Overview of Spark and its ecosystem

CS3678 Big Data and Hadoop

Team Members:

Sahana Priya S, 22011102088, B. Tech CSE (IoT) - B, 3rd year

Shruti Thiagu, 22011102100, B. Tech CSE (IoT) - B , 3rd year

V Harsha Vardhana Anand, 2201110114, B. Tech CSE (IoT) - B, 3rd year

Department of Computer Science and Engineering
School of Engineering

Shiv Nadar University Chennai

SHIV NADAR UNIVERSITY CHENNAI

Overview of Apache Spark

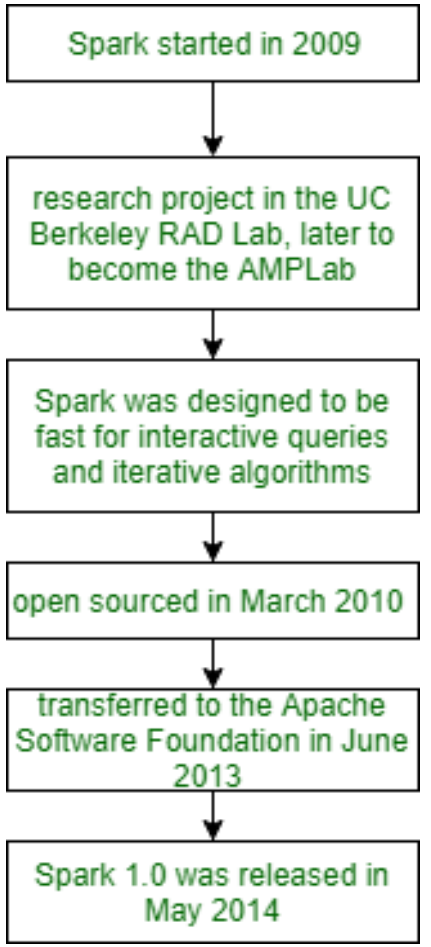
Developed in 2009 at UC Berkeley's AMP Lab for **fast iterative processing.**

Apache Spark

- **Open-source**
- **Distributed computing framework** for big data processing and analytics.
- **Faster In-memory computation**
- Supports **batch processing, real-time streaming, machine learning, and graph processing**.

How Spark Works

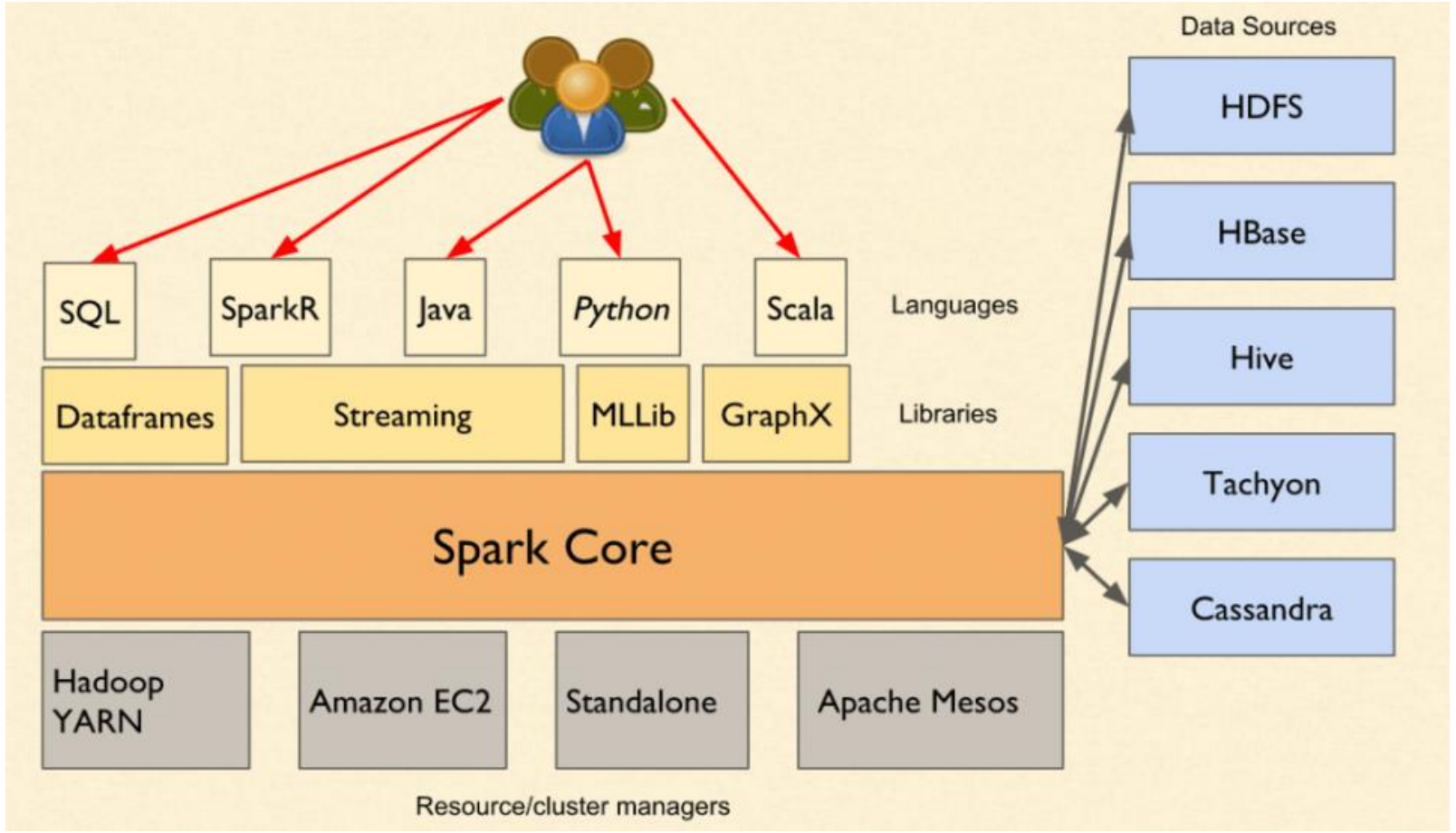
- Uses **in-memory computation** - reduce disk I/O, makes it faster than Hadoop.
- **Resilient Distributed Datasets (RDDs)** - Parallel processing & Fault tolerance
- **Data Frames** and **Datasets** optimize query execution.
- Supports various **storage** systems like
  - HDFS
  - Amazon S3
  - Cassandra
  - Redshift.



SHIV NADAR UNIVERSITY CHENNAI

Spark Core and Architecture

Master-slave Architecture



Components:

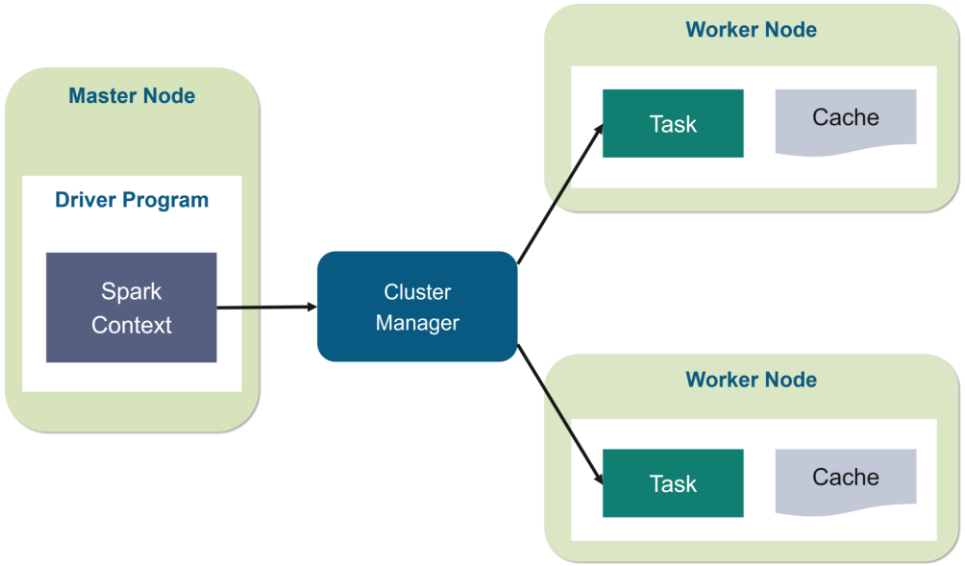
1. Scheduler or Resource Manager
2. Spark Driver & Executors
3. Spark Core (RDD)
4. Spark Libraries
5. API Bindings & Interactive Shells

- **RDD (Resilient Distributed Dataset)** is a fundamental, immutable, partitioned, fault-tolerant data structure that supports parallel processing.
- It enables in-memory computation, lazy evaluation, & transformations like map() and filter().

Spark Core and Architecture

Components:

1. **Scheduler or Resource Manager**
   - Manages cluster resources.
   - Can be **YARN, Mesos, cluster manager**.
2. **Spark Driver & Executors**
   - **Driver**:
     - Runs the main() method of the Spark application.
     - Creates **RDDs, transformations, and actions**.
     - Converts jobs into a **Directed Acyclic Graph** for execution.
   - **Executors**: Runs task assigned in driver & store data in memory for faster execution
3. **Spark Core (RDD)**
   - **Resilient Distributed Datasets (RDDs)**
     - Immutable, distributed data collection.
   - Supports transformations (map, filter) and actions (count, collect).
4. **Spark Libraries:**
   - Spark SQL, Spark Streaming, MLlib, GraphX
5. **API Bindings & Interactive Shells:**
   - APIs in Scala, Java, Python, R & SQL.
   - PySpark for Interactive data analysis.

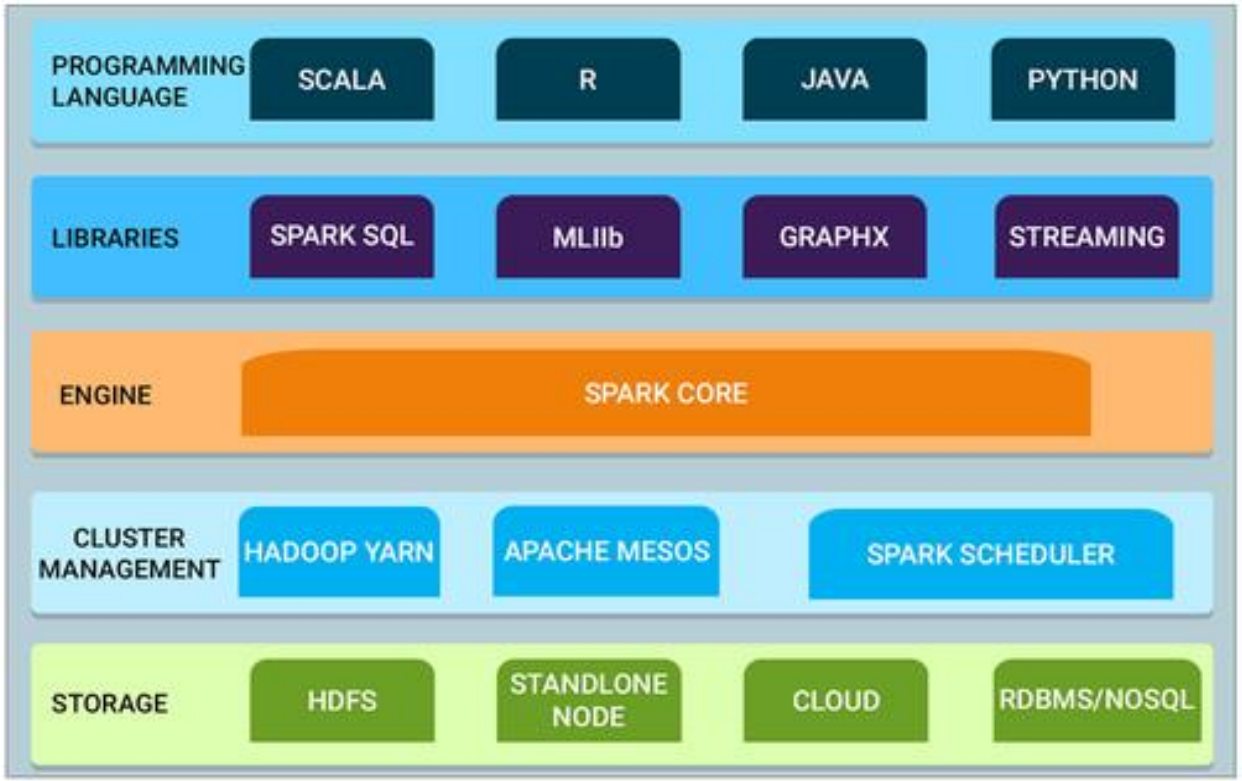


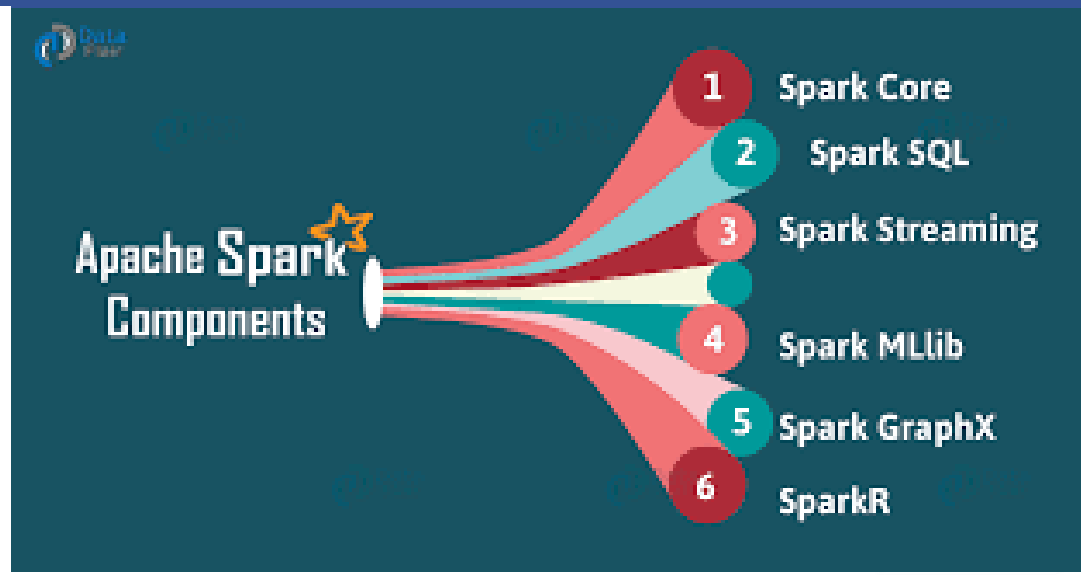
How Spark Applications Run on a Cluster:

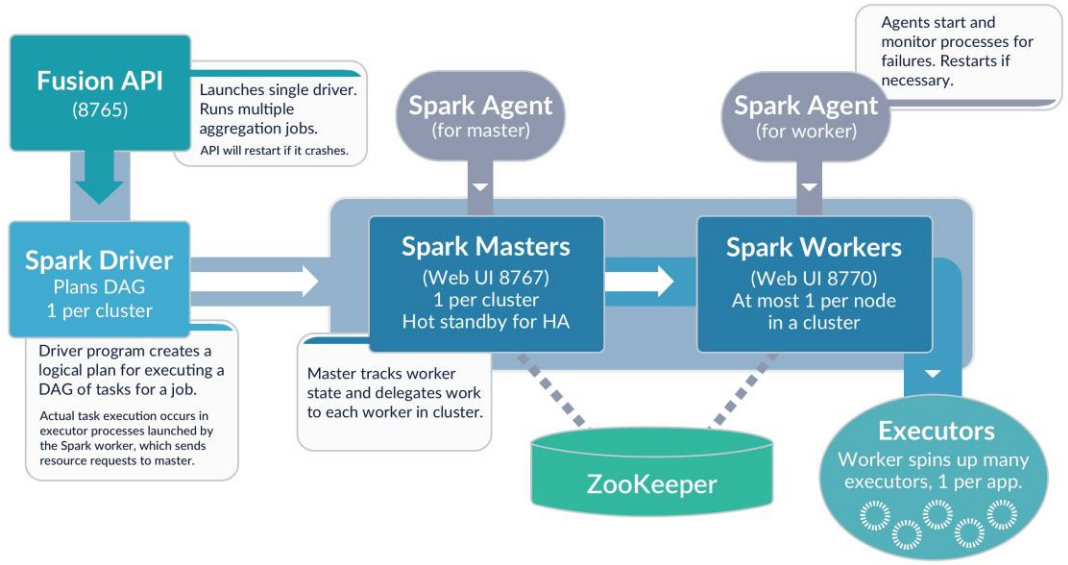
- User submits the Spark job
- Spark Driver starts and requests resources from Cluster Manager.
- Cluster Manager allocates resources, launches executors on workers.
- Driver schedules tasks and sends them to executors.
- Executors process data in parallel and store results in memory / disk
- Results are returned to the driver.

SHIV NADAR UNIVERSITY CHENNAI

Spark Ecosystem Components







Spark Programming and APIs

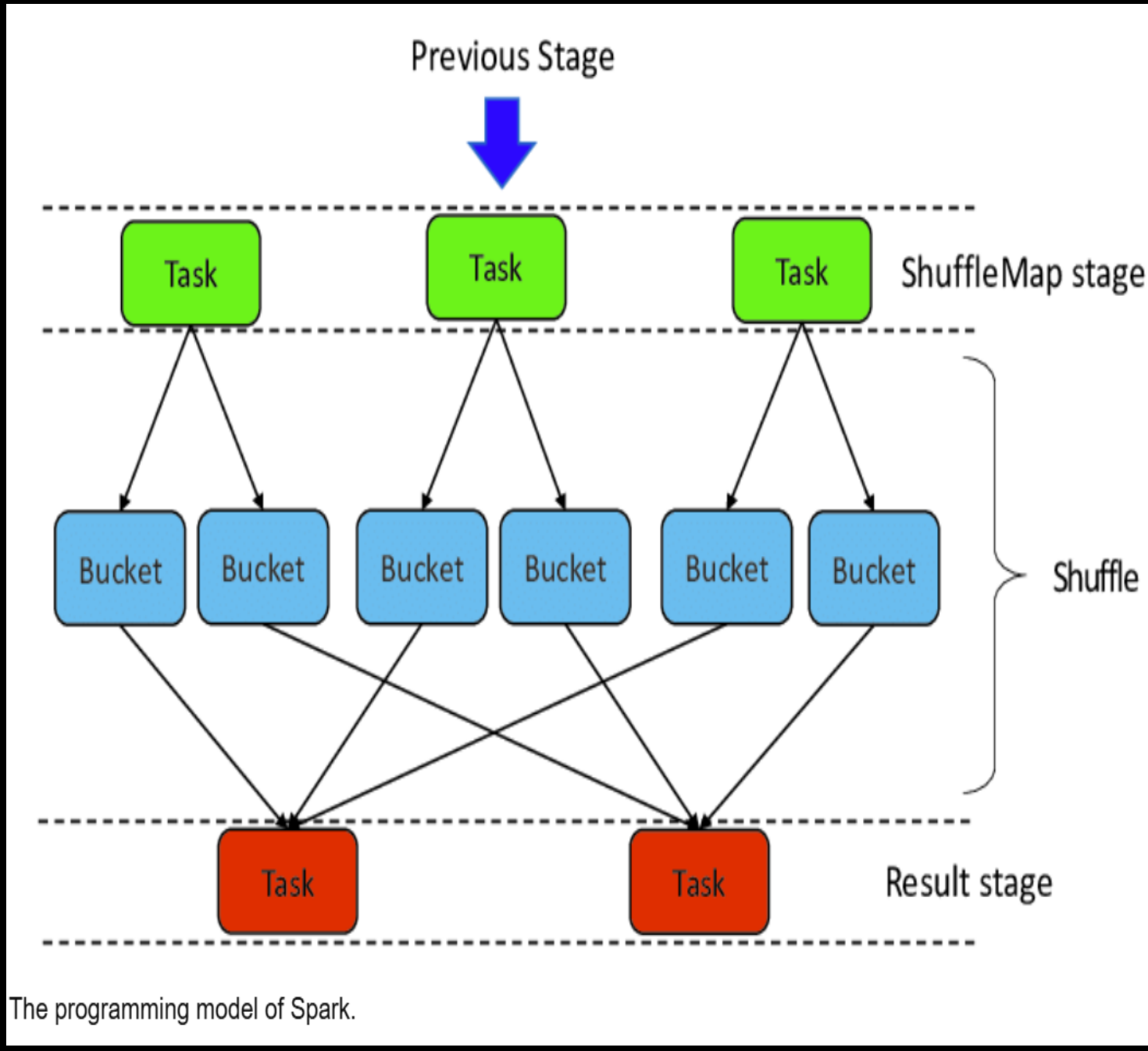
- Scaladoc- Spark Scala Api
- Javadoc - Spark Java Api
- Sphinx - Spark Python Api
- Roxygen2- Spark R api

Spark commands

```
scala> :help
All commands can be abbreviated, e.g., :he instead of :help.
:completions <string>    output completions for the given string
:edit <id>|<line>        edit history
:help [command]          print this summary or command-specific help
:history [num]           show the history (optional num is commands to show)
:h? <string>             search the history
:imports [name name ...] show import history, identifying sources of names
:implicits [-v]          show the implicits in scope
:javap <path|class>      disassemble a file or class name
:line <id>|<line>        place line(s) at the end of history
:load <path>             interpret lines in a file
:paste [-raw] [path]     enter paste mode or paste a file
:power                   enable power user mode
:quit                    exit the interpreter
:replay [options]        reset the repl and replay all previous commands
:require <path>          add a jar to the classpath
:reset [options]         reset the repl to its initial state, forgetting all session entries
:save <path>             save replayable session to a file
:sh <command line>       run a shell command (result is implicitly => List[String])
:setting <options>       update compiler options, if possible; see reset
:silent                  disable/enable automatic printing of results
:type [-v] <expr>        display the type of an expression without evaluating it
:kind [-v] <type>        display the kind of a type. see also :help kind
:warnings                show the suppressed warnings from the most recent line which had any
```

SHIV NADAR UNIVERSITY CHENNAI

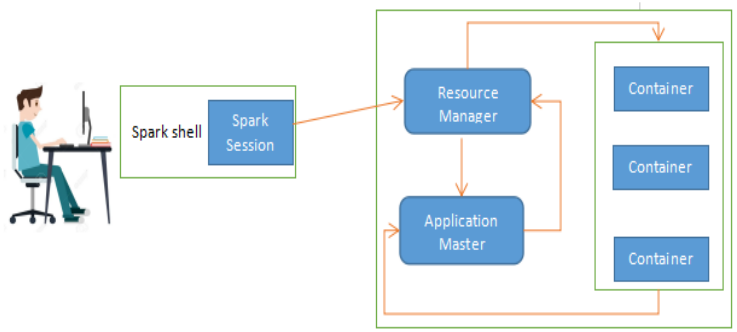
Spark programming model



The programming model of Spark.

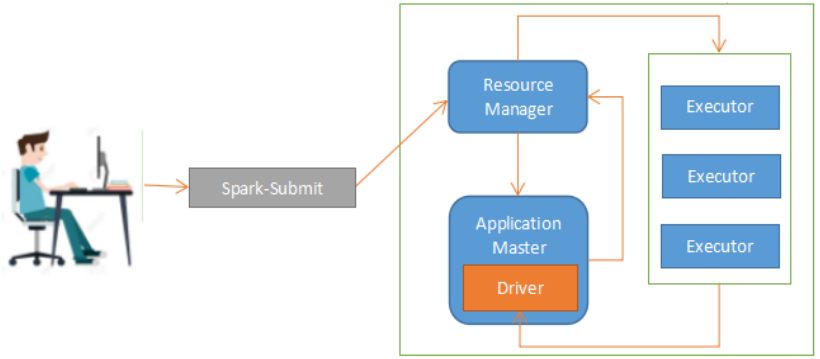
Spark Deployment

CLIENT MODE:



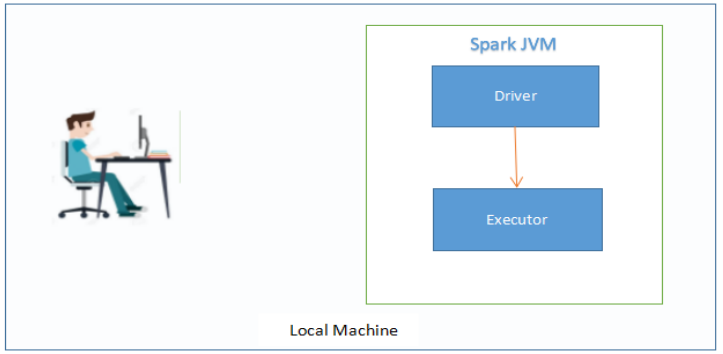
- Spark driver runs within the client machine from which you submit your Spark application.
- The client communicates directly with the cluster manager to request resources and execute tasks.
- Suitable for interactive environments (e.g., Jupyter notebooks) but **not recommended for production** due to dependency on the client machine.

CLUSTER MODE:



- Spark driver runs on one of the cluster nodes rather than on the client machine.
- The client submits the Spark application to the Resource manager, which launches the driver within one of its worker nodes.
- Ideal for **production** as it is more stable and scalable.

LOCAL MODE:



- Spark runs on a single machine, using all the cores of the machine.
- Best for **testing and debugging**, requiring minimal setup.

Cluster mode *is best for production,* ***client mode*** *is useful for debugging, and* ***local mode*** *is ideal for small-scale testing.*

SHIV NADAR UNIVERSITY CHENNAI

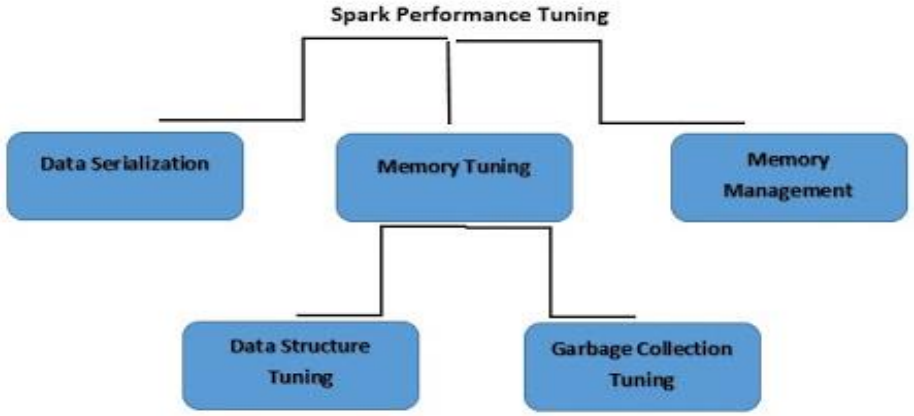
Performance Tuning

1. Data Serialization:
Serialization reduces memory usage and improves network performance in Spark. It enhances system efficiency by optimizing job execution and resource utilization. Spark supports:

- **Java Serialization** (default but slower)
- **Kryo Serialization** (faster and more memory-efficient)

2. Memory Tuning:
To optimize Spark's memory usage:

- Ensure the dataset fits in memory.
- Manage garbage collection efficiently.
- Minimize the cost of object access.



3. Data Structure Tuning:
To reduce memory consumption:

- For RAM < 32GB, enable `-XX:+UseCompressedOops` to reduce pointer size from 8 bytes to 4 bytes, optimizing memory usage.
- Avoid deeply nested structures.
- Prefer numeric IDs over strings.

4. Garbage Collection Tuning:
Efficient garbage collection minimizes overhead by:

- Using arrays instead of linked lists.
- Storing objects in serialized form to reduce memory fragmentation.

5. Memory Management:
Spark divides memory into:

- **Storage Memory:** Caches reusable data.
- **Execution Memory:** Used for computations (shuffles, sorts, joins).
  Challenges include balancing memory between execution, storage, and concurrent tasks.

Future of Spark and Emerging Trends

- **Serverless Computing**
  - Growing adoption of Spark in serverless architectures for better resource efficiency
- **AutoML & AI Integration**
  - Enhanced machine learning automation and integration with AI frameworks
- **Multi-Cloud & Hybrid Deployments**
  - Increased support for multi-cloud and hybrid cloud environments
- **Quantum Computing & Advanced Processing**
  - Potential applications of Spark in next-gen computing paradigms

THANK YOU !

References

- https://aws.amazon.com/what-is/apache-spark/
- https://avinash333.com/spark-architecture/
- https://www.youtube.com/watch?v=Zu9ZEZSfsbU
- https://medium.com/@sephinreji98/understanding-spark-cluster-modes-client-vs-cluster-vs-local-d3c41ea96073
- https://www.databricks.com/glossary/spark-tuning
- https://www.simplilearn.com/future-of-spark-article?utm_source=chatgpt.com#future_of_spark