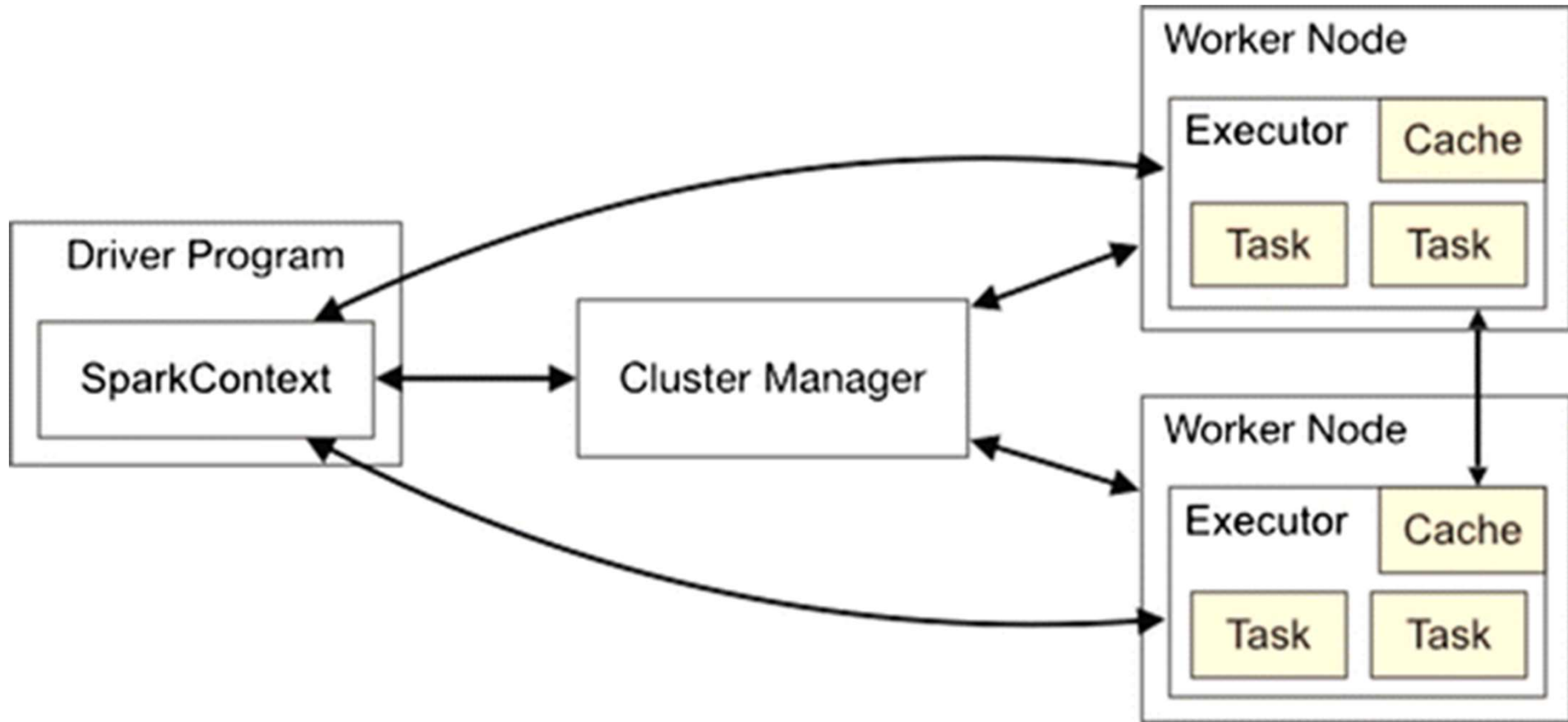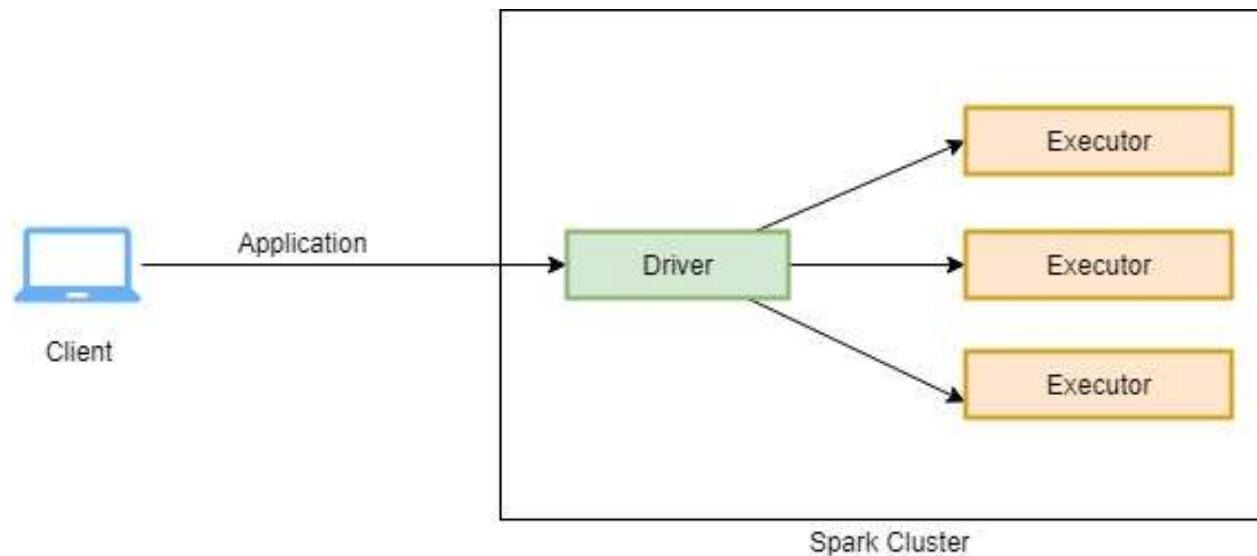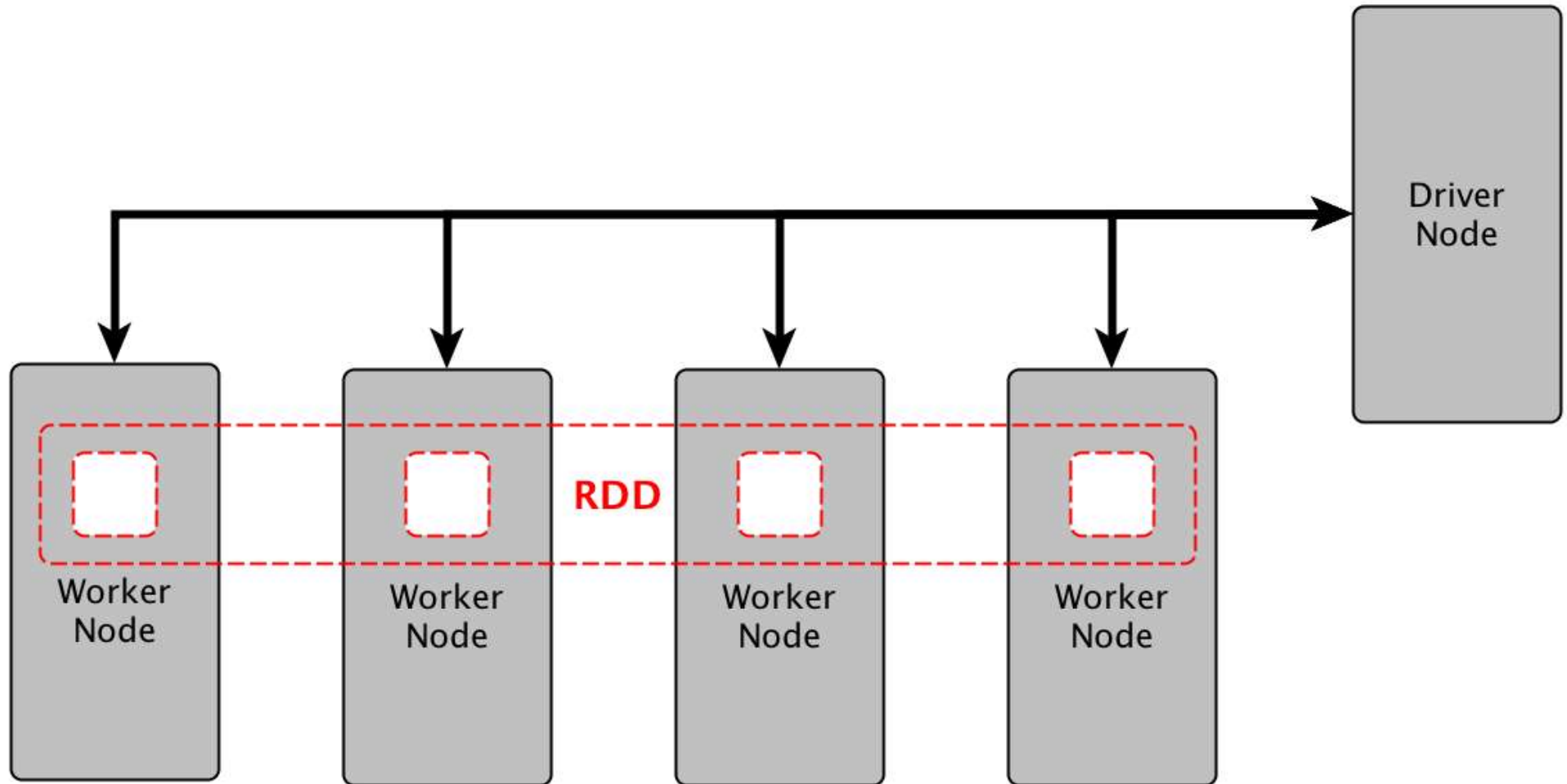# An Introduction to Apache Spark
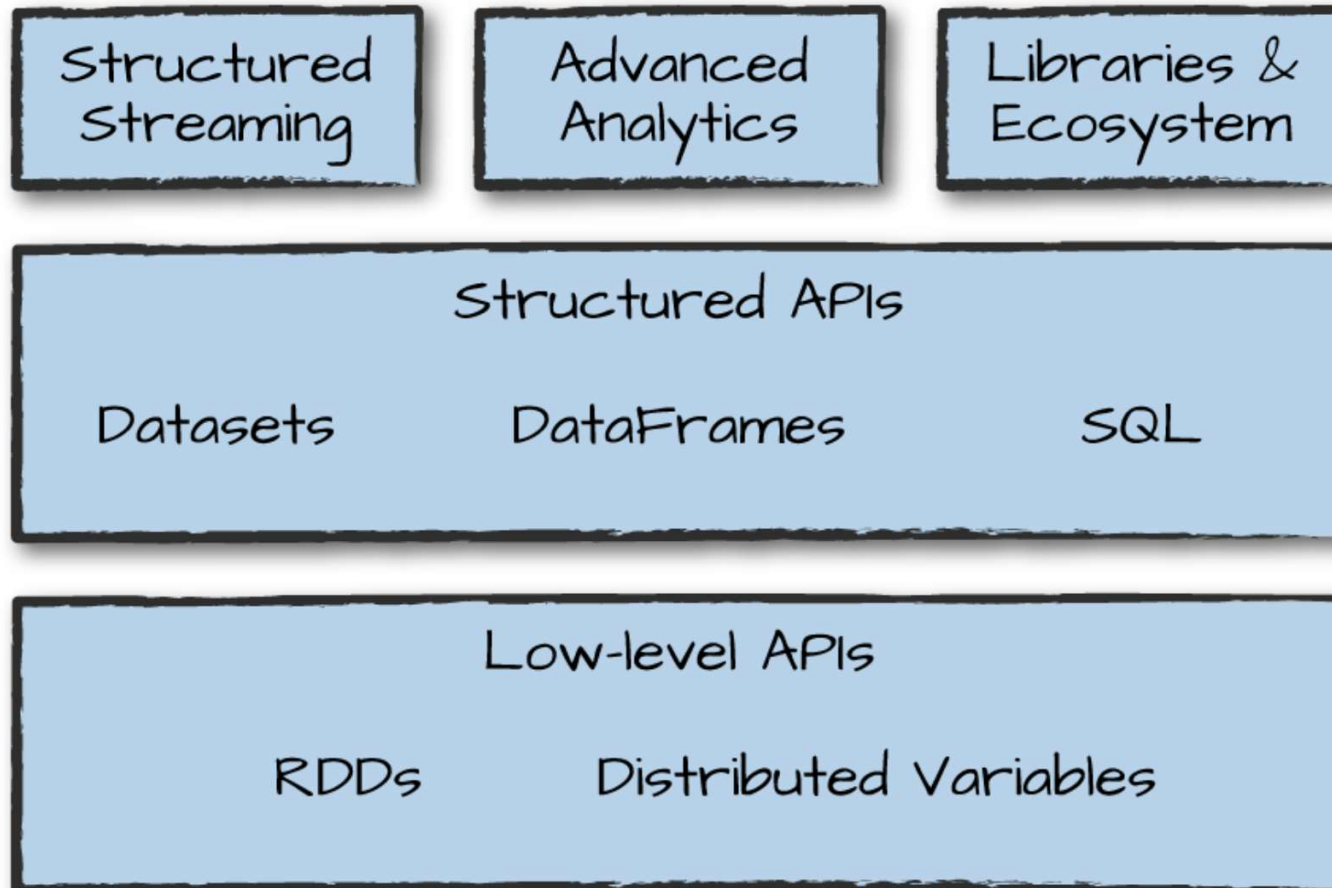
Spark Architecture

Apache Spark Execution

- For every application submitted on spark cluster, spark creates a dedicated Driver process and bunch of Executor processes.

- Driver process is responsible for analyzing, distributing, scheduling and monitoring of executor processes.

- Whereas the executor process is only responsible for running the task they were assigned by drivers and reporting the status back to the driver.

Apache Spark Execution

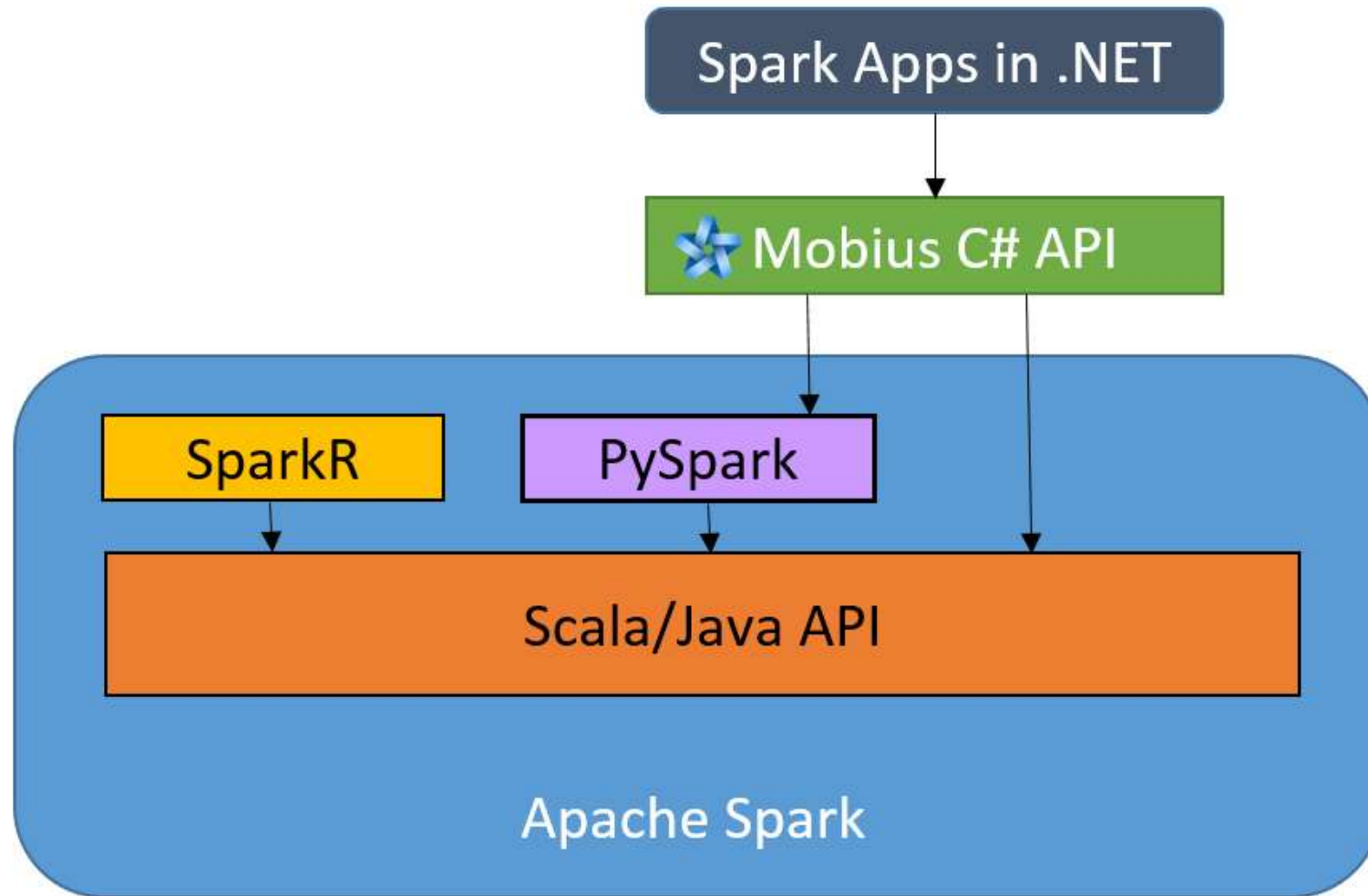# Spark's Language APIs

Structured Streaming	Advanced Analytics	Libraries & Ecosystem

Structured APIs

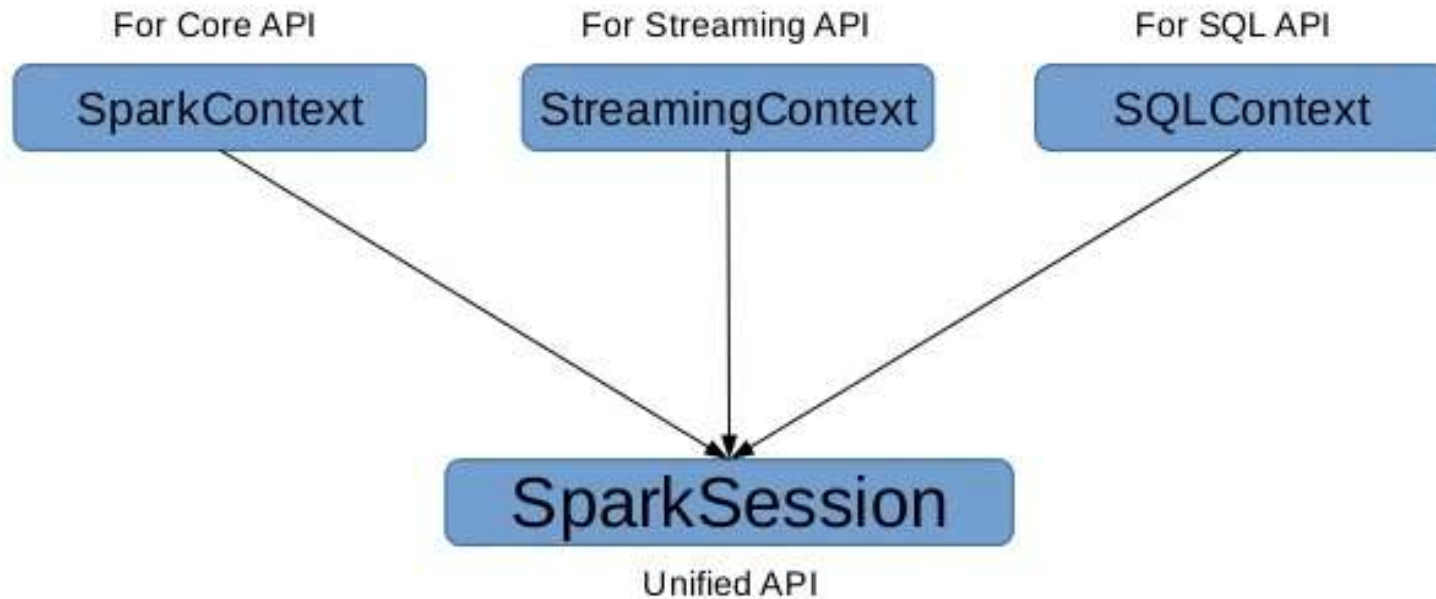Datasets   DataFrames   SQL

Low-level APIs

RDDs   Distributed Variables

Spark's Language APIs

The SparkSession

DataFrames

- In Apache Spark, a DataFrame is a distributed collection of rows
- It has below characteristics:
  - Immutable in nature
    - We can create DataFrame RDD once but can't change it.
  - Lazy Evaluations
    - Which means that a task is not executed until an action is performed.
 - Distributed

Why DataFrames are Useful?

- Designed for processing large collection of data.

- Has the ability to handle petabytes of data.

- Has API support for different languages like
  - Python,
  - R,
  - Scala,
  - Java.

Create a DataFrame

- Can be created using different data formats:
  - JSON
  - CSV
  - XML
  - Excel

- By loading data from Existing RDD

- By Programmatically specifying schema
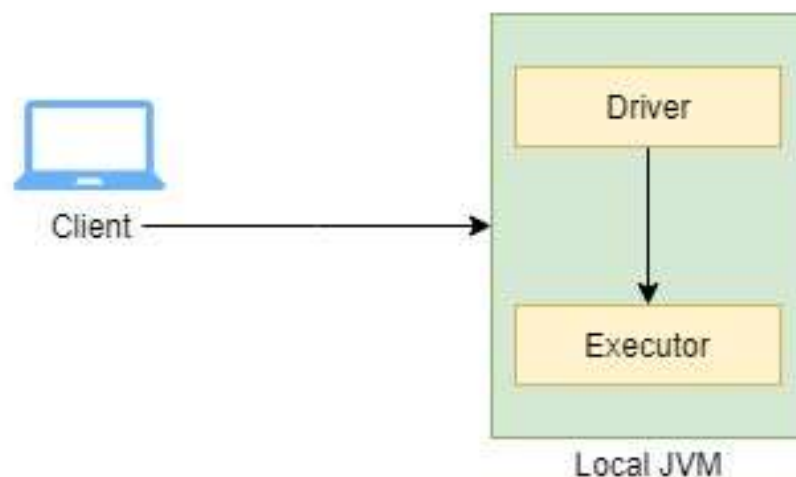
Ways to create DataFrame in Spark
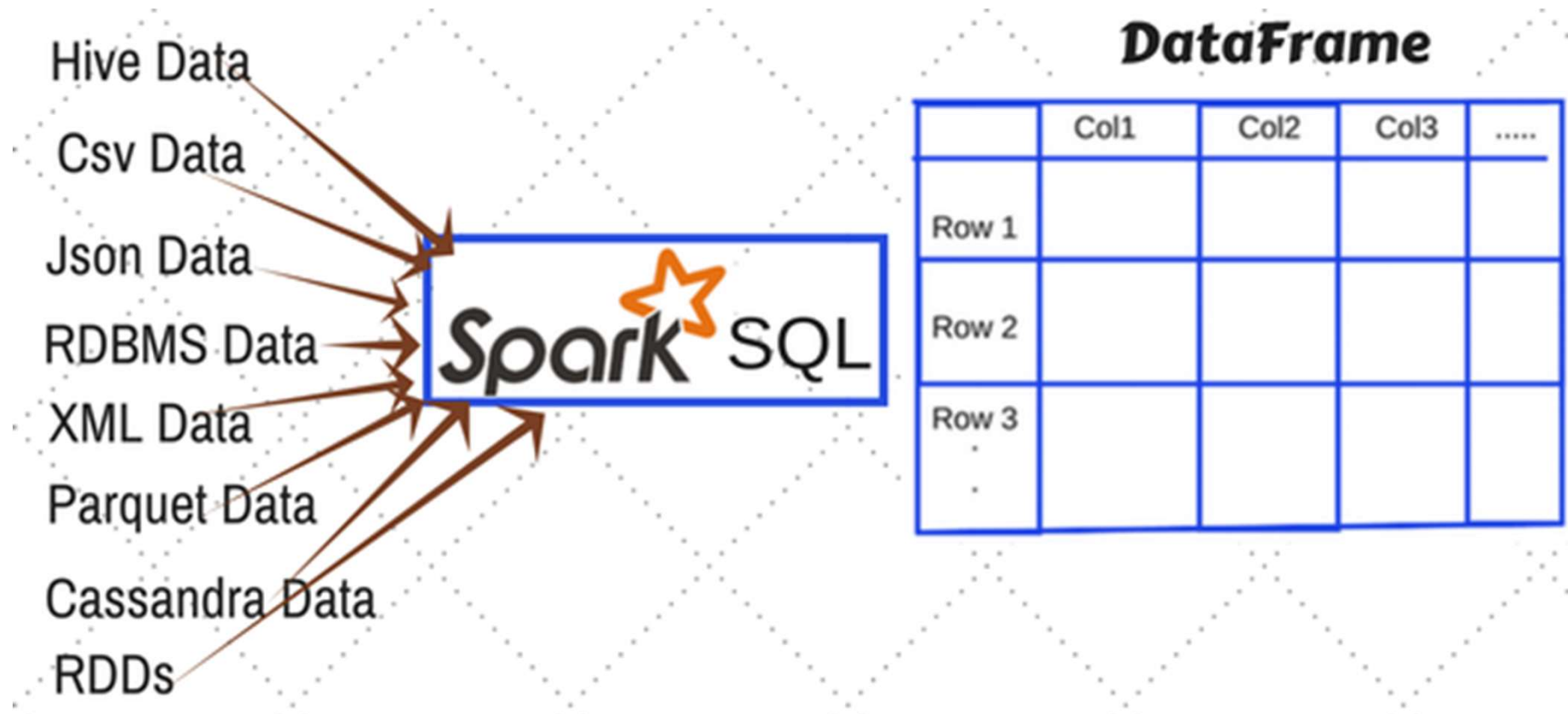
Creating DataFrame from RDD

- from pyspark.sql import Row

- l = [('Ankit',25),('Jalfaizy',22),('saurabh',20),('Bala',26)]

- rdd = sc.parallelize(l)

- people = rdd.map(lambda x: Row(name=x[0], age=int(x[1])))

- schemaPeople = sqlContext.createDataFrame(people)

- type(schemaPeople)

- #Output:
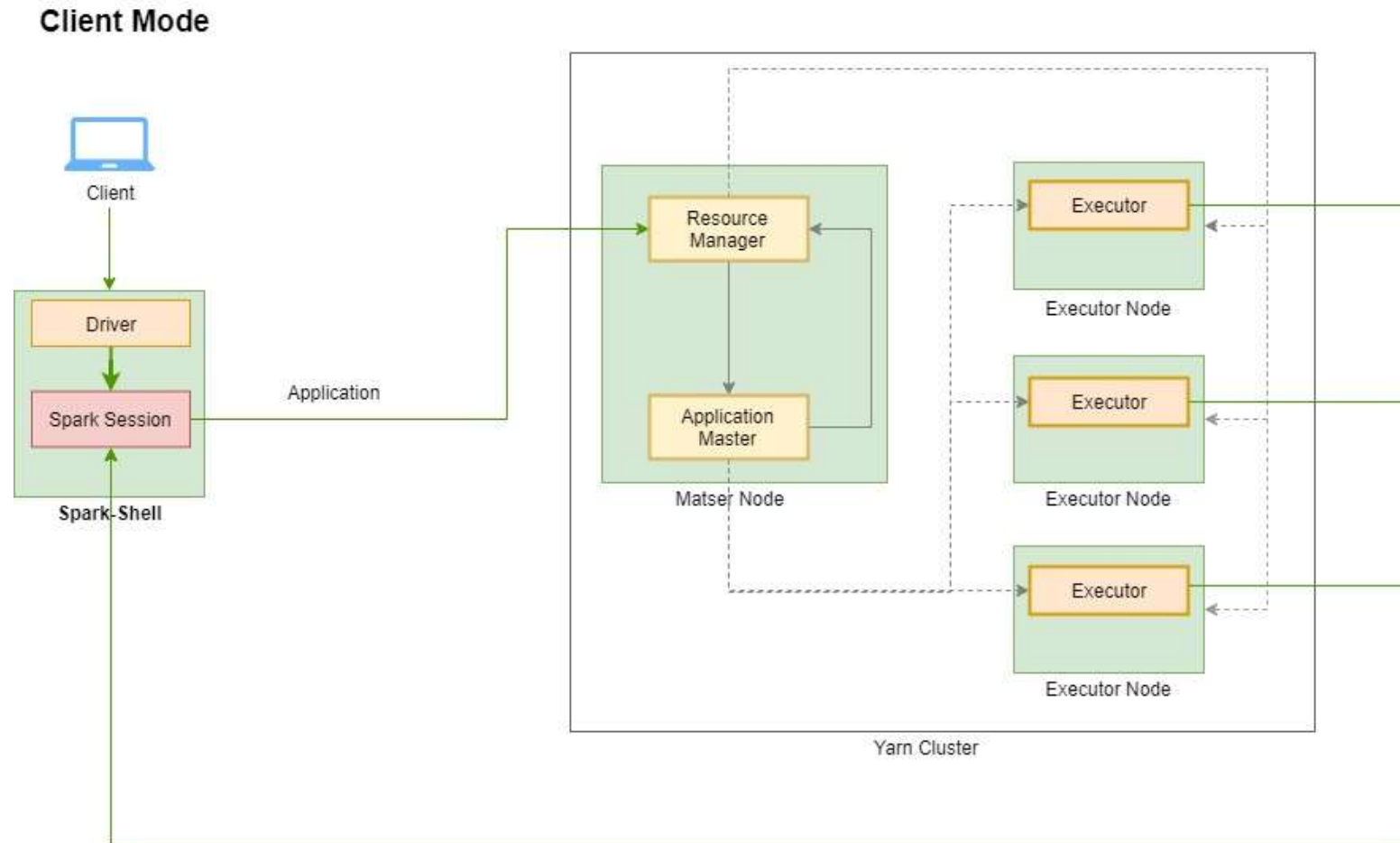
- #pyspark.sql.dataframe.DataFrame
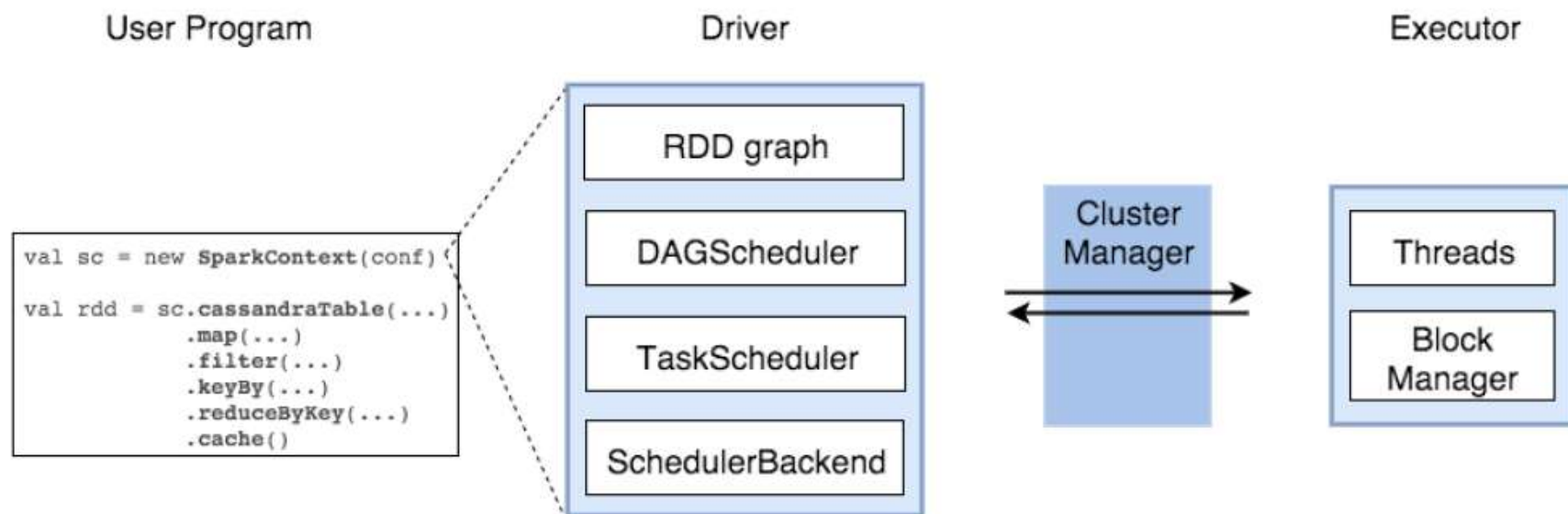
Apache Spark Execution – Local Mode

- There is another mode in which spark can be run locally without any cluster requirement.

- This mode is suitable for scenarios when we do not have enough resources to create cluster.

- But in this mode you get only one executor and both the Driver and Excuter runs in the same JVM.
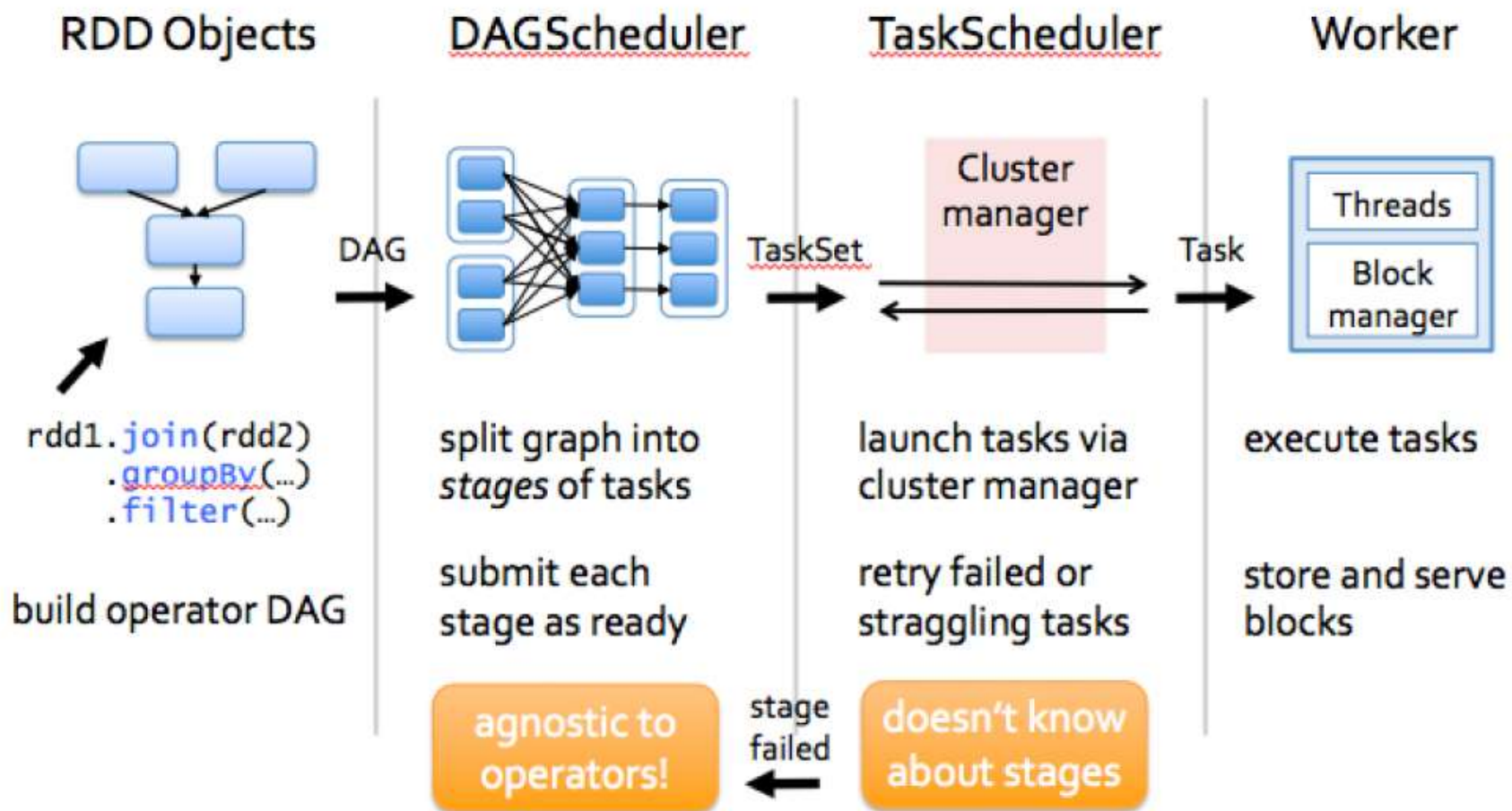
Local Mode

Client →

Driver

Executor

Local JVM

Apache Spark Execution – Client Mode

User Program

Driver

Executor

```
val sc = new SparkContext(conf)

val rdd = sc.cassandraTable(...)
              .map(...)
              .filter(...)
              .keyBy(...)
              .reduceByKey(...)
              .cache()
```

RDD graph

DAGScheduler

TaskScheduler

SchedulerBackend

Cluster Manager

Threads

Block Manager

How Sparks work?

Client

Driver   User Application code   DAG

Task 1
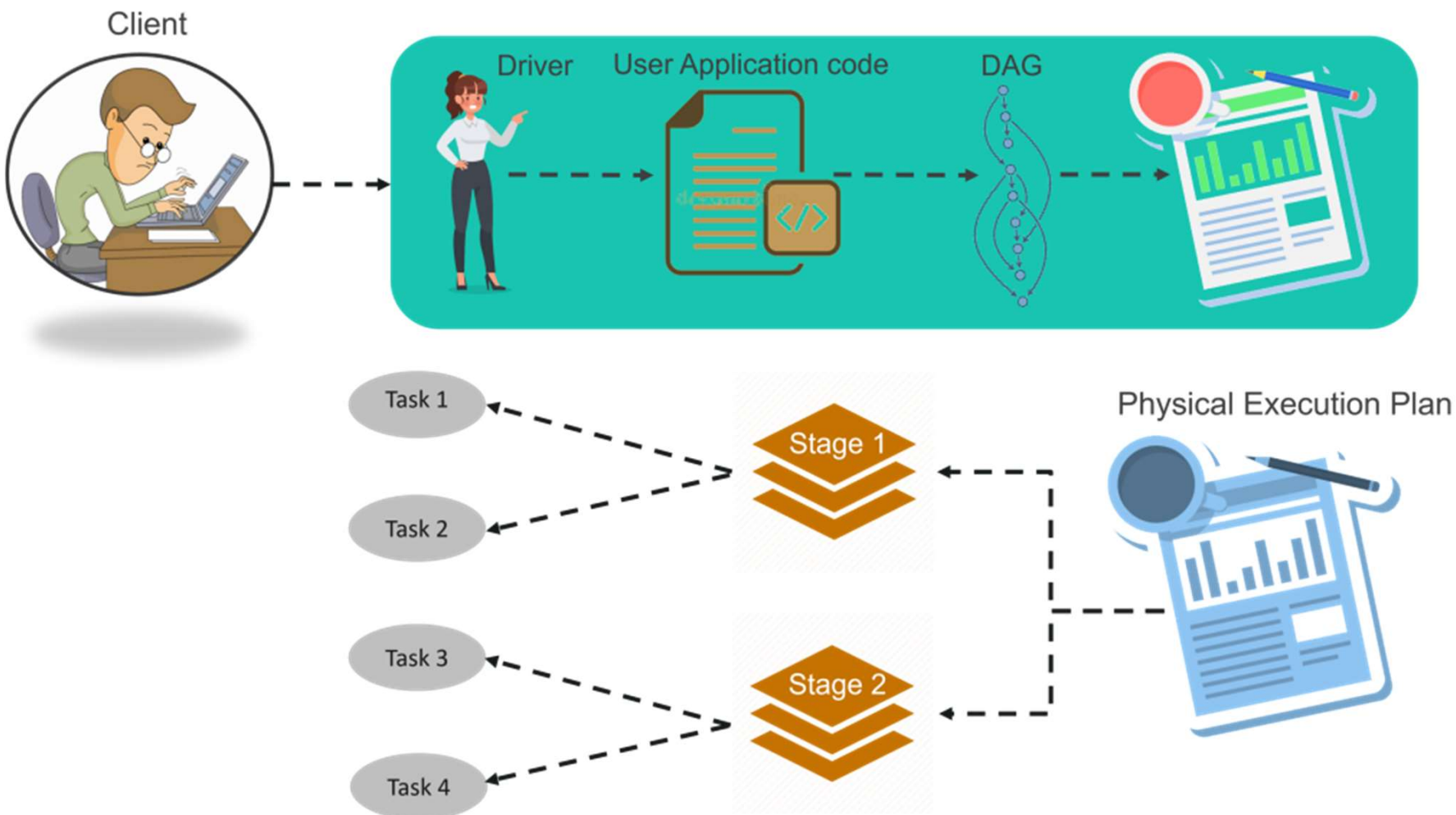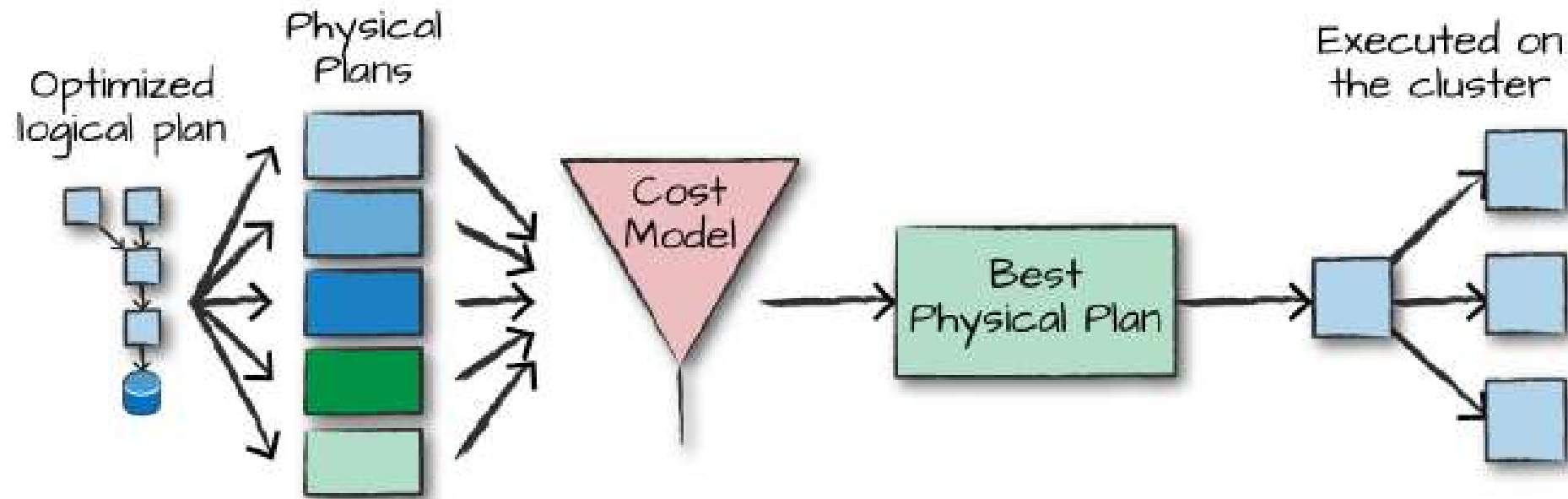
Task 2

Stage 1

Physical Execution Plan

Task 3

Task 4

Stage 2

Catalyst Optimizer

Figure 4-3. The physical planning process

Thanks