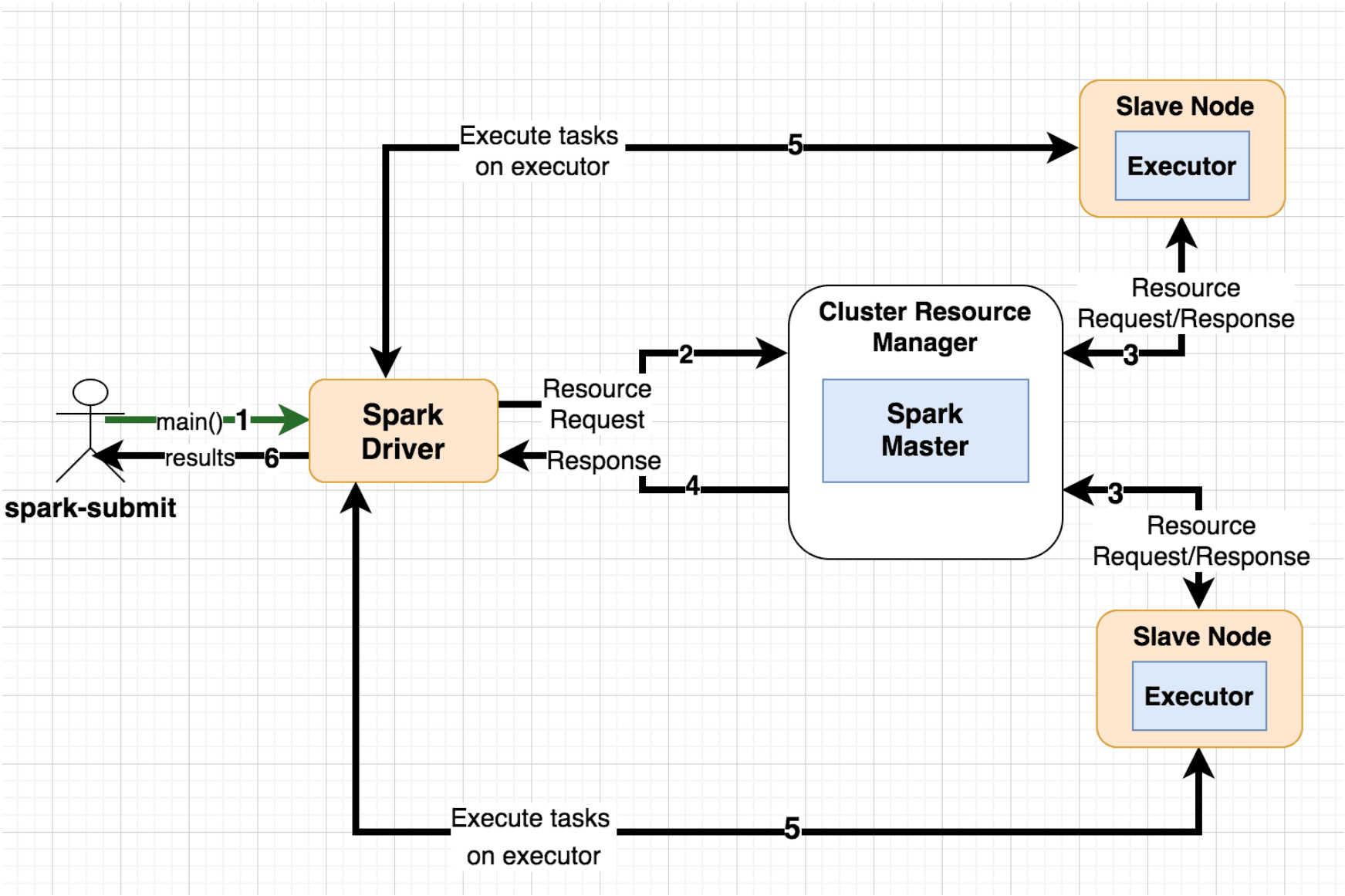
2) Application Level: Spark Driver is the **master** & Spark executors are the **slaves/workers**. Driver is responsible for sending the portion of the logic to each executor.

The key components of Apache Spark are:

1) Spark Driver (**master**)
2) Executors (**slave**)
3) Spark Session (or Spark **Context** prior to Spark 2.0).

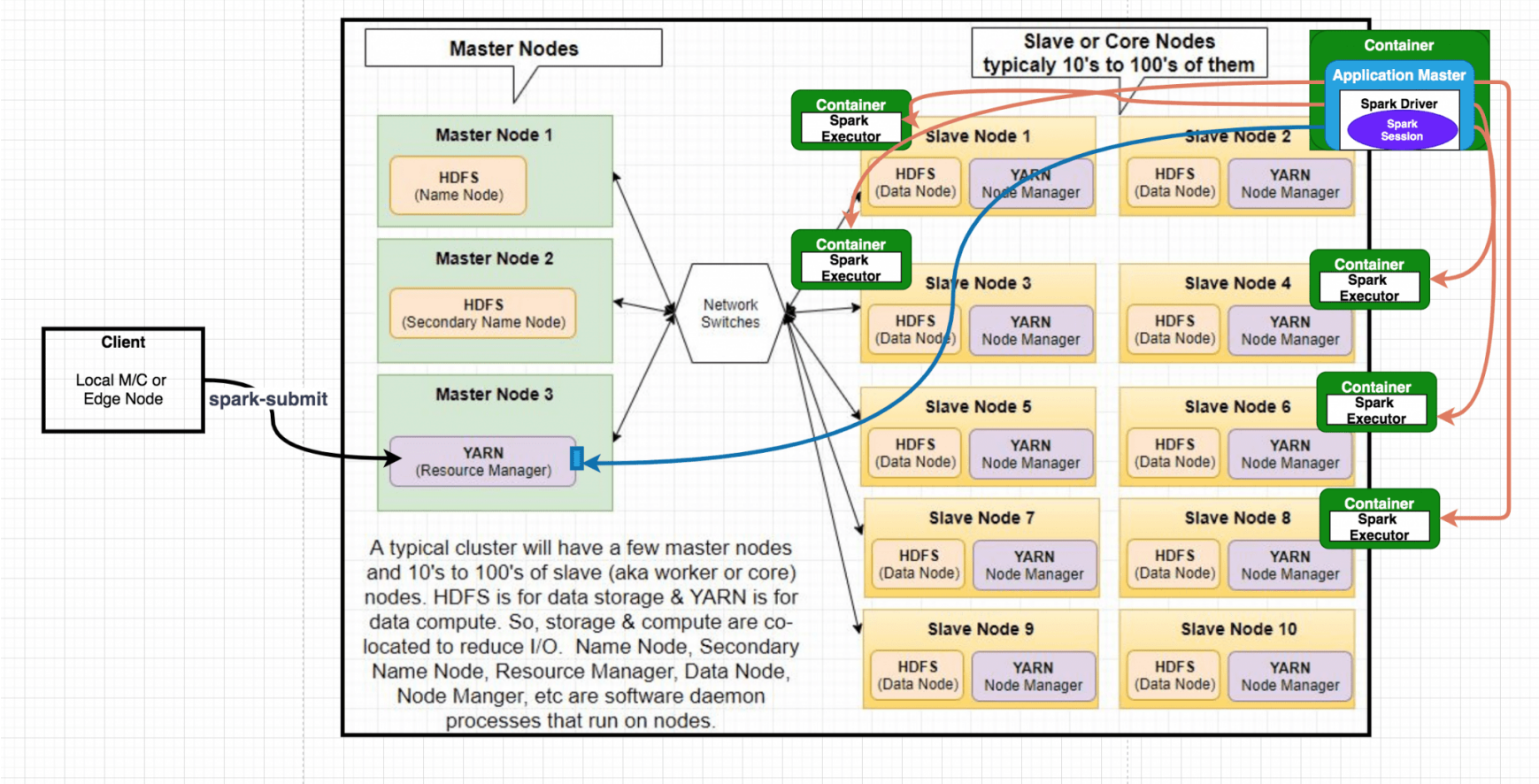
4) Cluster Resource Manager (E.g. **YARN**, Kubernetes, Mesos, Nomad & Spark's own standalone cluster manager).
5) Application Master i.e. Resource Manager (**master**).
6) Node managers i.e. slave node mgr (**slave**).

These components **PLAN**, **SCHEDULE**, **EXECUTE** & **MONITOR** the Spark application.



Apache Spark Execution

As shown below, Apache Spark uses Master/Slave architecture. The slave nodes are also known as the worker nodes or core nodes. A typical cluster will have 100's to 1000's of slave nodes. This is where you have the parallel execution of tasks. Tasks need to be planned, scheduled, queued, executed & monitored. If any executor crashes, its tasks will be sent to different executors to be processed again.



Apache Spark Architecture – Cluster Mode

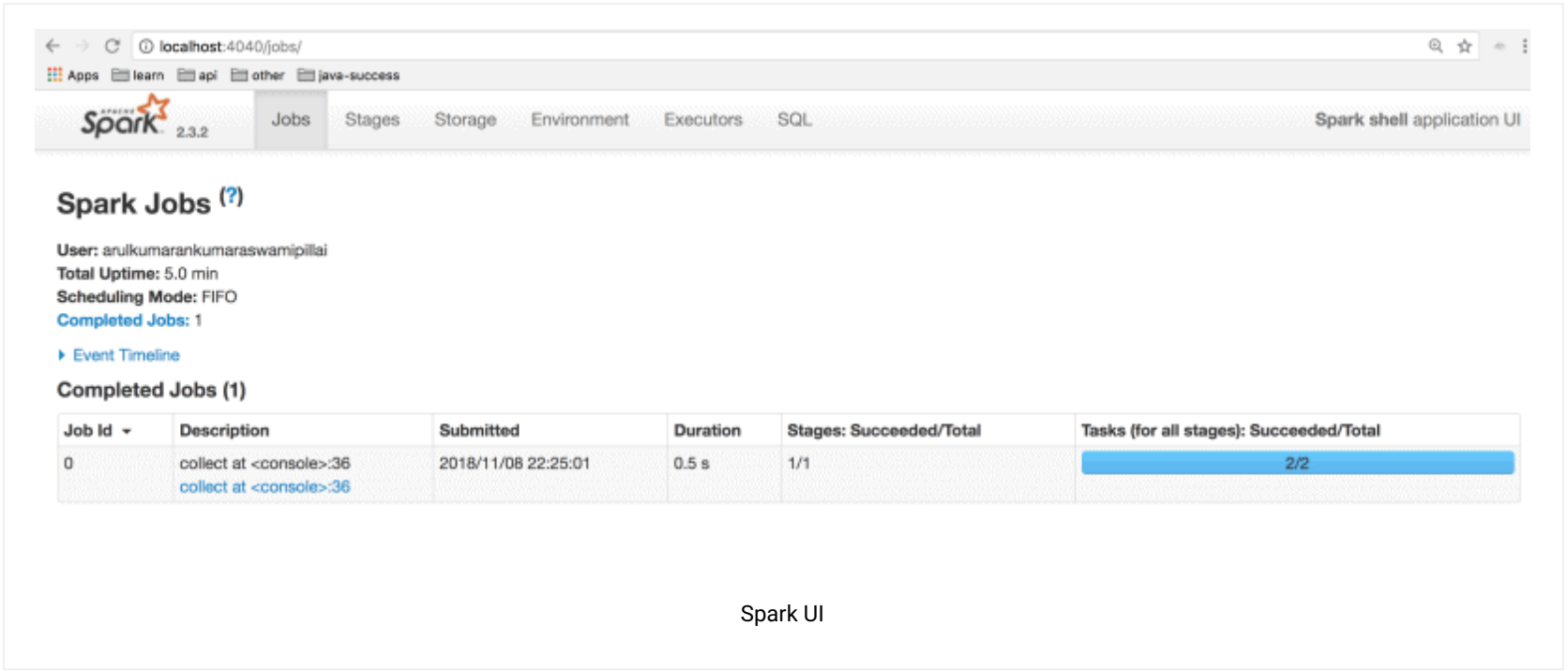
Q03. What is a Driver?

A03. A Spark **Driver** is a process where the main method runs. The Spark driver is the process which the clients used to submit the spark program. First it converts the user program into smaller execution units called **tasks** and after that it **schedules** the tasks on the executors.

For example, A driver initiates "**map**" tasks on the cluster executors against the data in the slave nodes, and each executor returns a subset of the data back to the Driver as a "**reduce**" operation to be combined & returned back to the client as a final result.

A Spark Driver contains components like DAGScheduler, TaskScheduler, etc responsible for converting user code into Spark jobs to be executed on the cluster. A Driver contains metadata of all the RDDs & their partitions.

The Spark driver runs on the port 4040 and UI is created automatically once the user submits the spark program to the spark driver. Sparkdriver:4040/jobs/



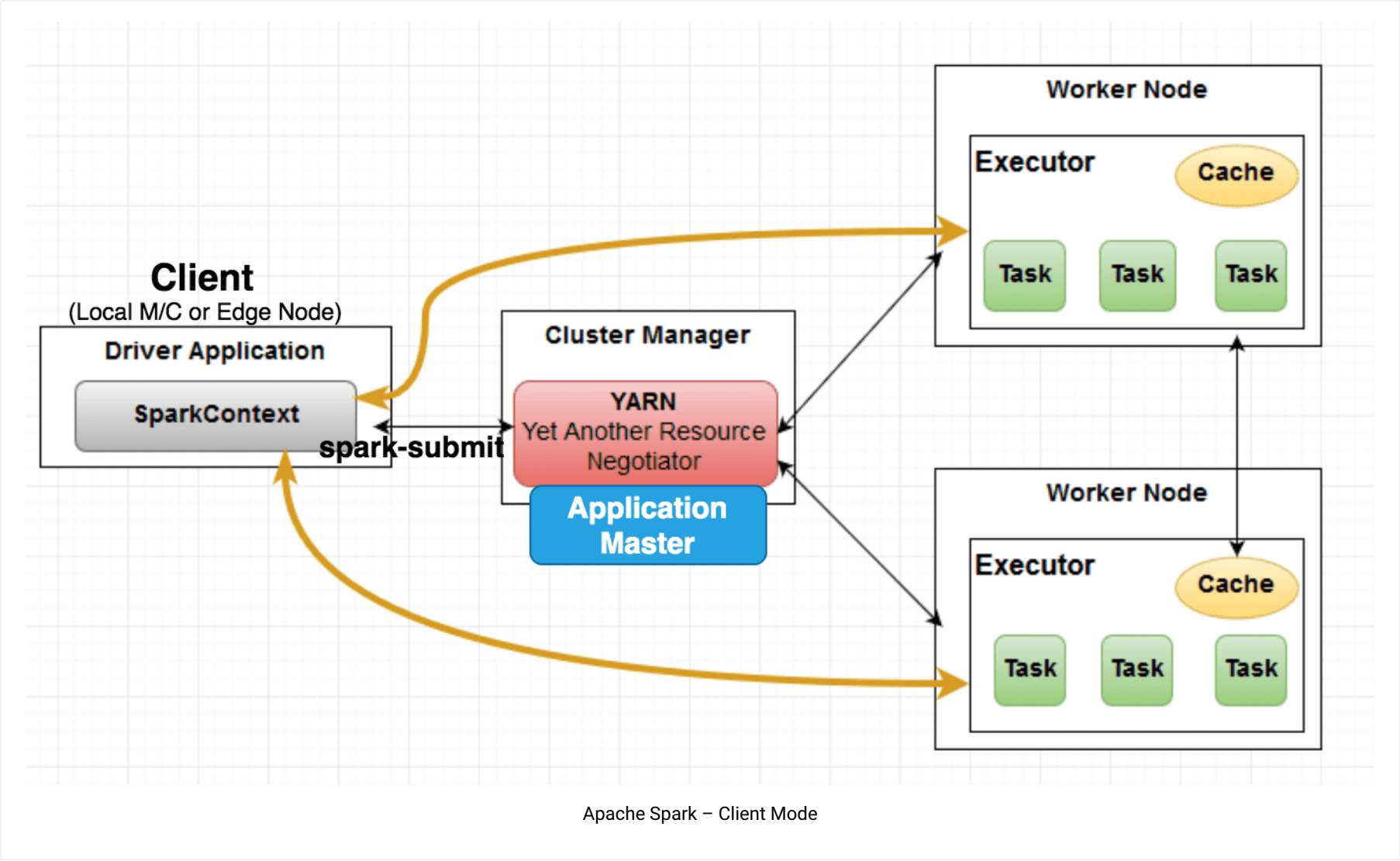
Spark UI

Q04. What are the different Spark modes of execution?

A04. **Client Mode** & **Cluster Mode**. These modes change the behavior as to where the "Driver" runs.

In Client Mode a **Driver** component of spark job will run on the machine from which a job is submitted. For example, your local machine.

In a **client mode**, the Spark master plays the role of Cluster manager. Spark master negotiates the resources with slave (aka worker) nodes and tracks their status & monitor the progress. It also makes the resources available to spark driver.



Apache Spark – Client Mode

In Cluster Mode job submitting machine is remote from “spark infrastructure” as shown in the cluster diagram. The job will be submitted from a local machine or an edge node, but the Driver will be running in the cluster.

spark-shell should be used for **interactive** queries as it needs to be run in yarn-client mode so that the machine you're running on acts as the driver. For spark-submit, you submit jobs to the cluster then the task runs in the cluster.

YARN client mode

```
1
2 $ spark-shell --master yarn
3 $ spark-shell --master yarn --deploy-mode client
4
```

YARN cluster mode

```
1
2 $ spark-submit --class com.myapp.MySparkApp myspark.jar yarn-client
3 $ spark-submit --class com.myapp.MySparkApp myspark.jar yarn-cluster
4
```

You can also run Spark in local mode. This is a non-distributed single JVM deployment mode, where Spark spawns all the execution components – driver, executor, and master in the same single JVM. This is the only mode where a driver is used for execution.

```
1
2 local[*],local,local[2]…etc
3 ...
```

```
$ spark-shell --master local[1]
$ spark-submit --class com.myapp.MySparkApp myspark.jar local[1]

```

Spark distribution comes with its own resource manager. When your program uses spark's resource manager, execution mode is called **Standalone cluster mode**.

```

$ spark-shell --master spark://hduser:7077
$ spark-submit --class com.myapp.MySparkApp myspark.jar spark://hduser:7077

```

The only difference between Standalone mode and local mode is that in Standalone mode you are defining "containers" for the worker and spark master to run in your machine, but in local mode you are just running everything in the same JVM in your local machine.

Q05. What is an Application Master in Spark?
A05. An **Application Master** is the process that requests resources from the cluster and make these available to the spark driver in-turn to execute the tasks in the executors. It is created on the same node as the Driver in the **cluster mode** when **spark-submit** is invoked. Each Spark application will have its **own** dedicated Application Master.

In a **client mode**, the Spark master plays the role of Cluster manager. Spark master negotiates the resources with slave nodes and tracks their status & monitor the progress. It also makes the resources available to spark driver.

Q06. What is a Spark Session or Spark Context?
A06. A "SparkContext" is the **main entry point** for a Spark job prior Spark version 2.0. Starting from Apache Spark **2.0**, Spark Session is the new entry point for Spark applications. A Spark context is created by the Spark driver for each individual Spark programs when it is first submitted by the user.

Q07. What is a Cluster Resource Manager?
A07. In a distributed computing, a **cluster resource manager** is responsible for monitoring the **containers** in the slave nodes and reserving the resources on these nodes upon request by the application master. The application master in turn makes these resources available to the spark driver program to execute the tasks and stages in executors. These containers are reserved based on the needs of the executors.

A **SparkSession** can connect to any cluster resource manager like YARN, Kubernetes, Mesos, etc.

Q08. What are the Spark executors?
A08. Spark **executors** in the slave (aka worker or core) nodes are responsible for executing the assigned tasks. The results of each task are returned to the Spark Driver. Executors can be **statically allocated** via spark-submit arguments or **dynamically allocated** based on the overall work load by adding & removing executors. The dynamic allocation can adversely impact other spark jobs running in the cluster.

Executors only know of the tasks allocated to them and it's the responsibility of the spark driver to coordinate a set of tasks with the correct dependencies.

Q09. How do you know which piece of code runs on driver or executor?
A09. A Spark application consists of a **single** Driver process and **one or more** Executor processes. Driver process is

responsible for a lot of things including directing the overall control flow of your application, restarting failed stages and the entire high level direction of how your application will process the data.

You can increase or decrease the number of Executors dynamically depending upon your usage, but the Driver will exist throughout the lifetime of your application.

Which part of code runs in a driver vs executor?

As a rule of thumb everything that is executed inside functions like map, filter, flatMap, combineByKey, etc should be handled by **executor nodes**. Everything outside these are handled by the **driver**.

```
1
2  from pyspark.sql import SparkSession
3
4  # Runs in Driver
5  conf = SparkConf().setAppName(appName).setMaster(master)
6  sc = SparkSession\
7          .builder\
8          .appName("PythonWordCount")\
9      .config(conf=conf)
10         .getOrCreate()
11
12 # Runs in Driver. Driver splits linesRDD into tasks to be run in Executors
13 # Driver will send tasks to executors via Cluster Manager
14 linesRDD = sc.textFile("hdfs://...")
15
```

```
# Runs in executors as parallel tasks
wordsRDD = linesRDD.flatMap(lambda line: line.split(" ")

# Runs in executors as parallel tasks
wordCountRDD= wordsRDD.map(lambda word: (word, 1))

# Runs in executors as parallel tasks.
resultRDD = wordCountRDD.reduceByKey(lambda a, b: a + b)

# Runs in executors
resultRDD.saveAsTextFile("hdfs://...")

# Runs in Driver
spark.stop()

```


‹ 02: 30+ Java architect interview questions & answers – Part 2

Spark join strategies & performance tuning interview questions and answers ›