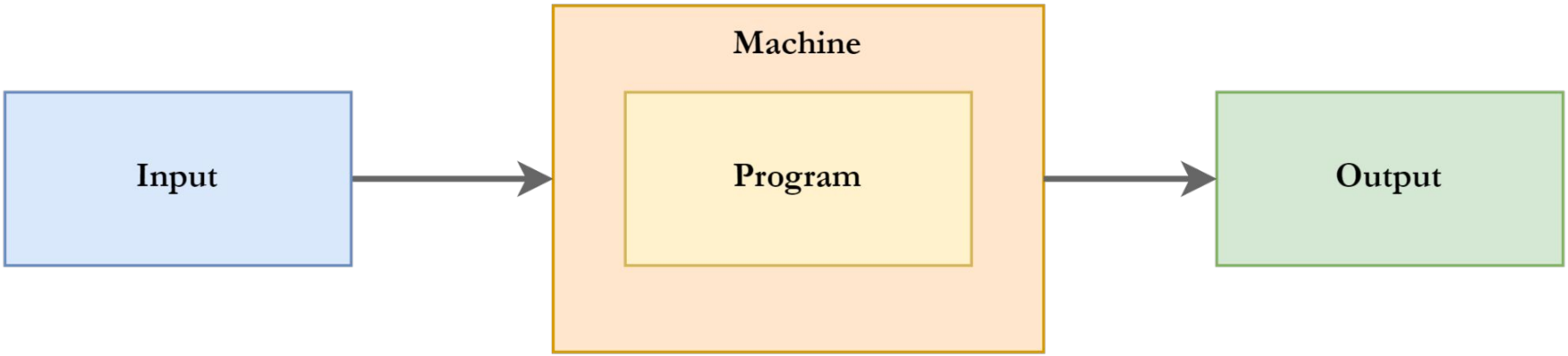
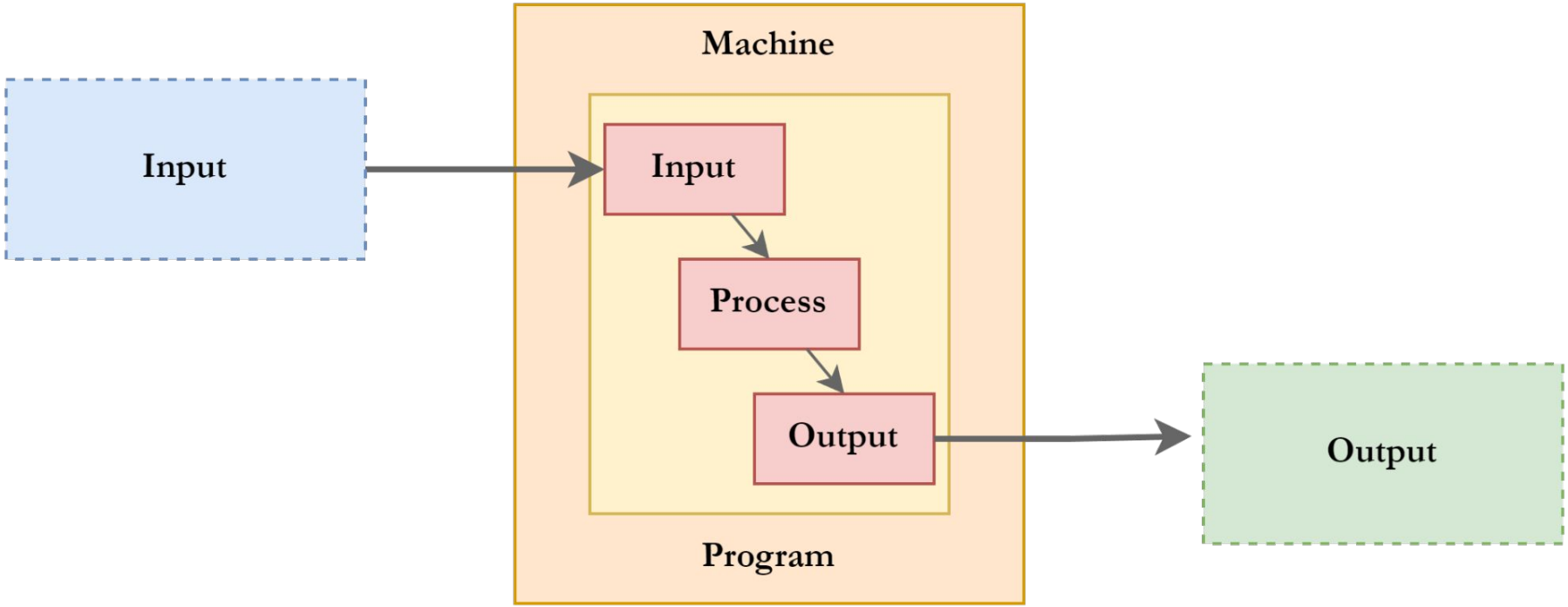
Data Processing with PySpark

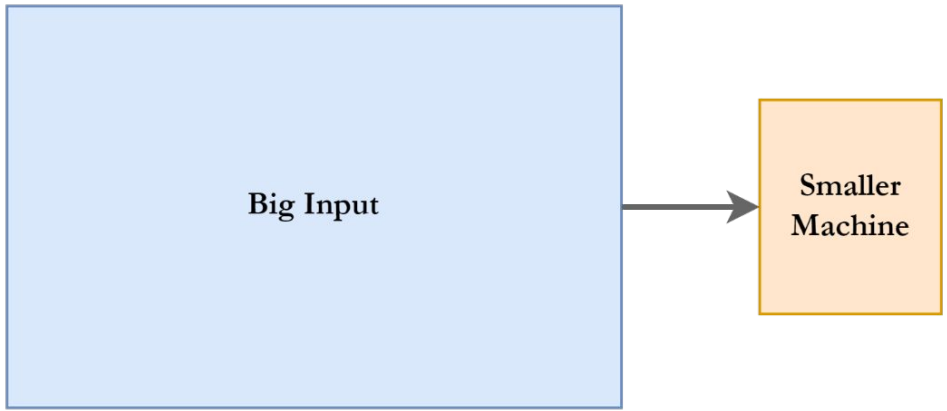
Introduction

Data Processing Model



Enough CPU, Memory and Disk

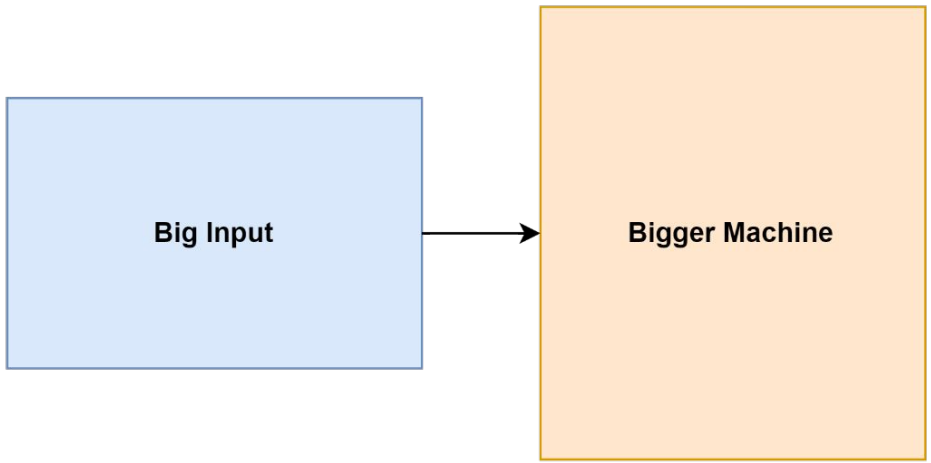




Problem cannot divided to produce partial outputs

Partial outputs must be reduced to the final solution

Vertical Scaling



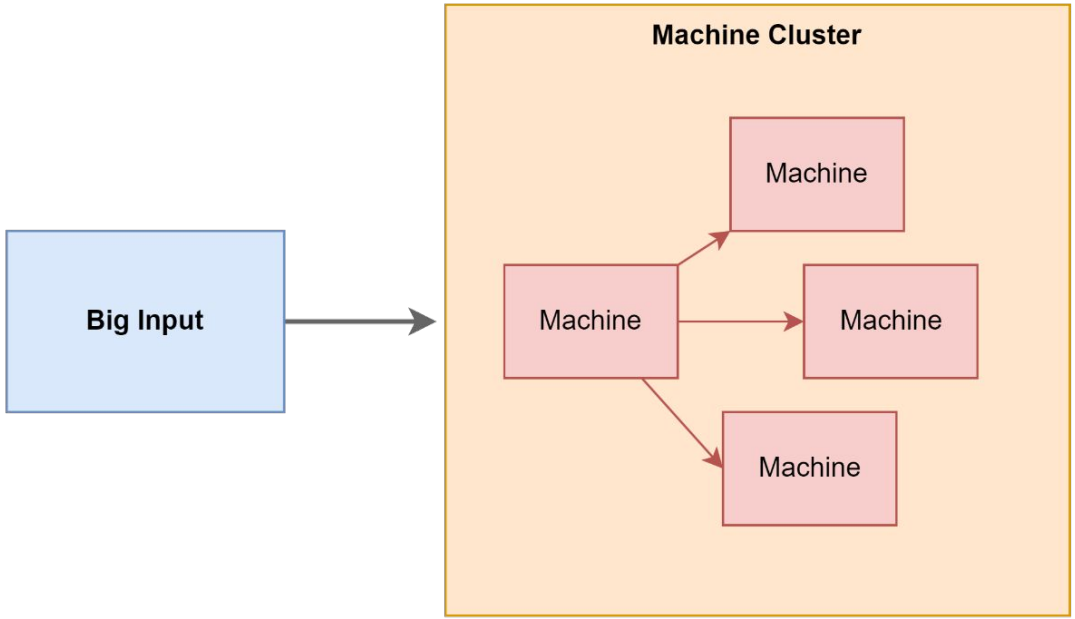
Should be the best solution if possible

No latency of network in processing

Finally the limitation of machine size will be reach
If the input size still in growing

In the end, cost of machine does not grow in linear relationship

Horizontal Scaling



It is possible to buy an affordable machine size in market

There is a challenge in how to make a program running in each machine to collaborate for processing together

Network cost in communition between machines, but it is worth to lessen the processing time

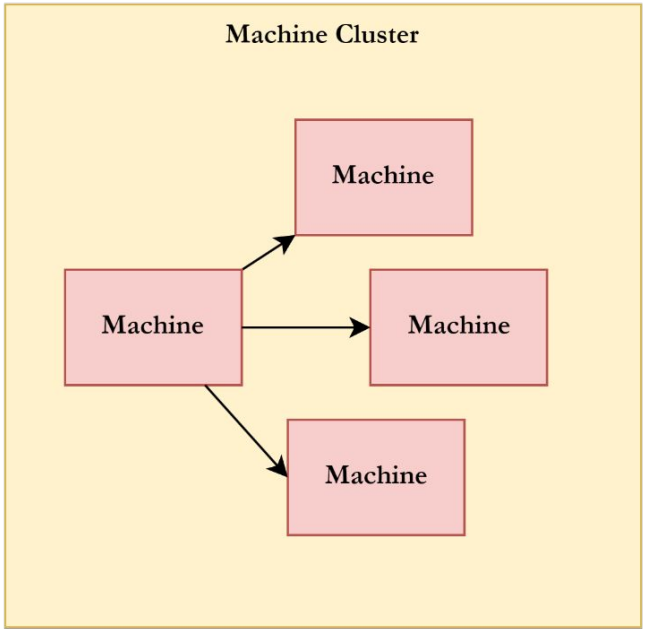
Today, available cloud solution (public or private) make this solution easier to apply

Condition to choose processing solutions

Dataset type	Size range	Fits in RAM?	Fits on local disk?
Small dataset	Less than 2–4 GB	Yes	Yes
Medium dataset	Less than 2 TB	No	Yes
Large dataset	Greater than 2 TB	No	No

Hardest Parts and Challenges

Even if nowaday we have many programming languages which suport network programming..
There are remaining hardest problem need to design and resolve.



Network Fallacies

Resiliency

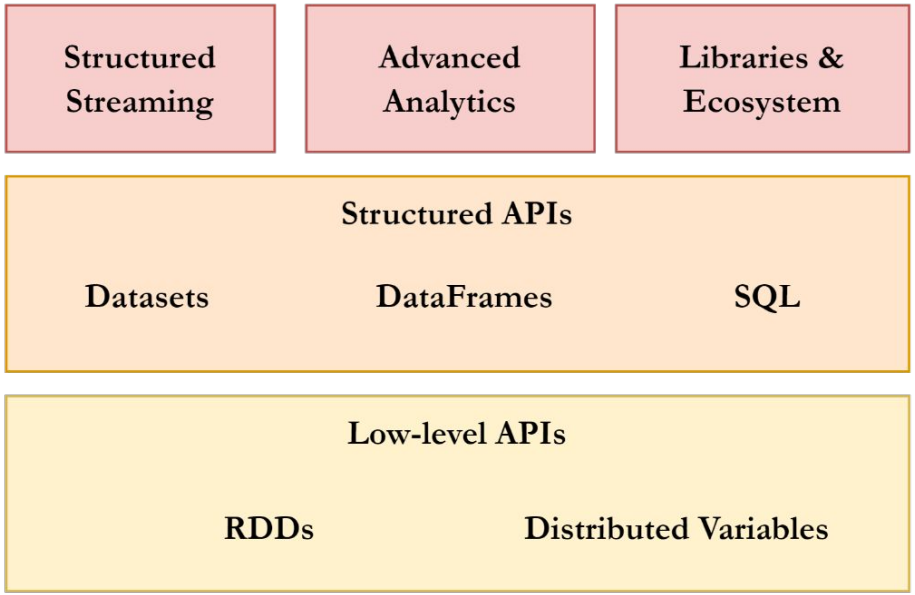
Efficient

Maintainable

Require experience and skills to achieve them

It is not a good decision to make a progam from scratch

Apache Spark is a unified computing engine and a set of libraries for parallel data processing on computer clusters.



- Unified
- Compute Engine
- Libraries

Unified

Supports data processing from simple to complex tasks

Supports scalability in data processing tasks:

- Single Core, Single Machine
- Multiple Core, Single Machine
- Multiple Core, Multiple Machine

Support multiple programming languages

- Python
- R
- Scala
- Java
- SQL
- C# (Microsoft)
- Go

Libraries

With core libraries provided, there are additional libraries extending Spark, which are supported by community

- SparkML
- GraphFrame

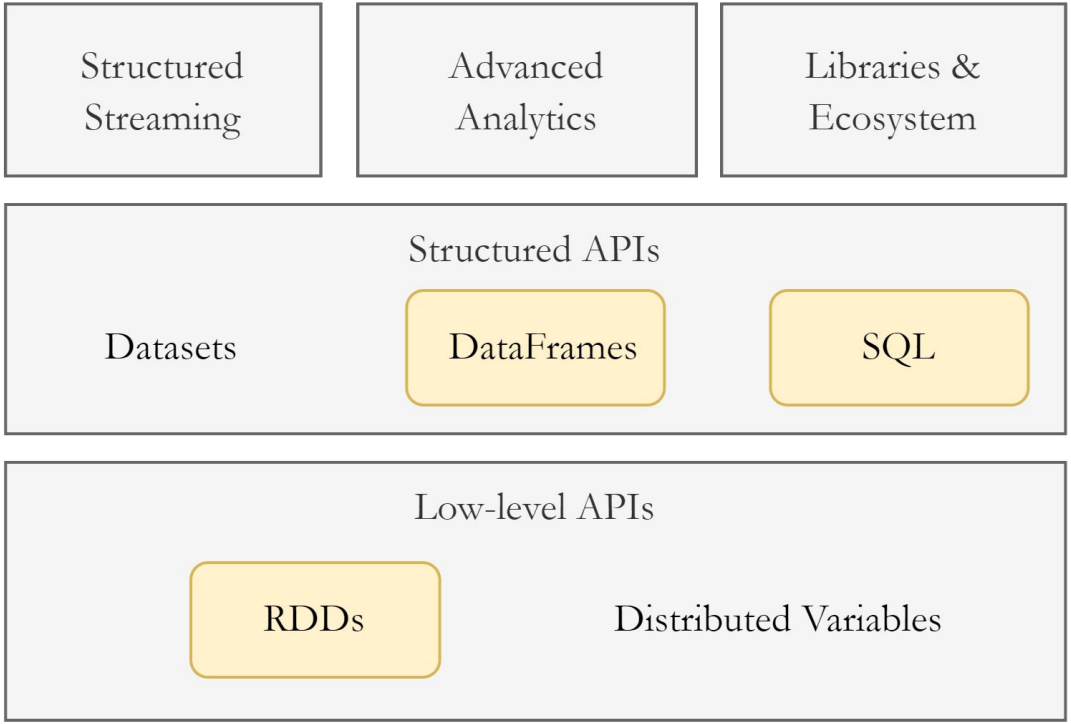
Compute Engine

Focus on computation processing and caching temporary solution for accelerate computation

No concept in long term storage in Spark, but it can integrate with other solutions

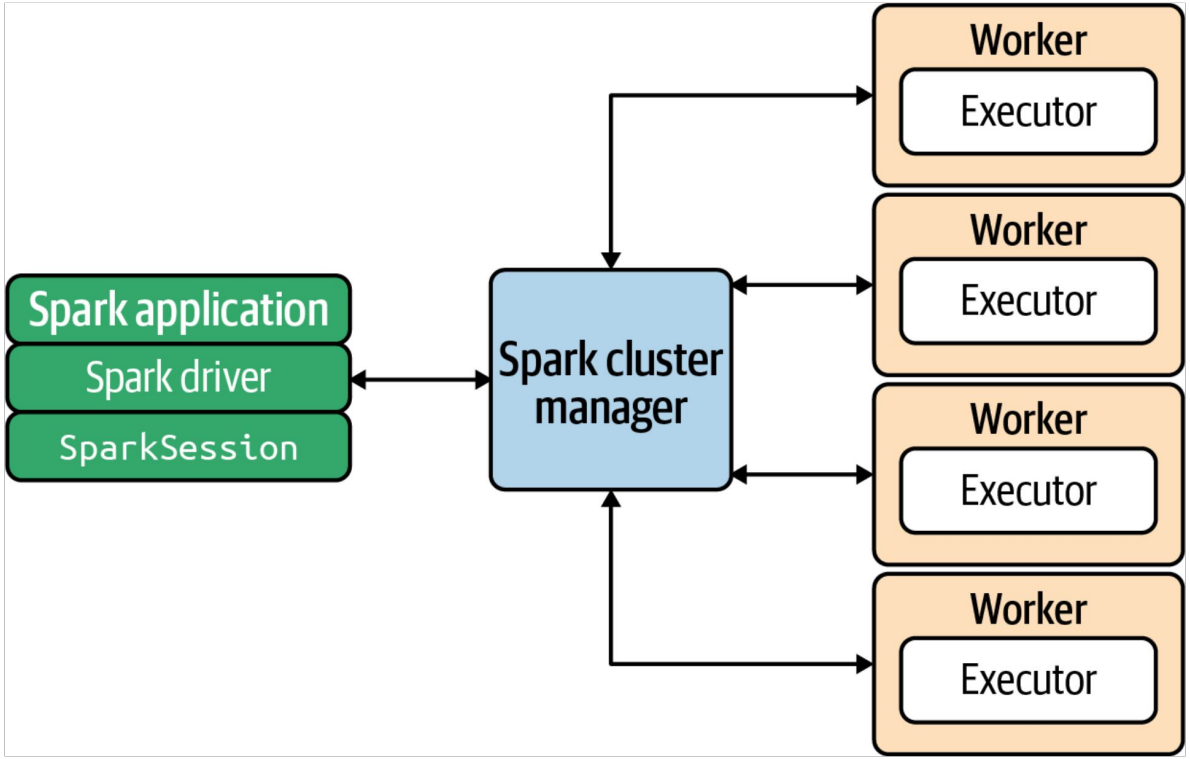
Other storage solutions: S3, Azure Data Lake, RDBMS, NoSQL

What we will focus in this course

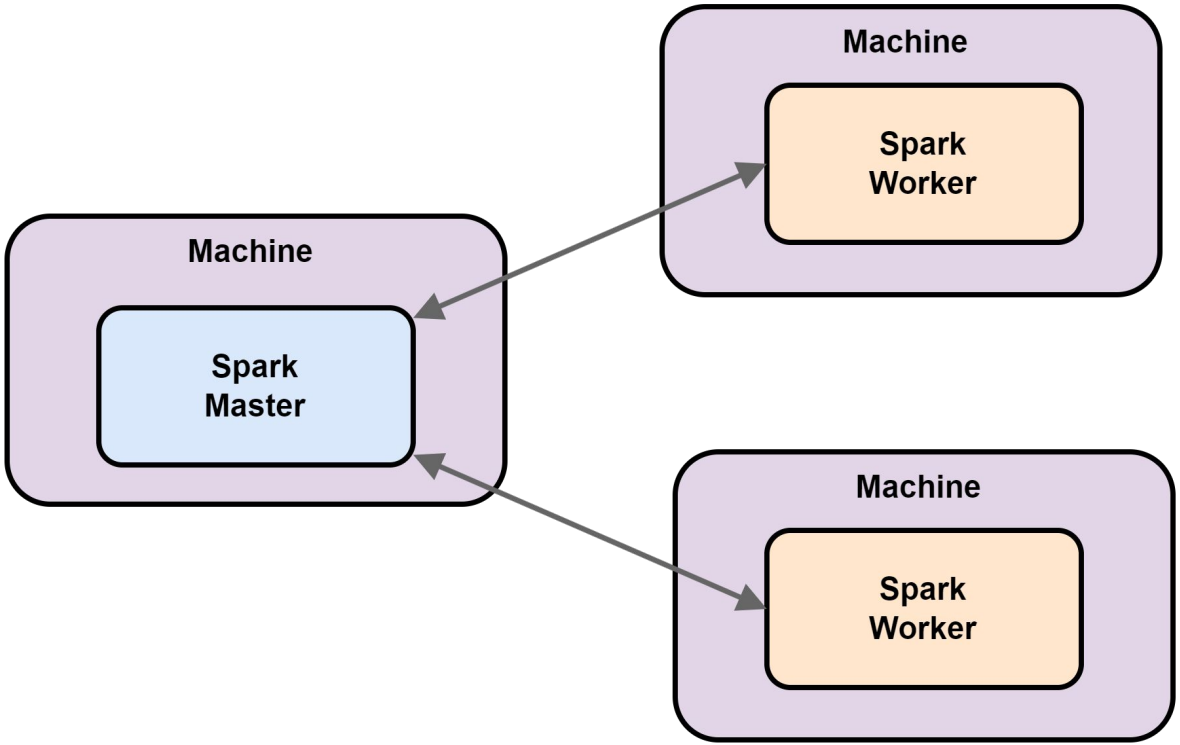


Spark Cluster

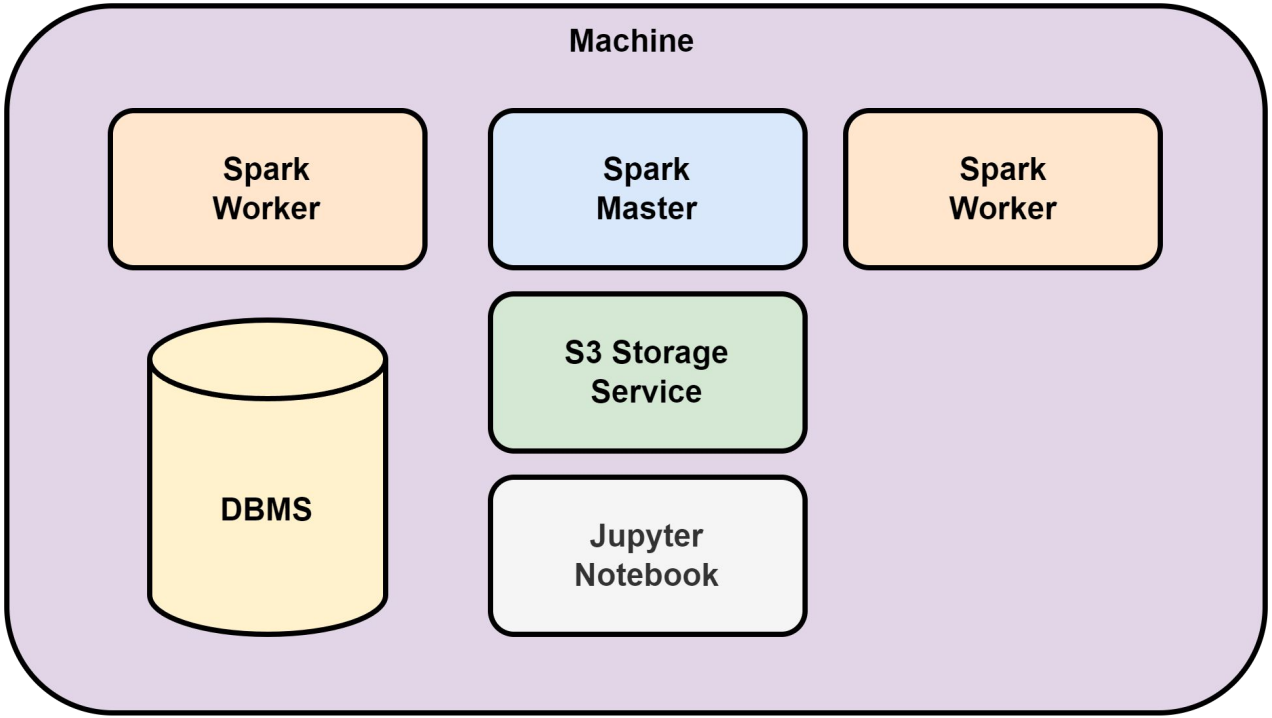
Spark Cluster Architecture



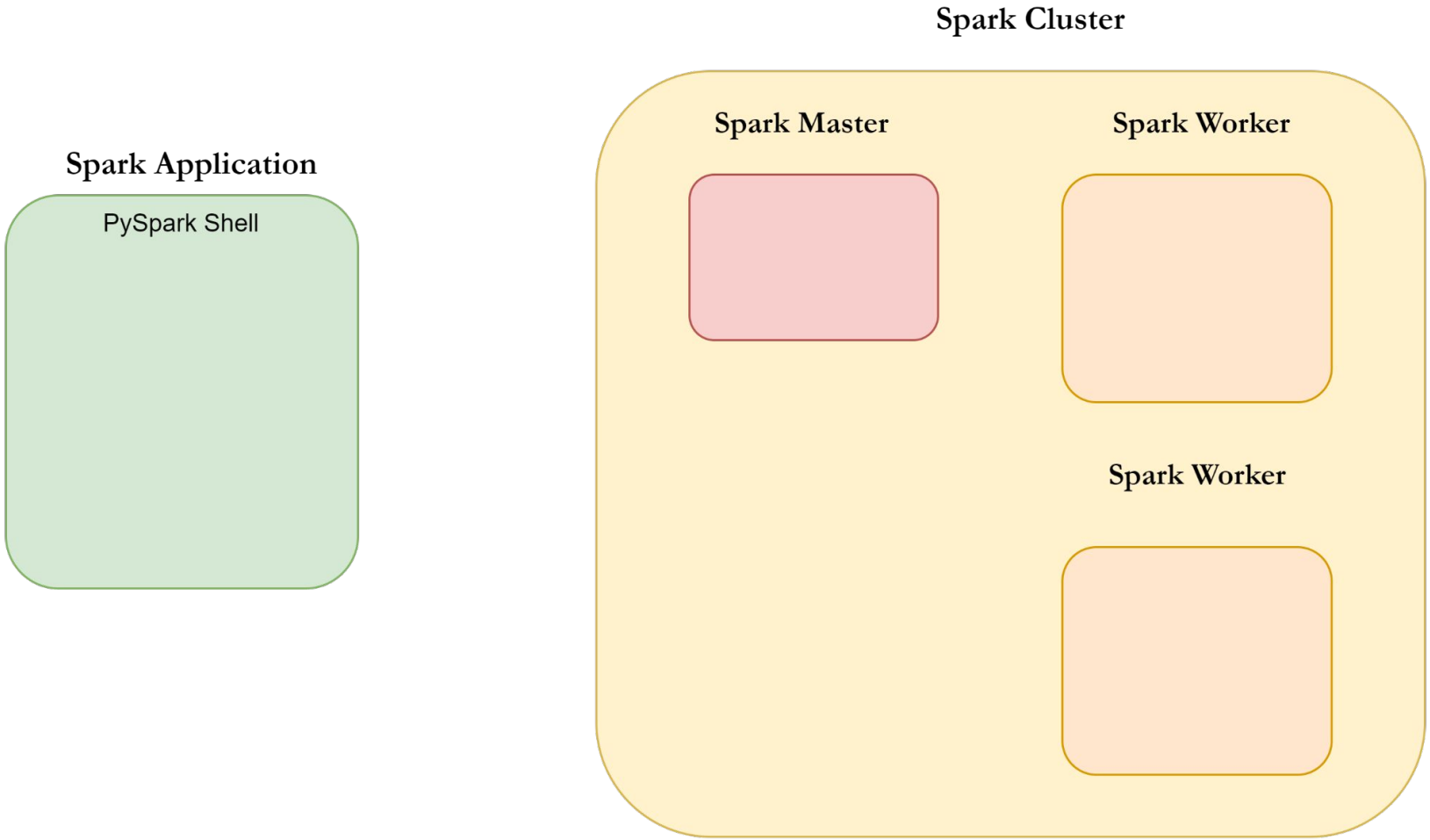
Real Spark Cluster

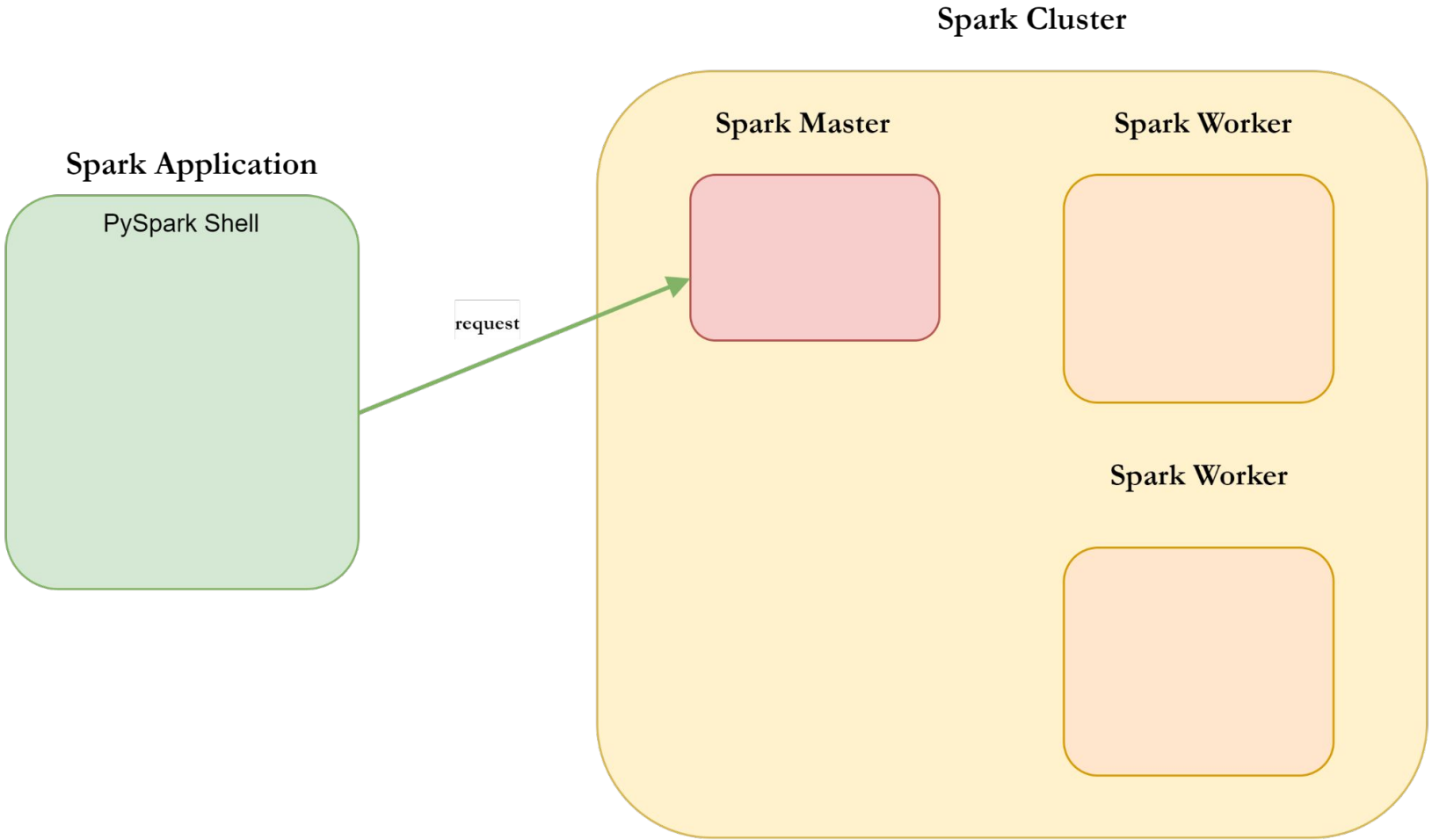


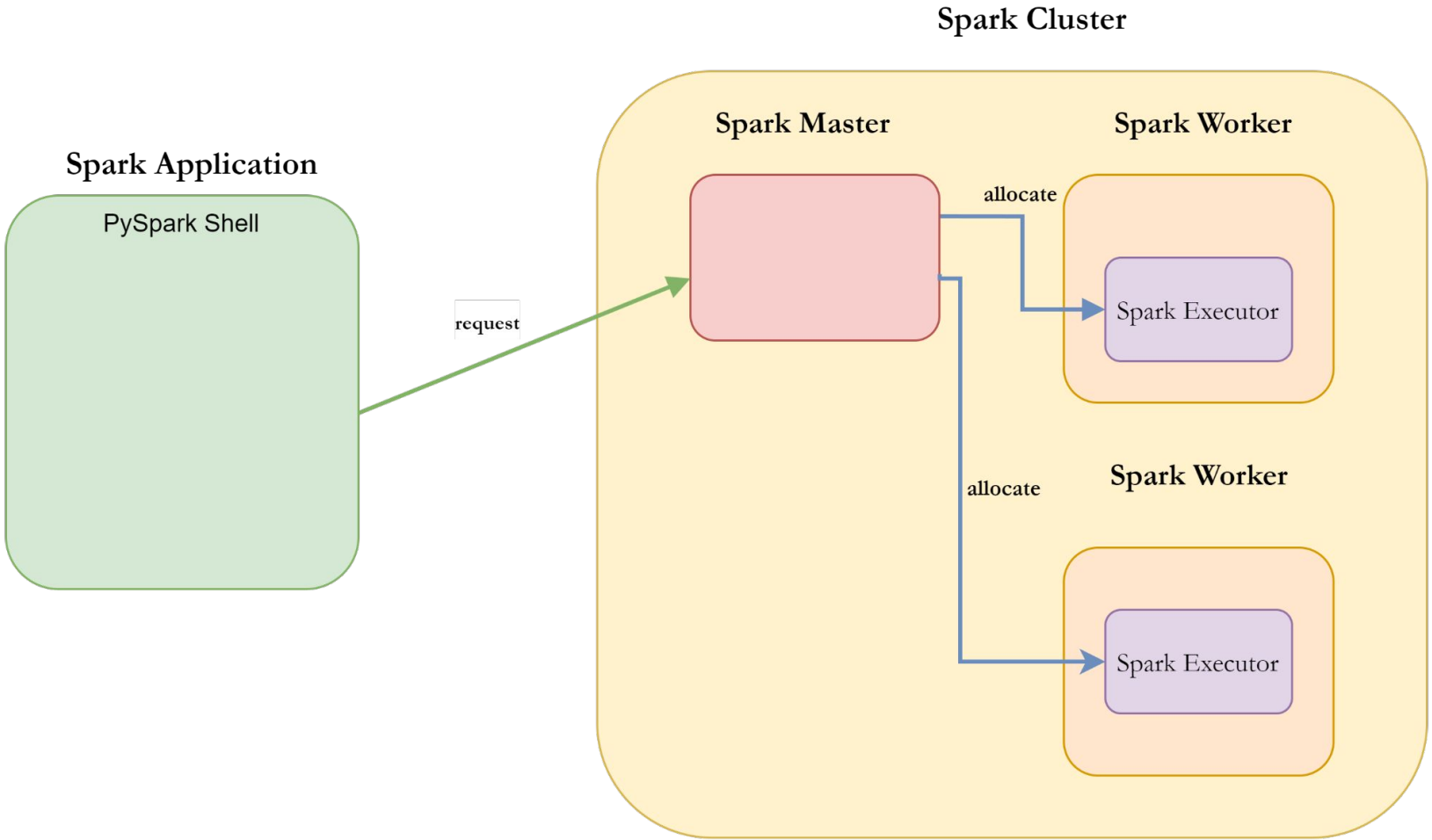
Poorman's Spark Cluster

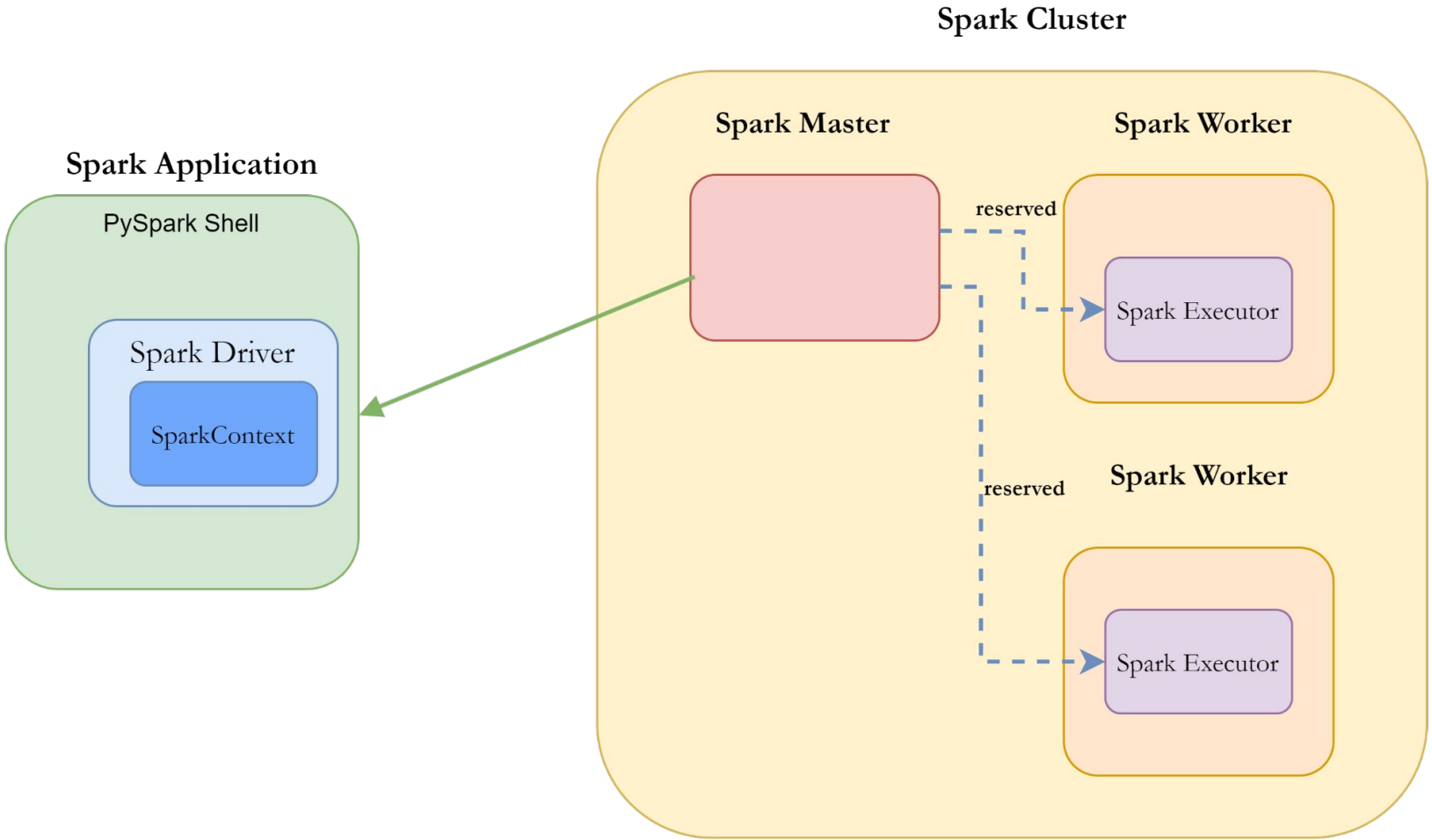


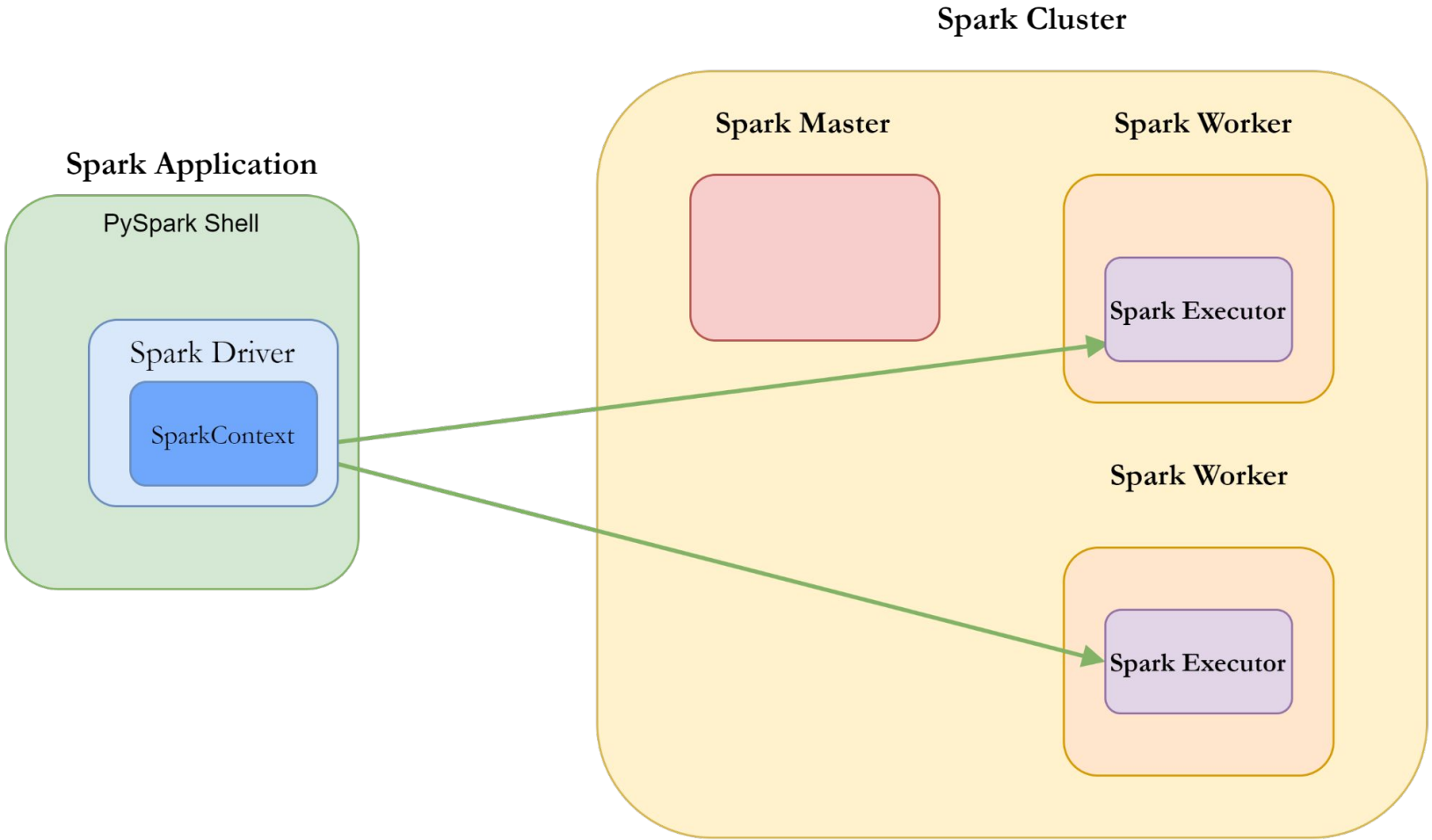
PySparkShell


Spark Application
PySpark Shell
Spark Cluster
Spark Master
Spark Worker
Spark Worker

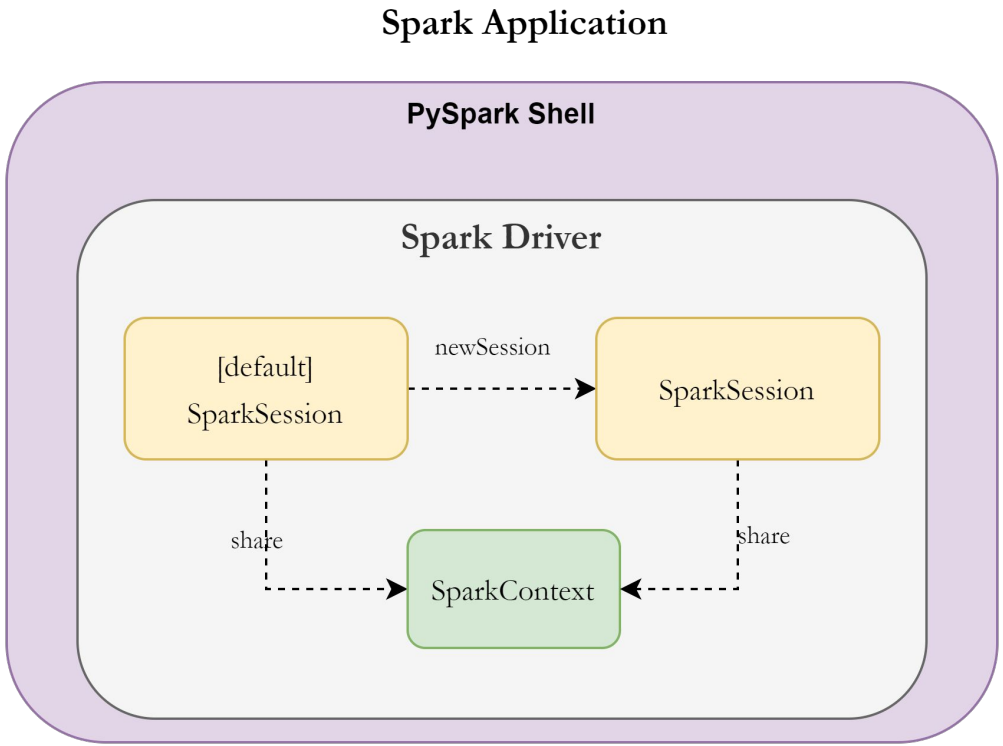

Spark Cluster
Spark Application
PySpark Shell
request
Spark Master
Spark Worker
Spark Worker


Spark Cluster
Spark Application
PySpark Shell
request
Spark Master
Spark Worker
allocate
Spark Executor
allocate
Spark Worker
Spark Executor

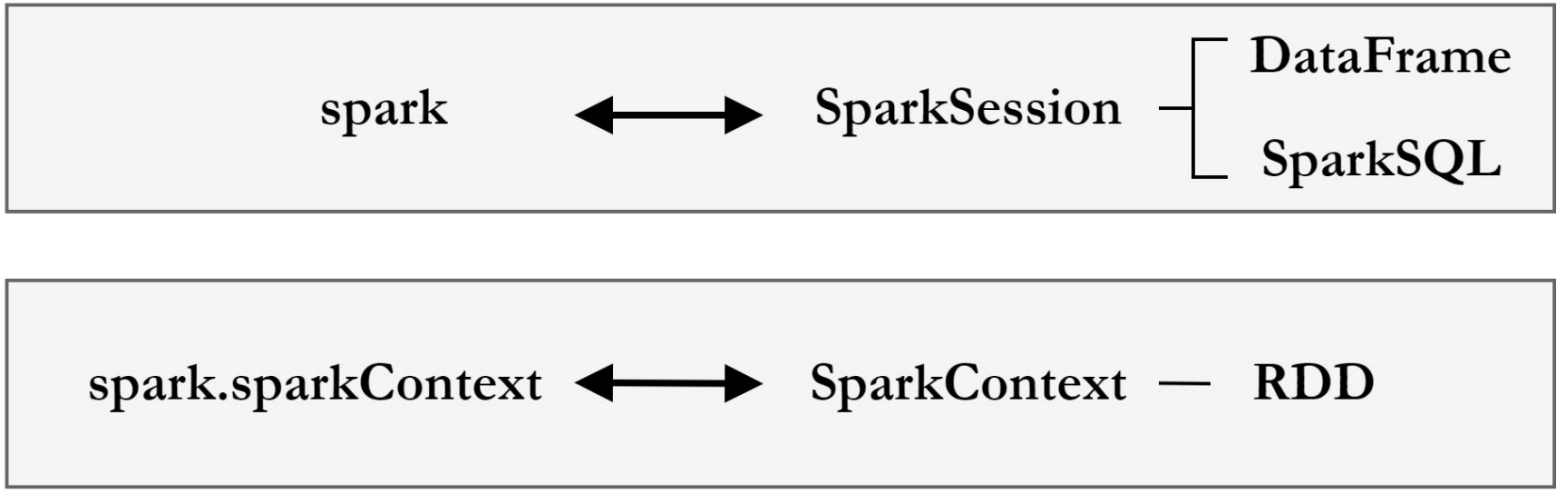

Spark Cluster
Spark Application
PySpark Shell
Spark Driver
SparkContext
Spark Master
Spark Worker
reserved
Spark Executor
reserved
Spark Worker
Spark Executor


Spark Cluster
Spark Application
PySpark Shell
Spark Driver
SparkContext
Spark Master
Spark Worker
Spark Executor
Spark Worker
Spark Executor

What is SparkSession?



Where is SparkSession and SparkContext?



Spark App

How to develop Spark Application

1. Create a SparkSession instance by your program

2. How to run a Spark application

    2.1 Run it directly from a machine which can access all machines in Spark cluster.

    2.2 Use spark-submit script in Spark package (the application cannot override spark.master configuration

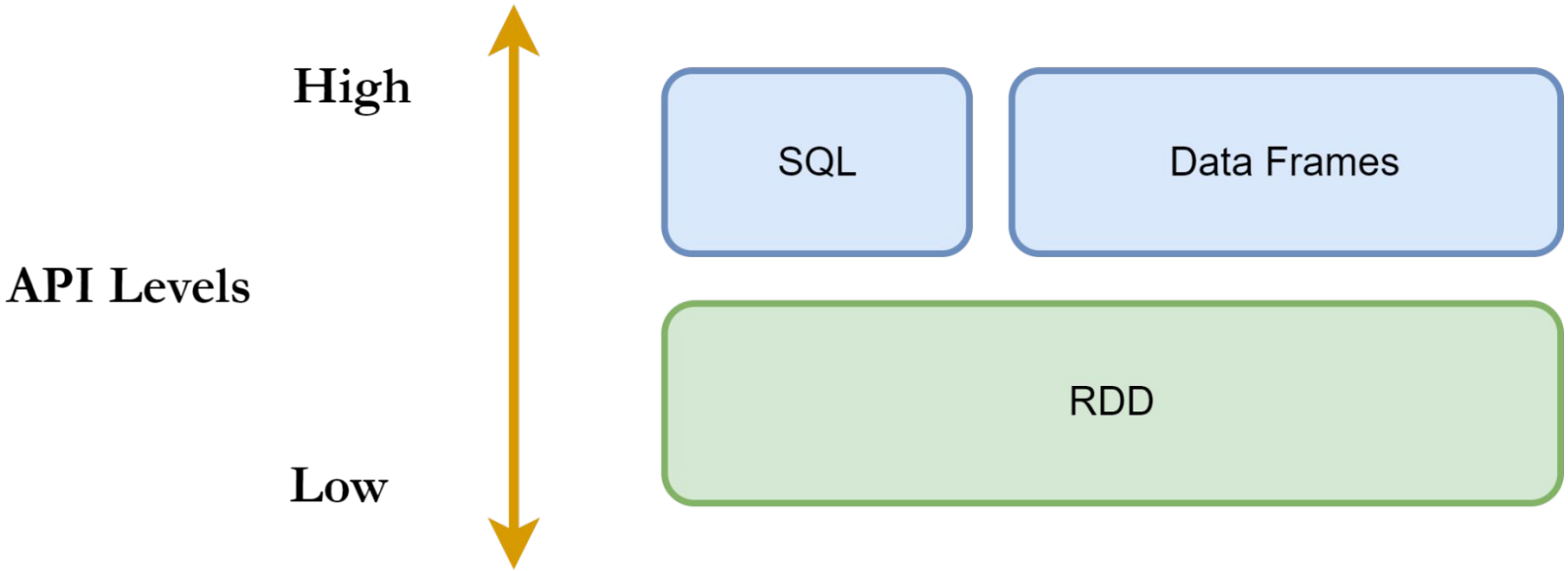
```python
from pyspark.sql import SparkSession


spark = (
  SparkSession
    .builder
    .appName("spark_app_name")
    .master("spark_master_url")
    .config("config_name", "config_value")
    .getOrCreate()
)

spark   # SparkSession instance
spark.sparkContext   # SparkContext instance
```

Spark APIs

PySpark APIs Evoluation



How are data treated in each APIs

RDD

- Processing data like **list of object**
- object type can be primitive types like int, bool, string, or dict, list
- Data are **distributed** and **partitioned** in cluster

DataFrame

- Processing data like **tables and rows with schema** (e. g. Excel or R DataFrame),
- Data are **distributed** and **partitioned** in cluster (as RDD is an underlying layer)

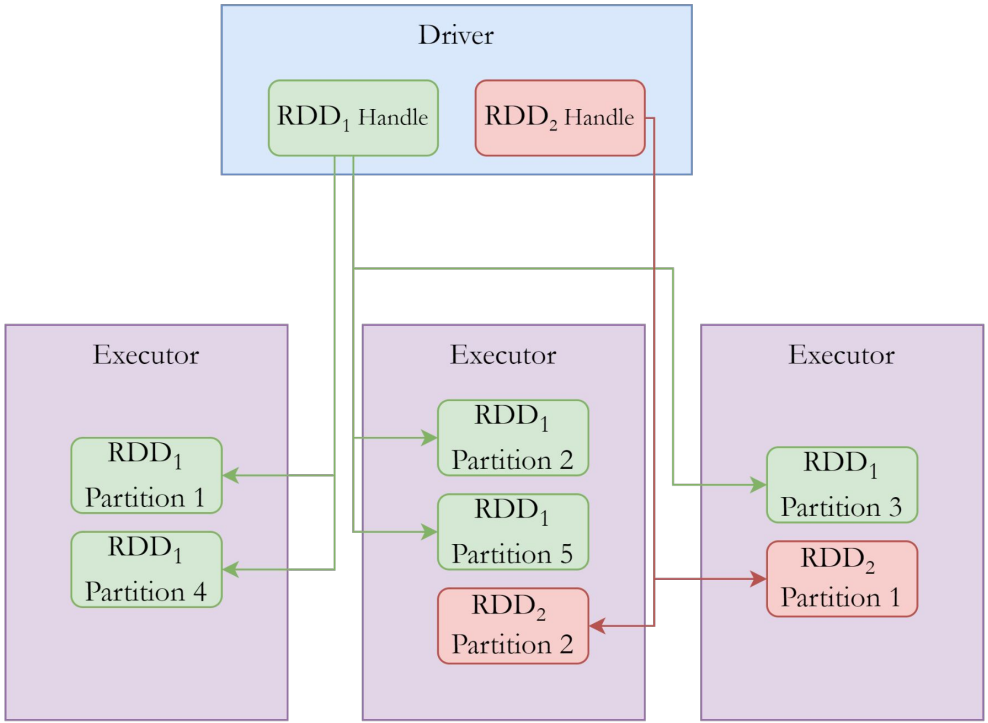
SparkSQL

- Processing data like **tables and rows with schema** (e. g. Excel or R DataFrame)
- Data are **distributed** and **partitioned** in cluster (as RDD is an underlying layer)

RDD

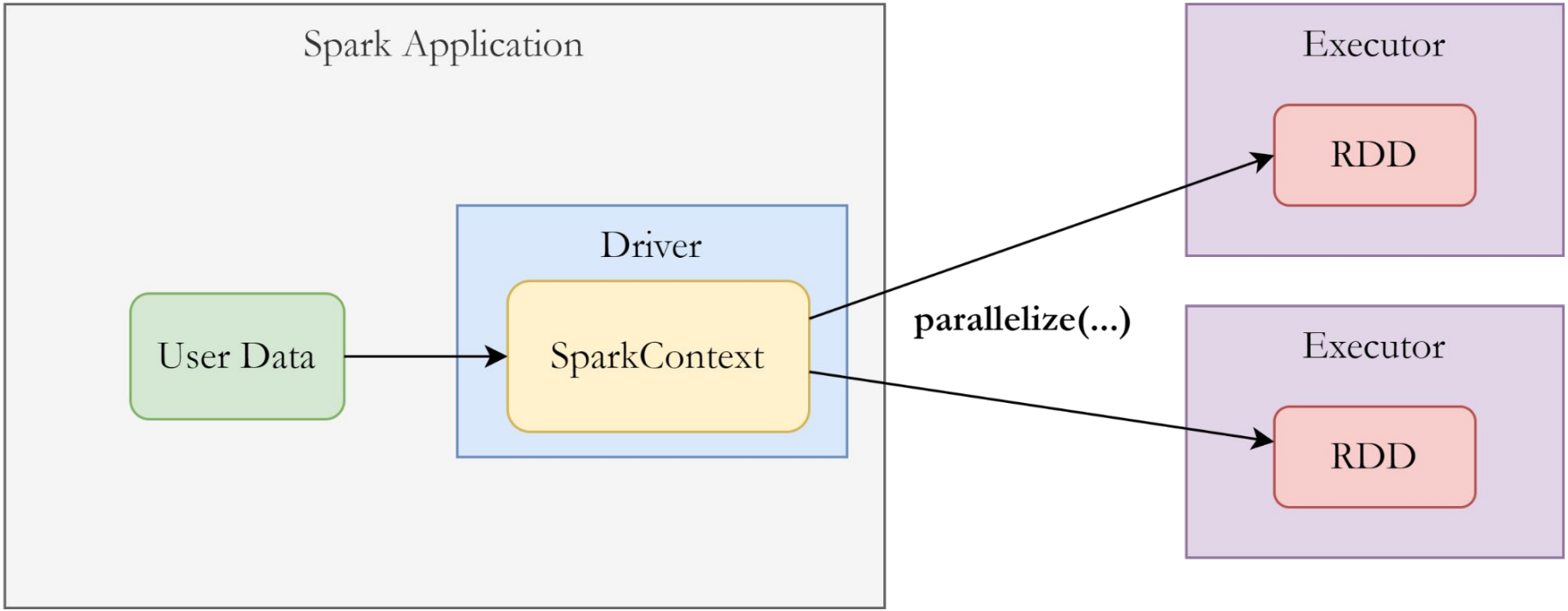
RDD: Resilient Distributed Dataset

Distributed

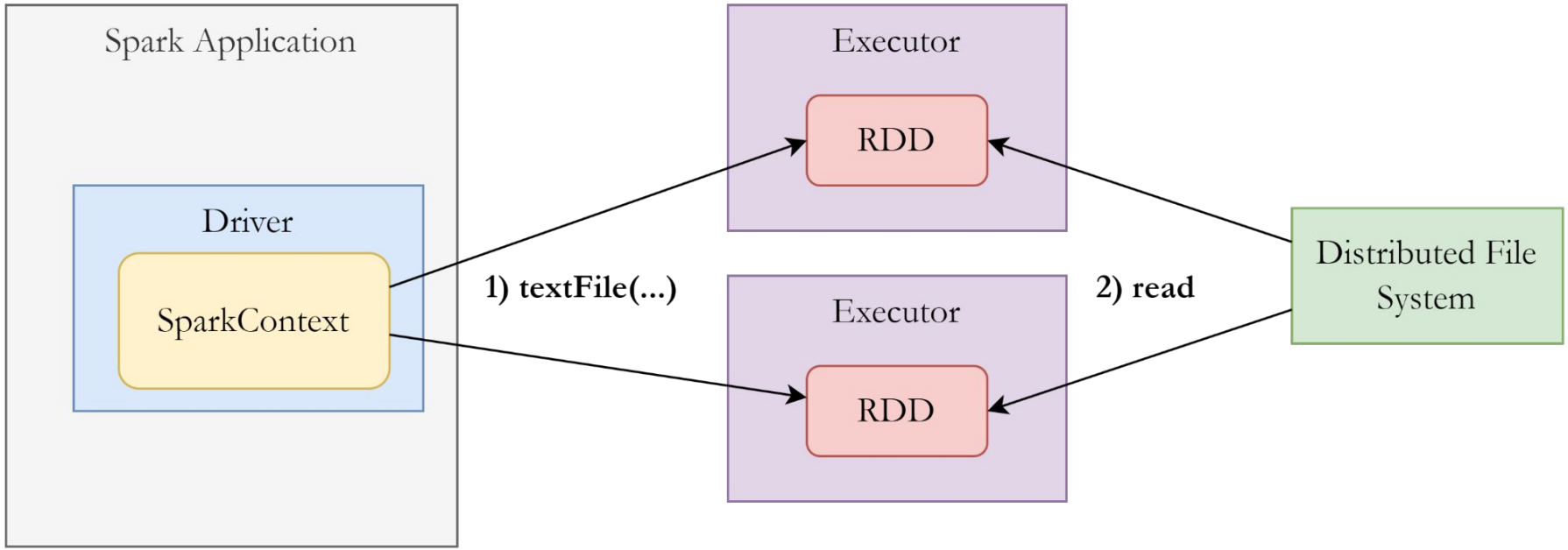


How to create a RDD

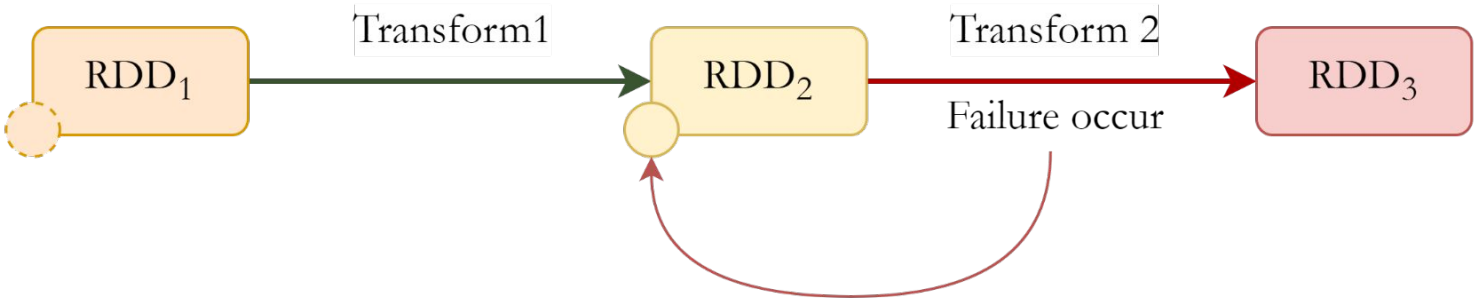
Manually distributed

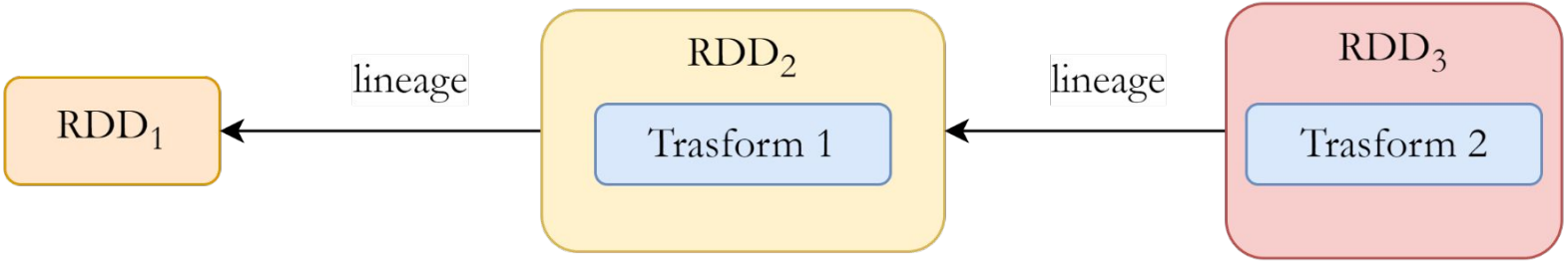


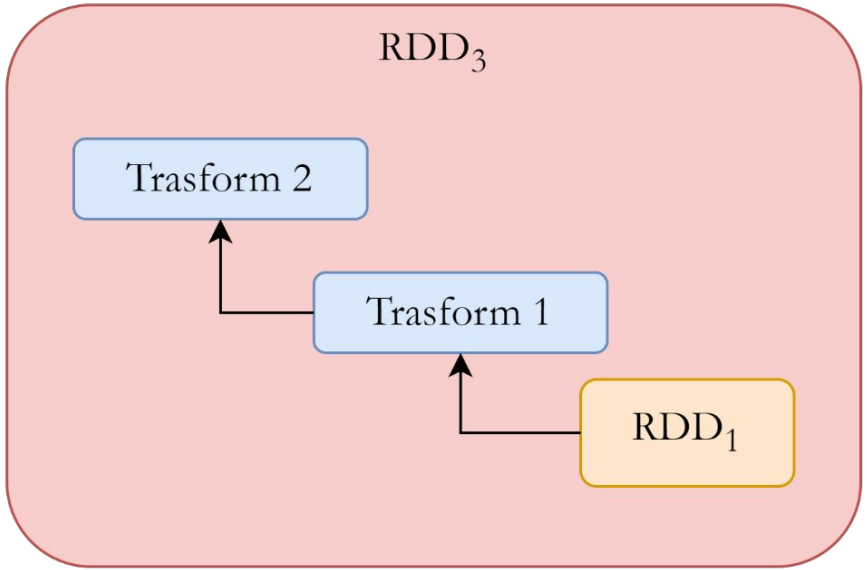
Spark provided function



Resilient


RDD1
Transform1
RDD2
Transform 2
RDD3
Failure occur


RDD1
lineage
RDD2
Trasform 1
lineage
RDD3
Trasform 2



Prefer memory-only processing,
but it can use disk it is necessary

Prefer recomputation to cache temporary result,
but it can cache for accerlating computation time.

RDD Operation (Lazy Evaluation)

How to call RDD operation

In Spark Application

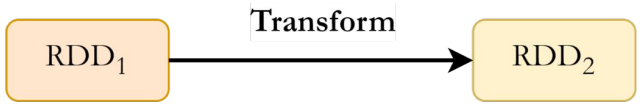
[rdd_handle].[operation_name](arg1, arg2, ...)

Transform

Generate anothe RDD (create a data processing lineage)

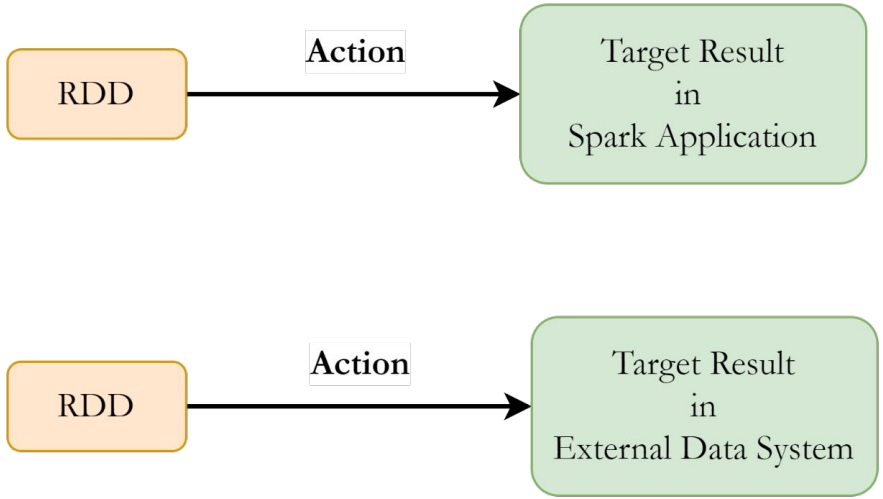
No operation happens in Spark cluster

Allow Spark to optimize operations when execute all combined operations in lineage

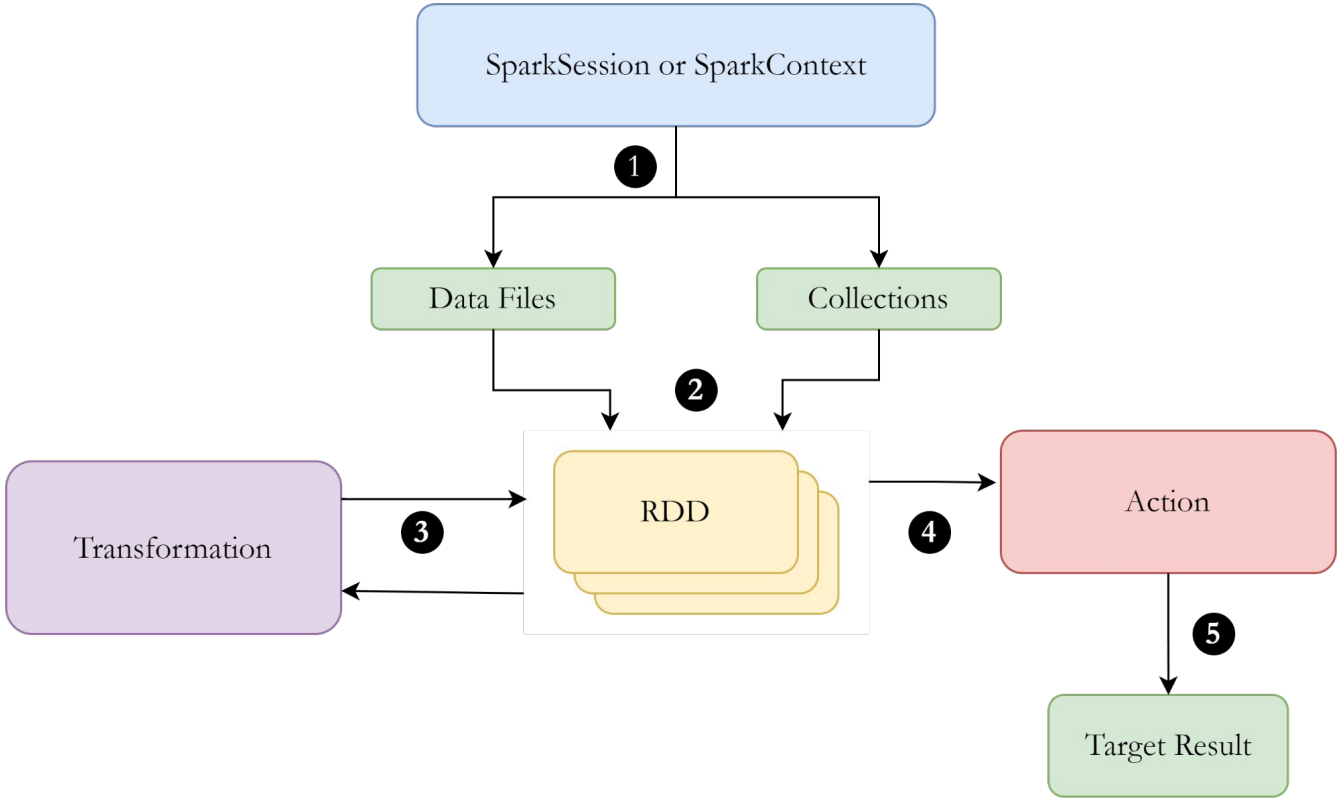


Action

- Trigger the Spark cluster to processing as to the processing lineage in RDD which is called
- Some actions send result back the driver
  (Beware: memory limit on the driver machine)
- Some actions have the executors write the result to specified destination
  (Beware: the destination should support intensive write if the data is big)

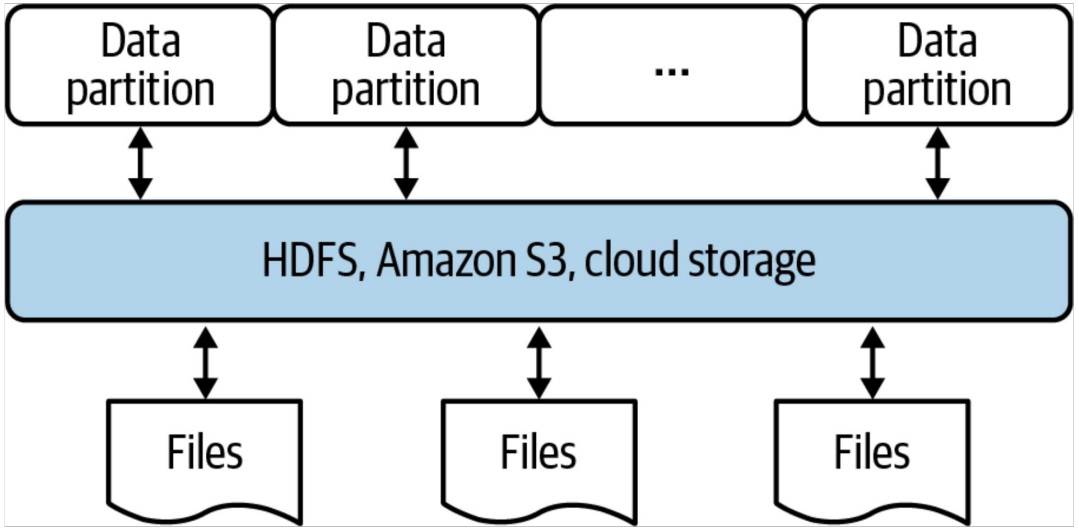


Working with RDD summary

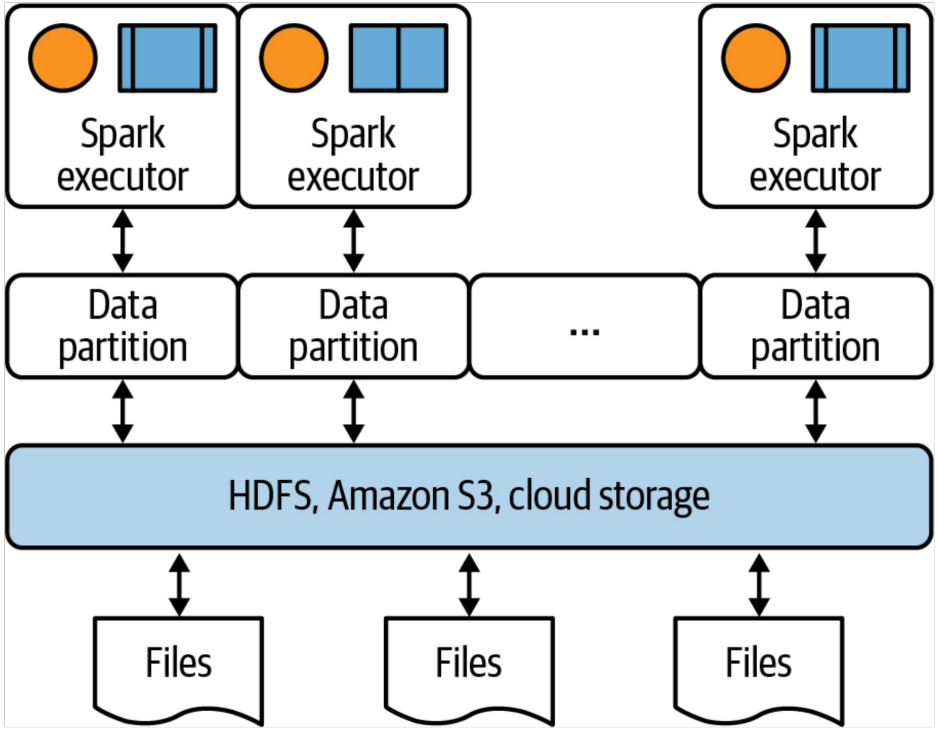


Partition and Shuffle

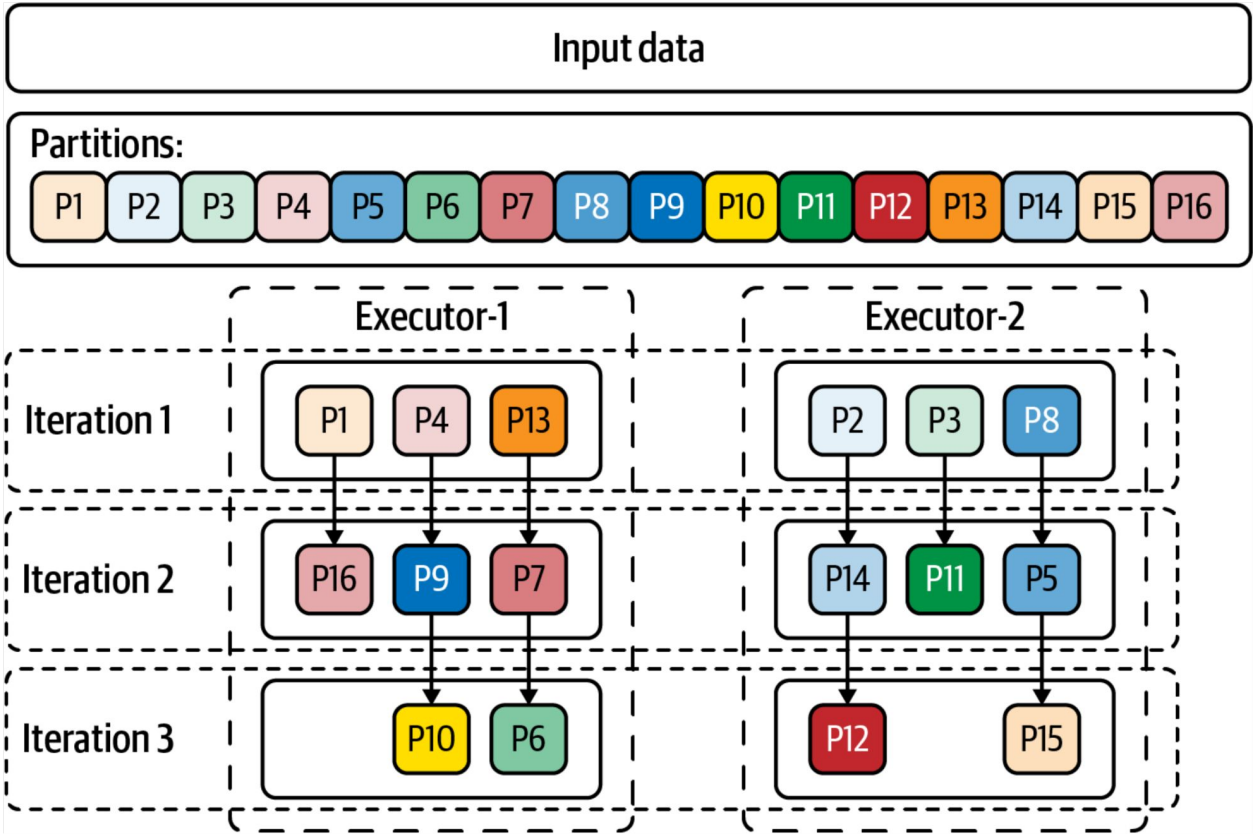
Source: Data Algorithms in Spark



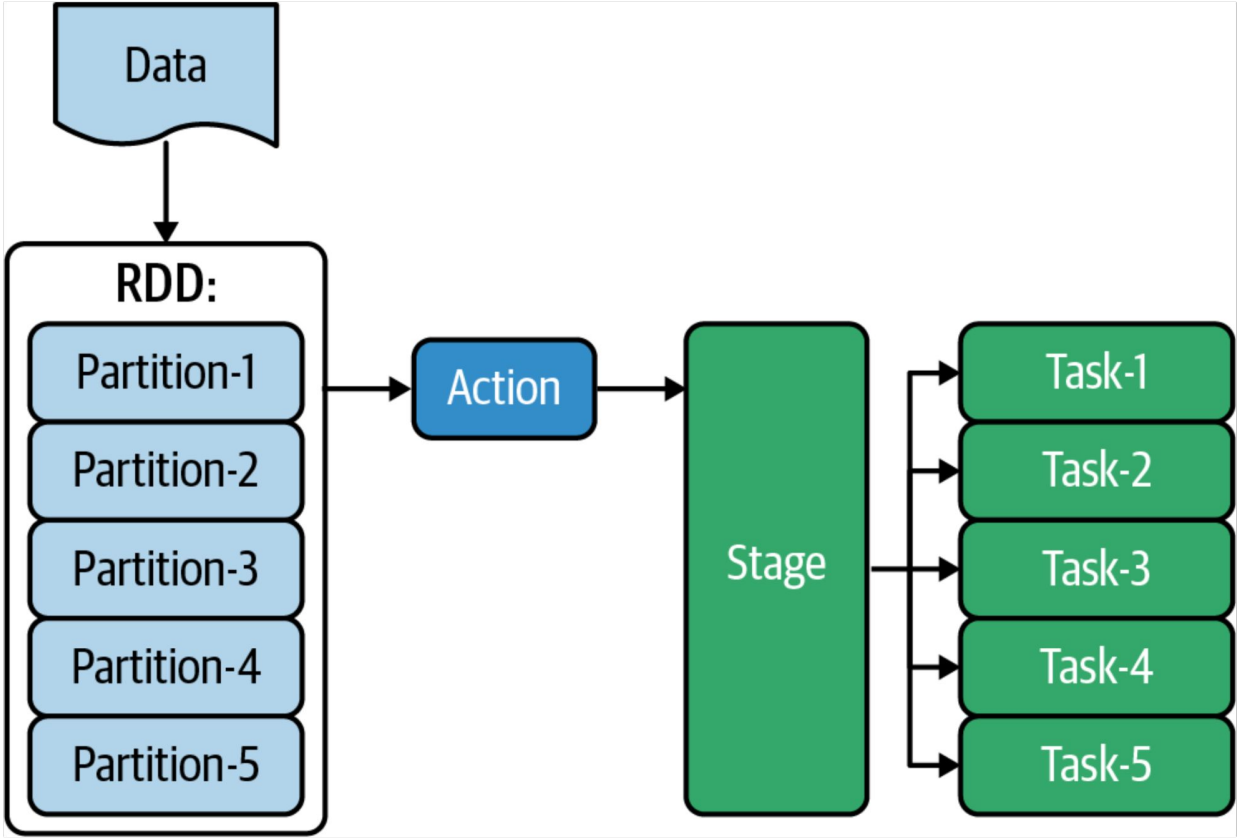
Source: Data Algorithms in Spark



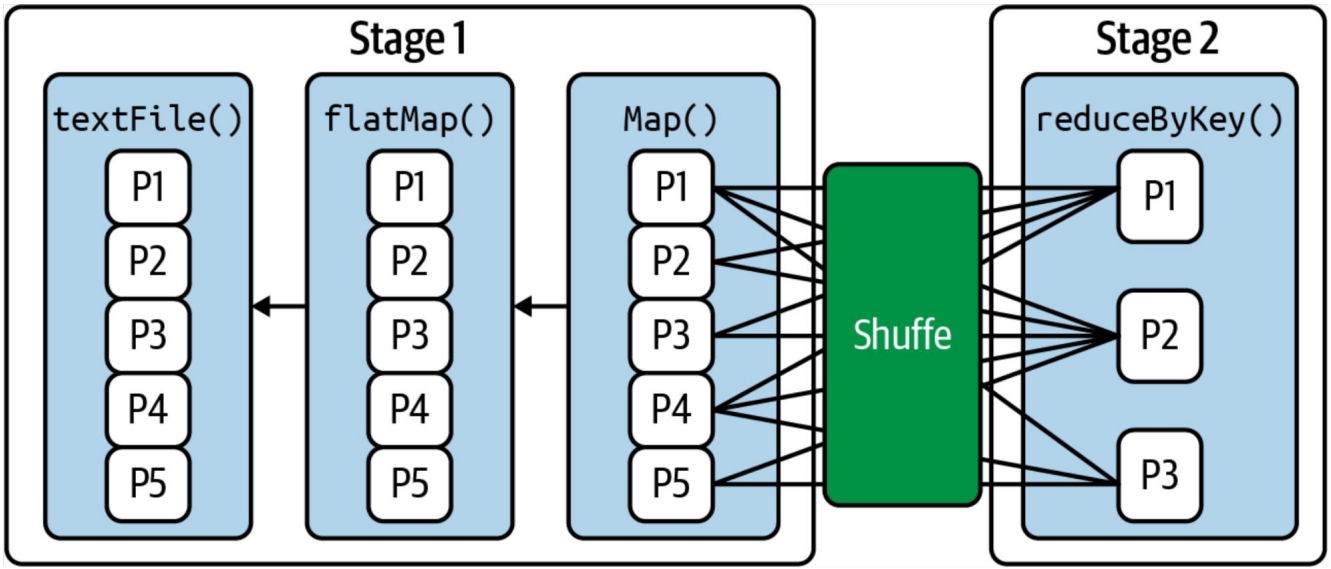
Source: Data Algorithms in Spark



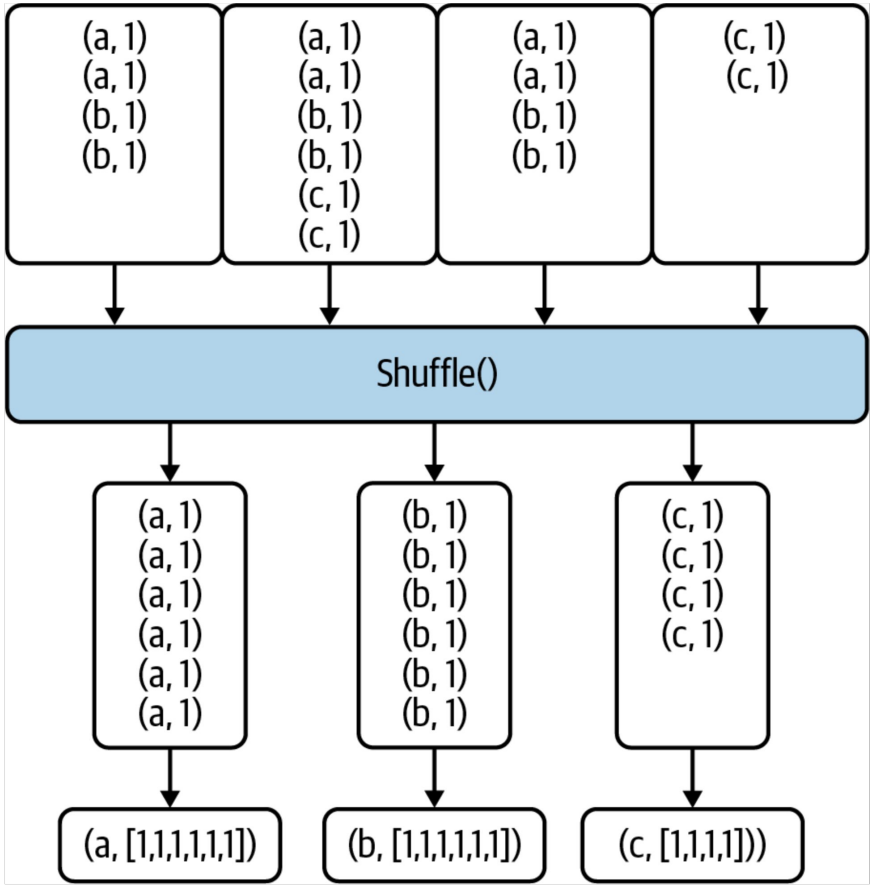
Source: Data Algorithms in Spark



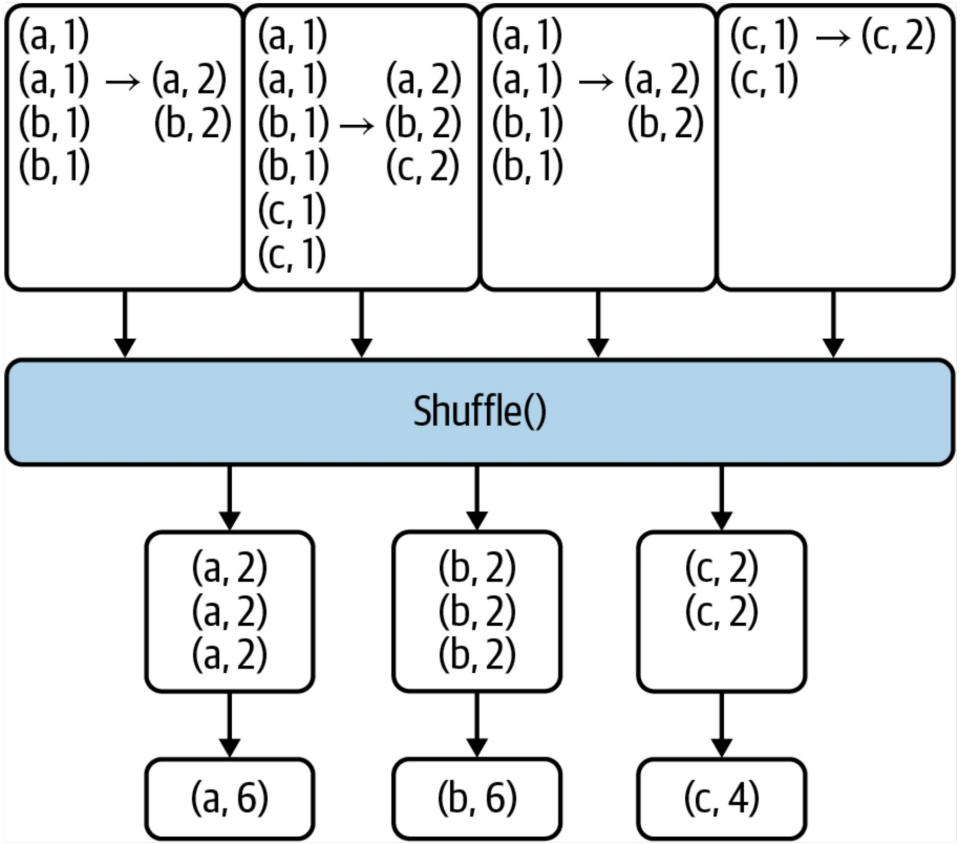
Source: [Data Algorithms in Spark]



Source: Data Algorithms in Spark

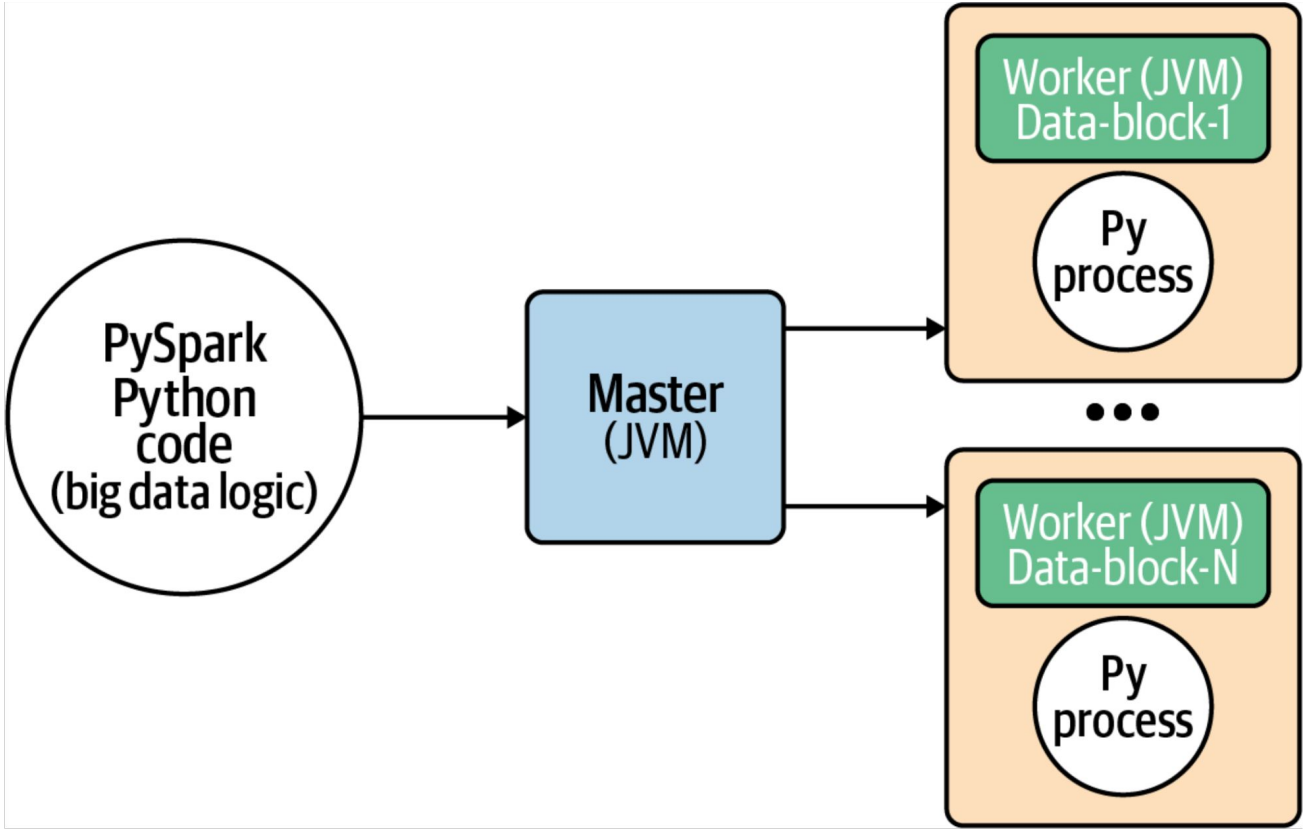


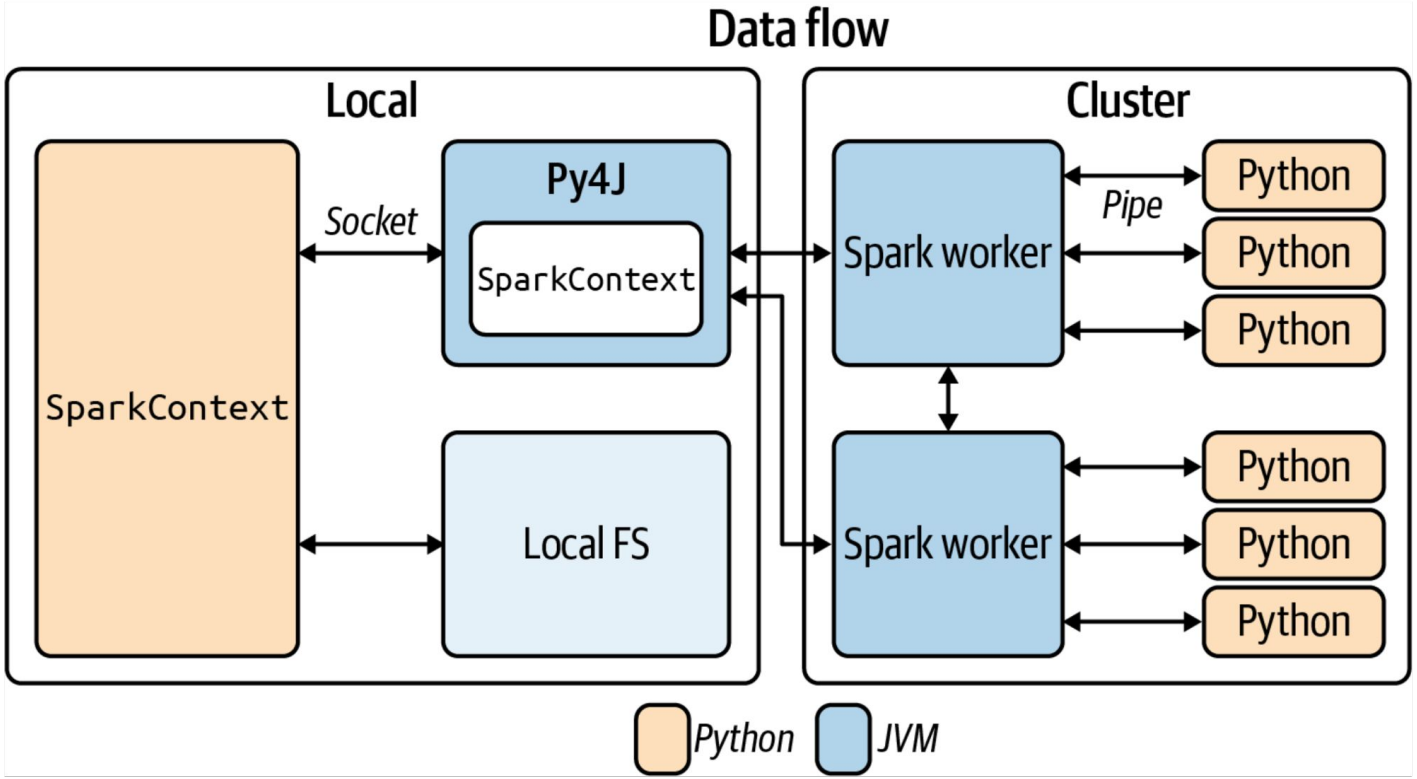
Source: Data Algorithms in Spark



DataFrame

Source: [Data Algorithms in Spark]




Data flow
Local
Cluster
SparkContext
Socket
Py4J
SparkContext
Local FS
Spark worker
Spark worker
Pipe
Python
Python
Python
Python
Python
Python
Python
JVM

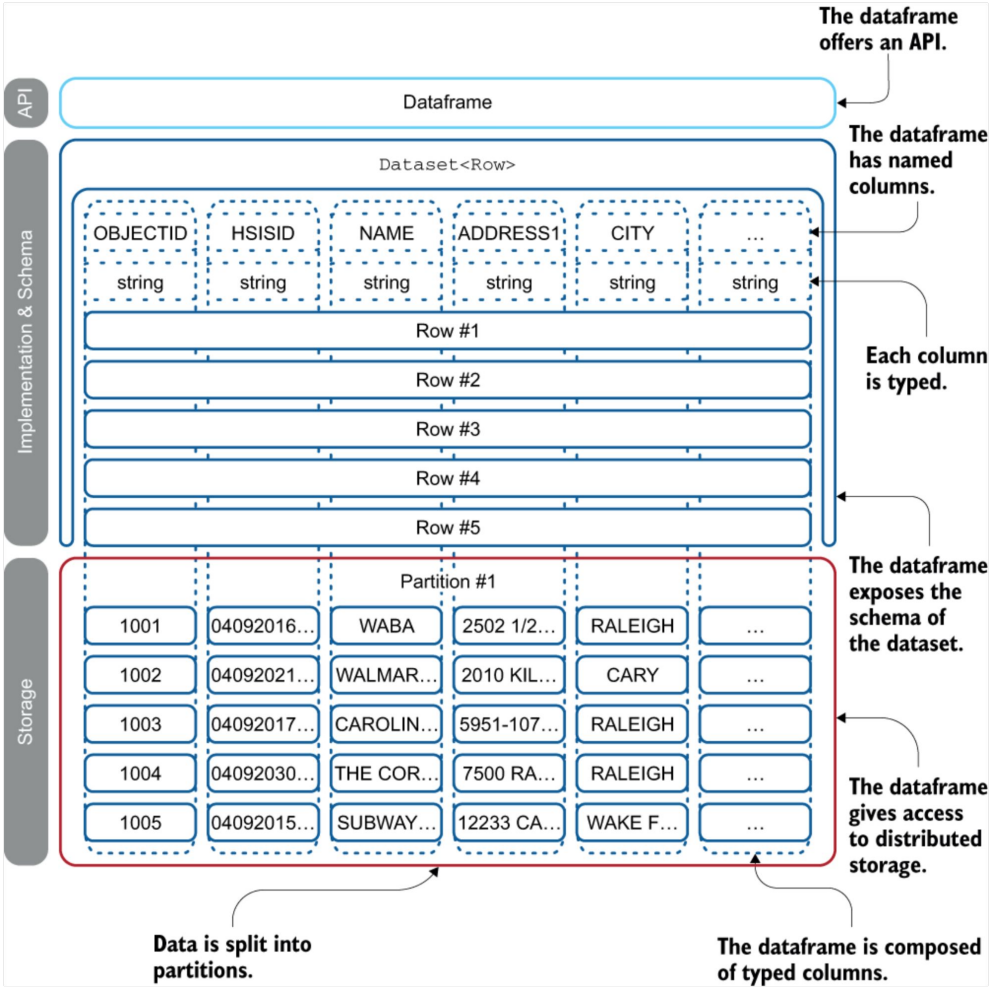
rdd.[transform_or_action] ([python function])

Intra-Process Seralization Cost

dataframe.[operation](arg1, arg2, ...)

No intra-process serialization cost

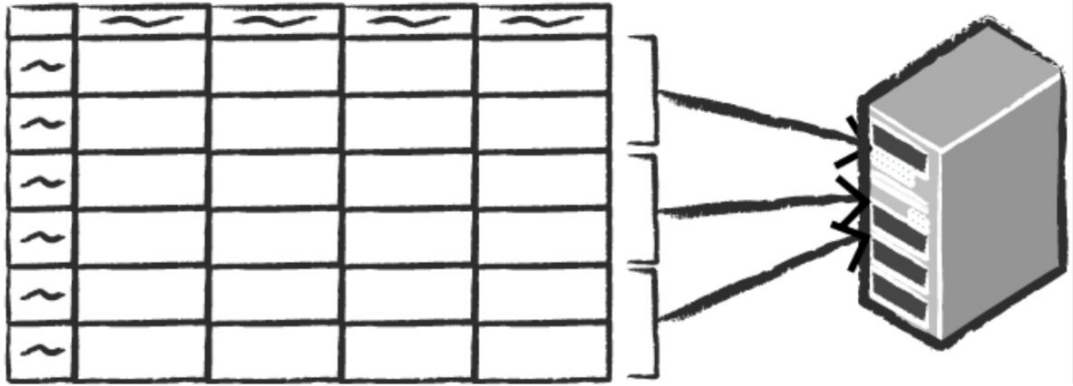
Almost popular tools have concepts of DataFrame


The dataframe offers an API.
API
Dataframe
The dataframe has named columns.
Implementation & Schema
Dataset<Row>
OBJECTID
HSISID
NAME
ADDRESS1
CITY
...
string
string
string
string
string
string
Row #1
Row #2
Row #3
Row #4
Row #5
Each column is typed.
The dataframe exposes the schema of the dataset.
Storage
Partition #1
1001
04092016...
WABA
2502 1/2...
RALEIGH
...
1002
04092021...
WALMAR...
2010 KIL...
CARY
...
1003
04092017...
CAROLIN...
5951-107...
RALEIGH
...
1004
04092030...
THE COR...
7500 RA...
RALEIGH
...
1005
04092015...
SUBWAY...
12233 CA...
WAKE F...
...
The dataframe gives access to distributed storage.
Data is split into partitions.
The dataframe is composed of typed columns.

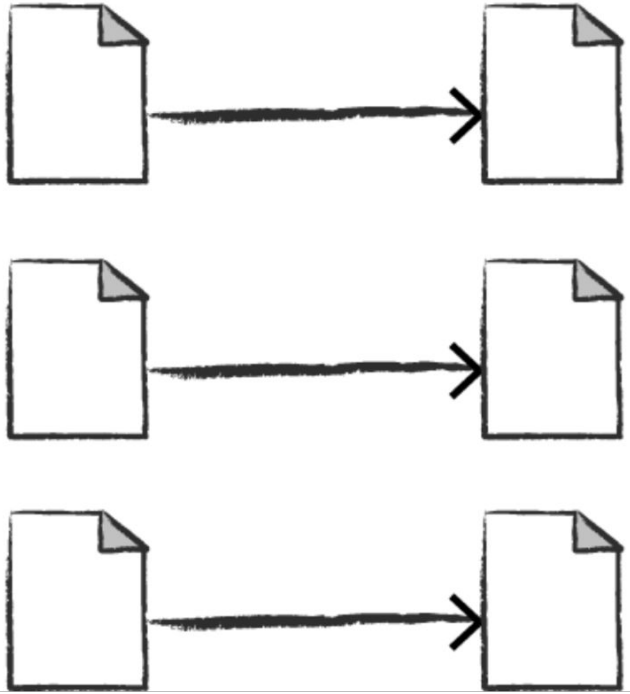
Source: [Spark The Definitive Guide]

Spreadsheet on a single machine

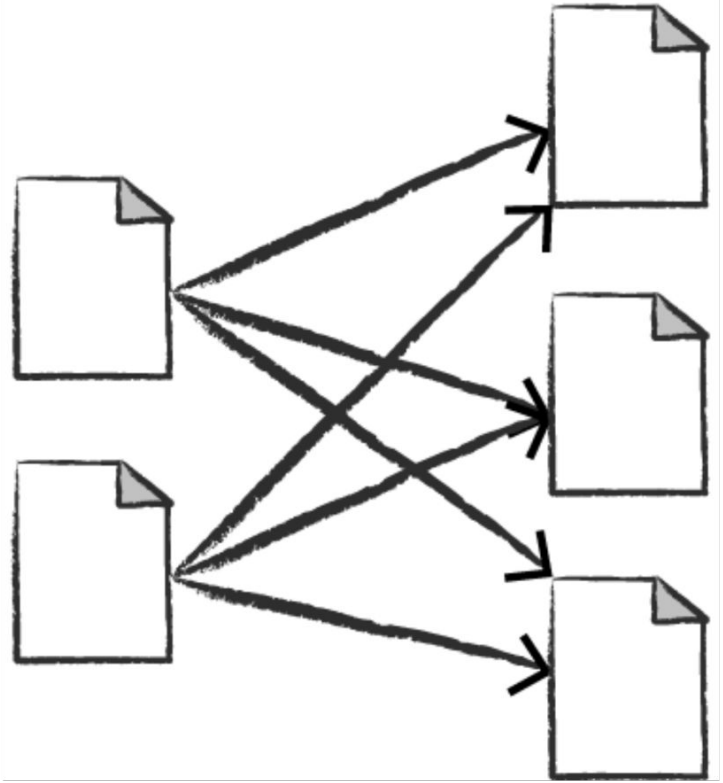
Table or Data Frame partitioned across servers in a data center

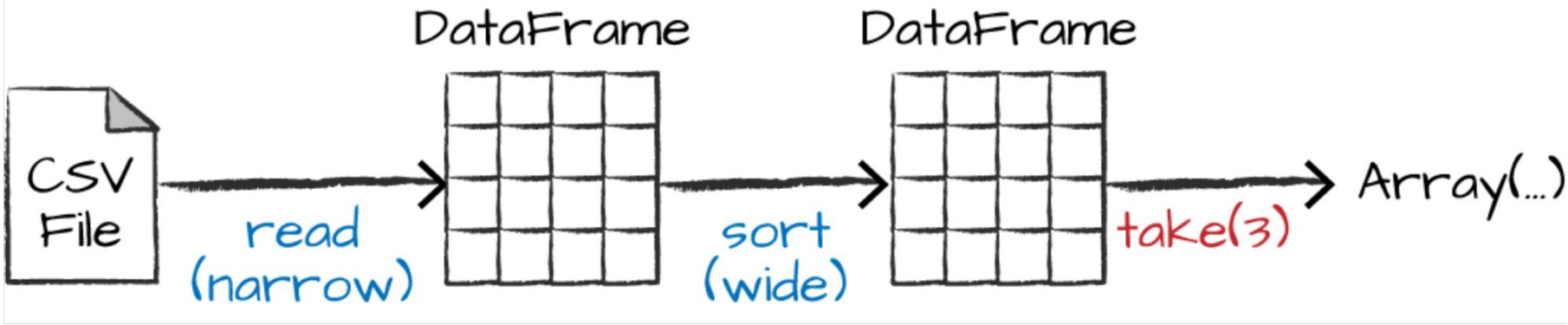


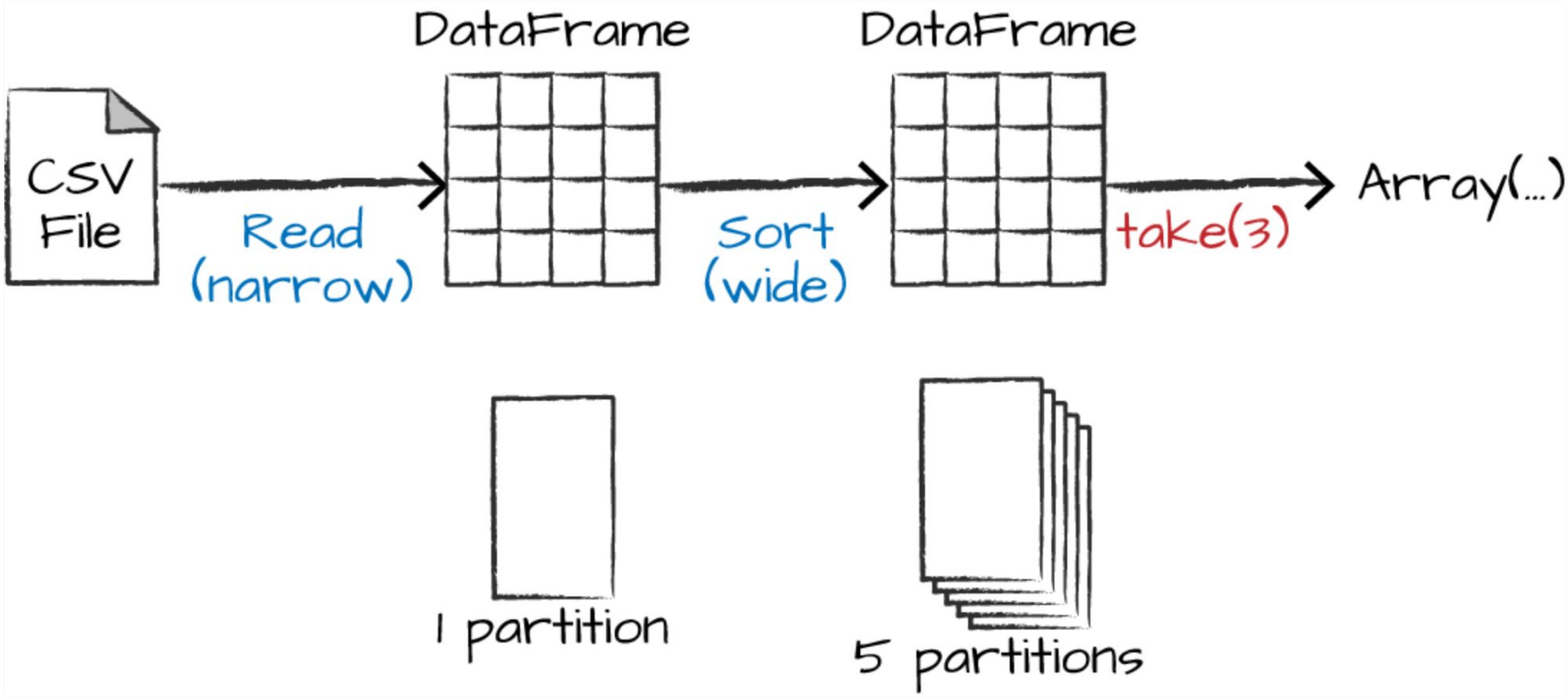
Narrow transformations
1 to 1

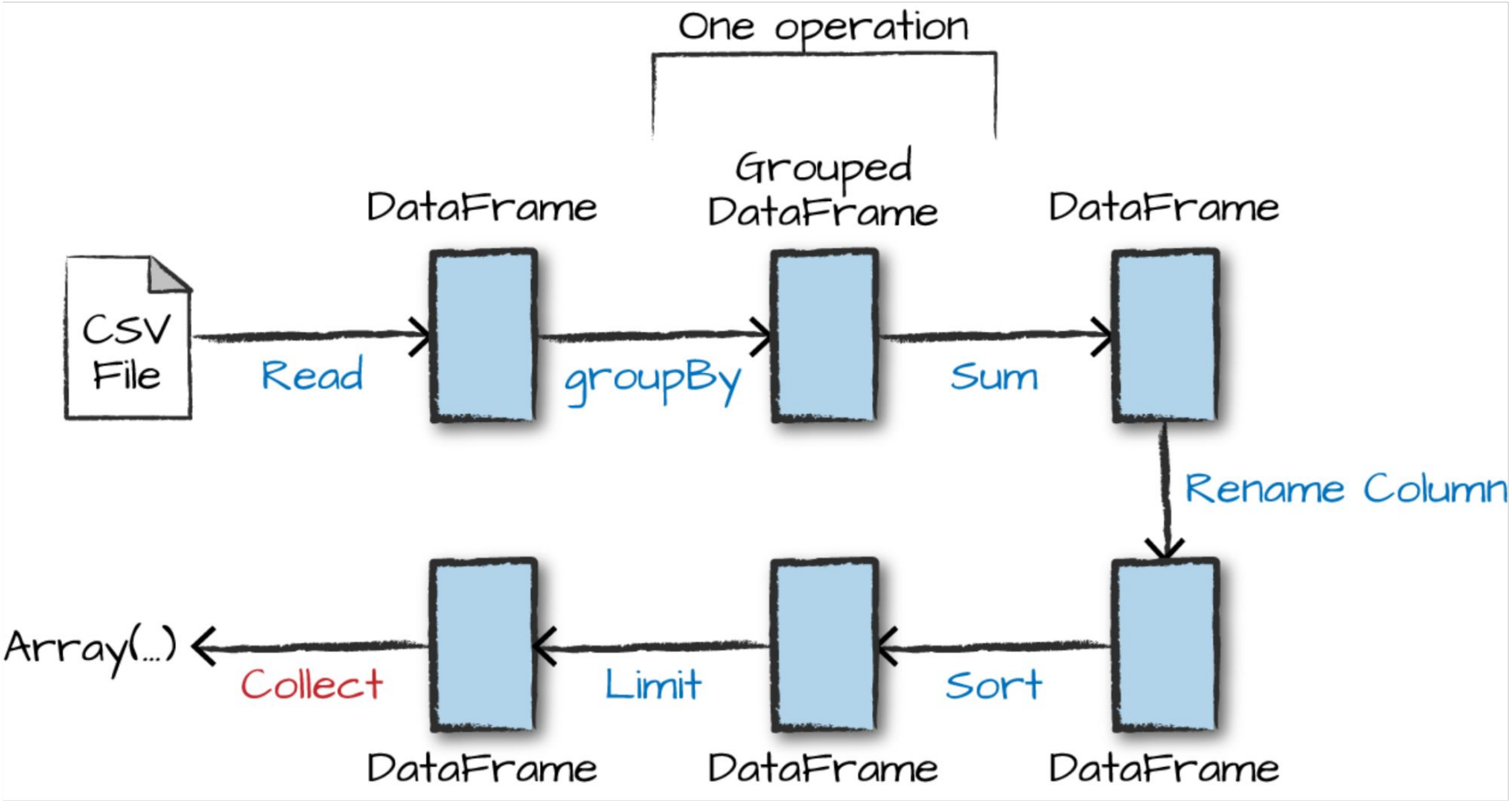


Wide transformations
(shuffles) 1 to N



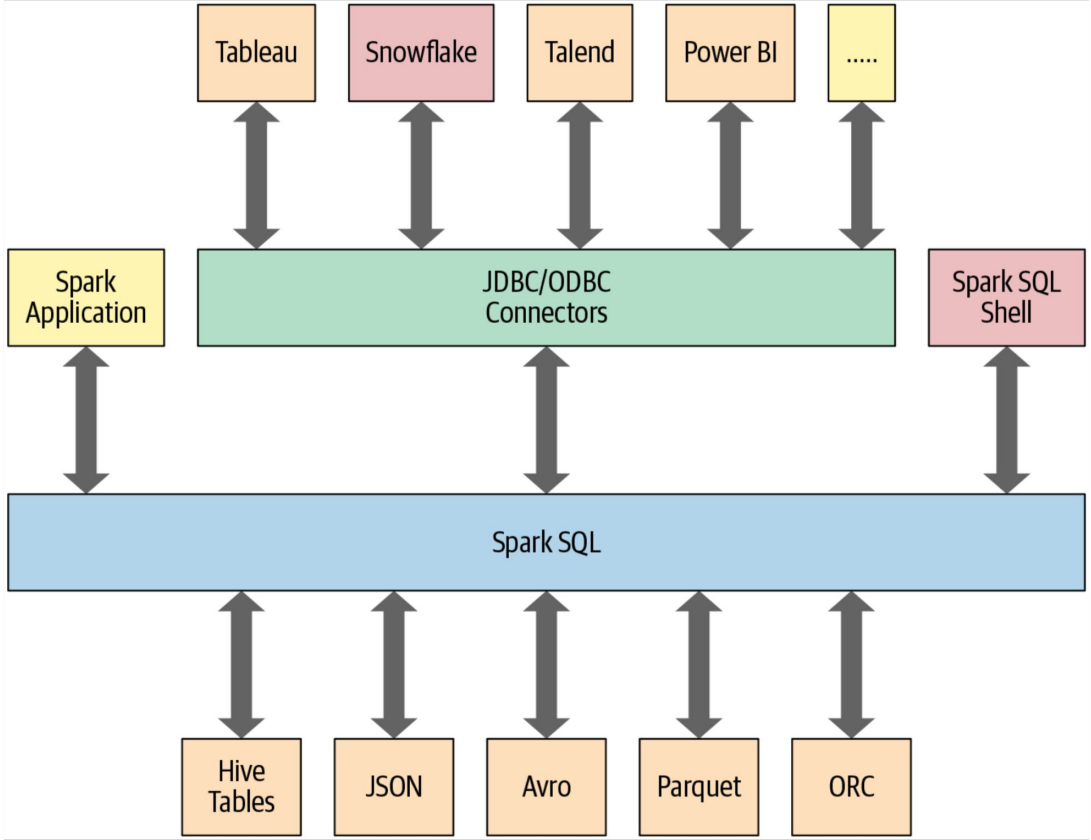

DataFrame
DataFrame
CSV File
read (narrow)
sort (wide)
take(3)
Array(...)

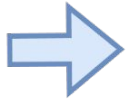

DataFrame
DataFrame
CSV File
Read (narrow)
Sort (wide)
take(3)
Array(...)
1 partition
5 partitions


One operation
Grouped DataFrame
DataFrame
DataFrame
CSV File
Read
groupBy
Sum
Rename Column
Array(...)
Collect
Limit
Sort
DataFrame
DataFrame
DataFrame

SparkSQL

Source: Learning Spark



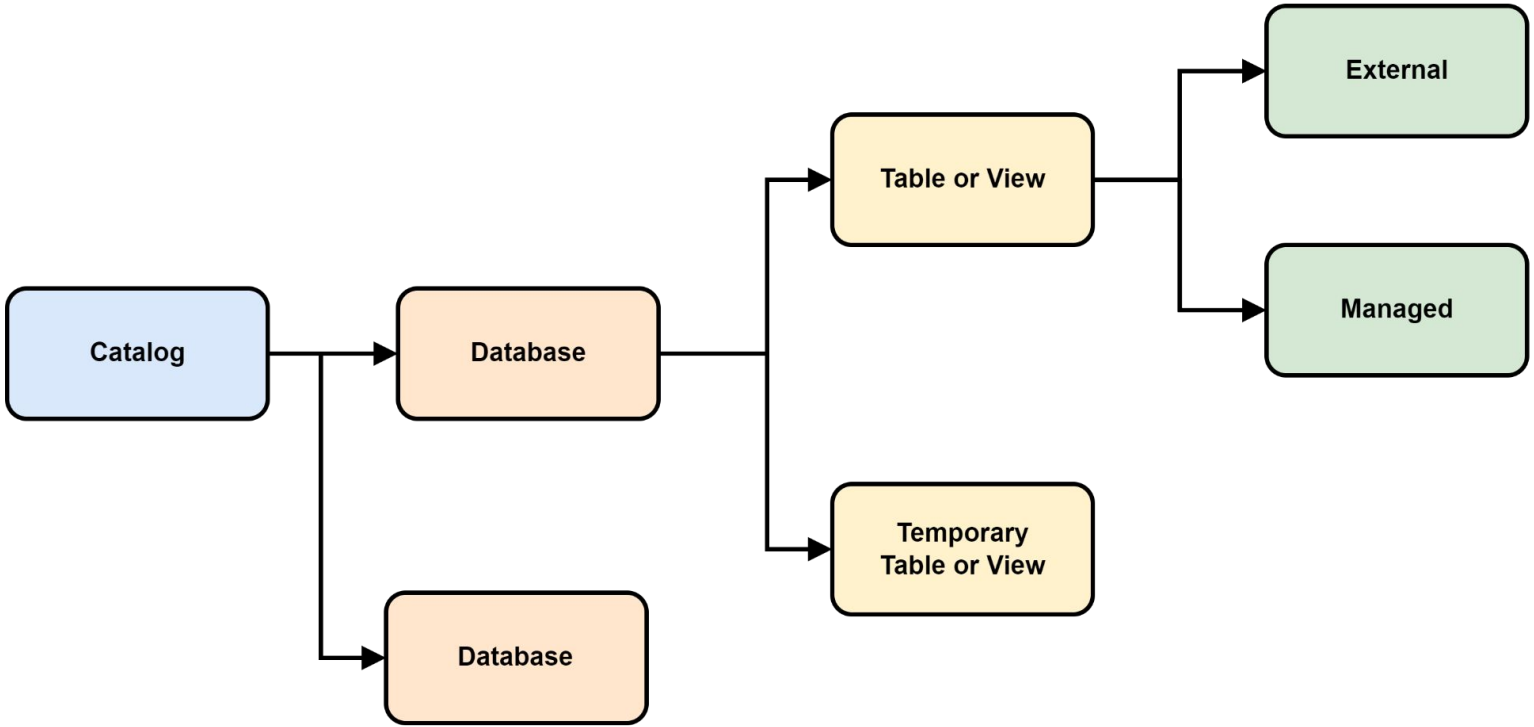
spark.sql([SQL query])  **DataFrame***

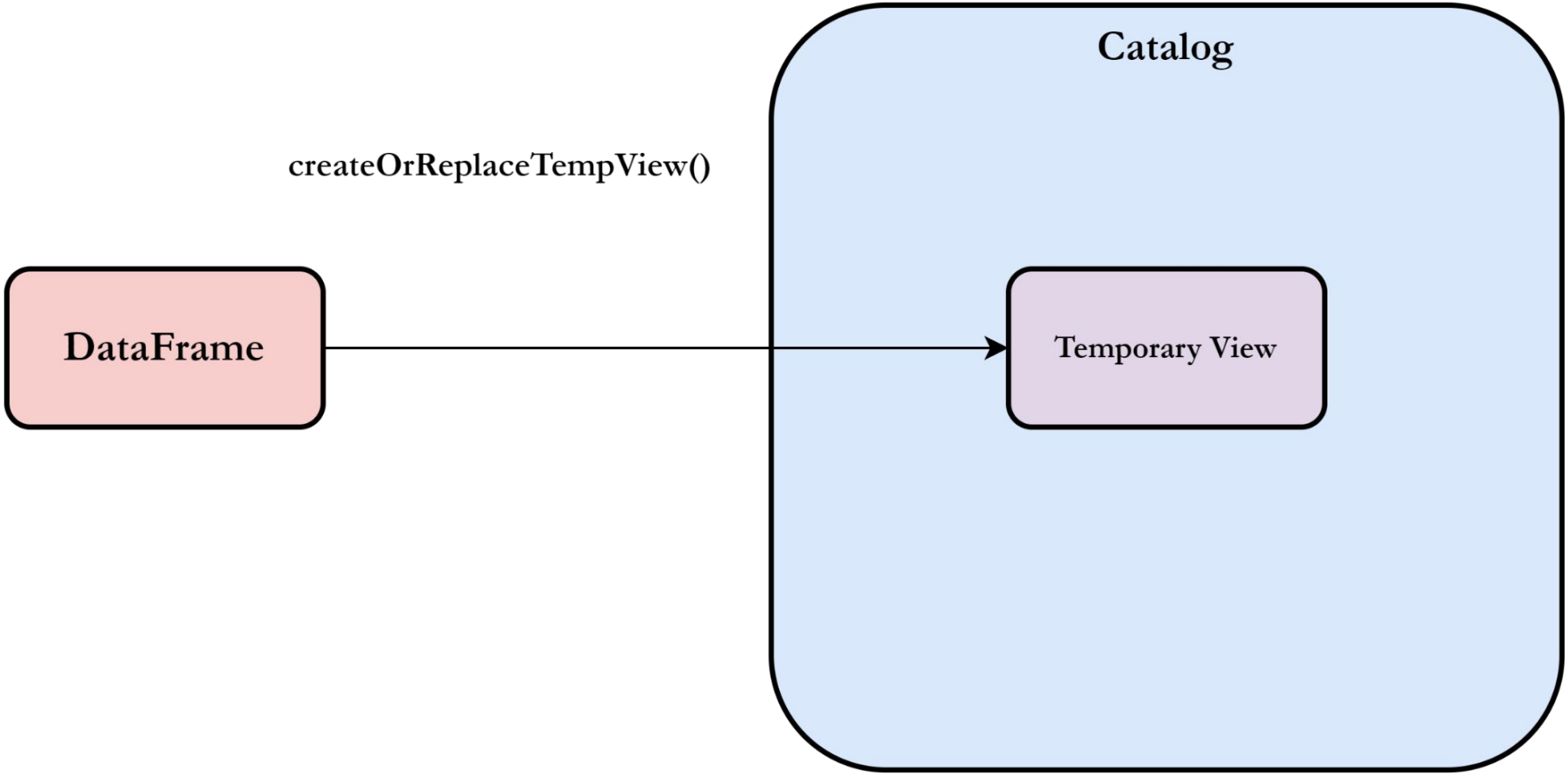
How spark knows about databases and tables to process SQL ?

* SQL command like INSERT does not return a dataframe

Spark Catalog

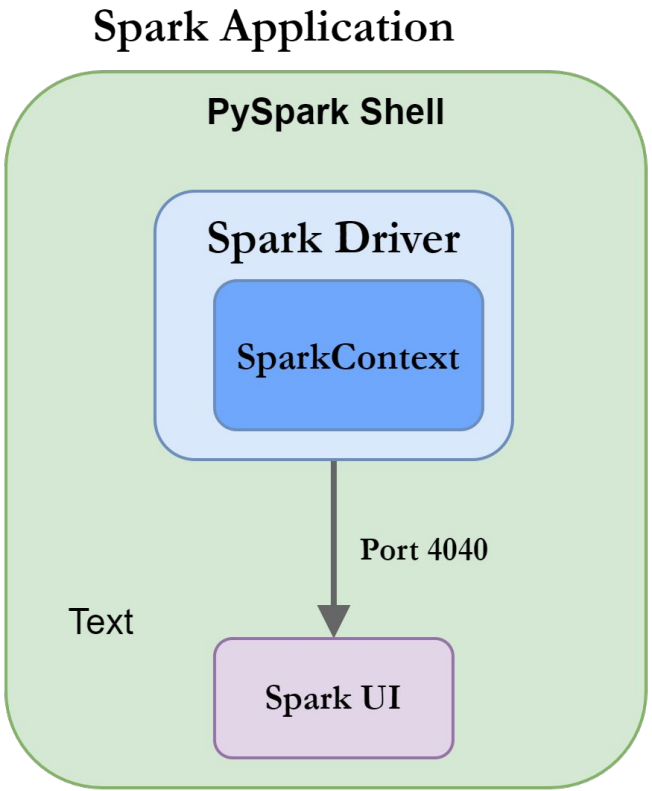
Catalog is just a metadata how to use a stored data: access, read, write



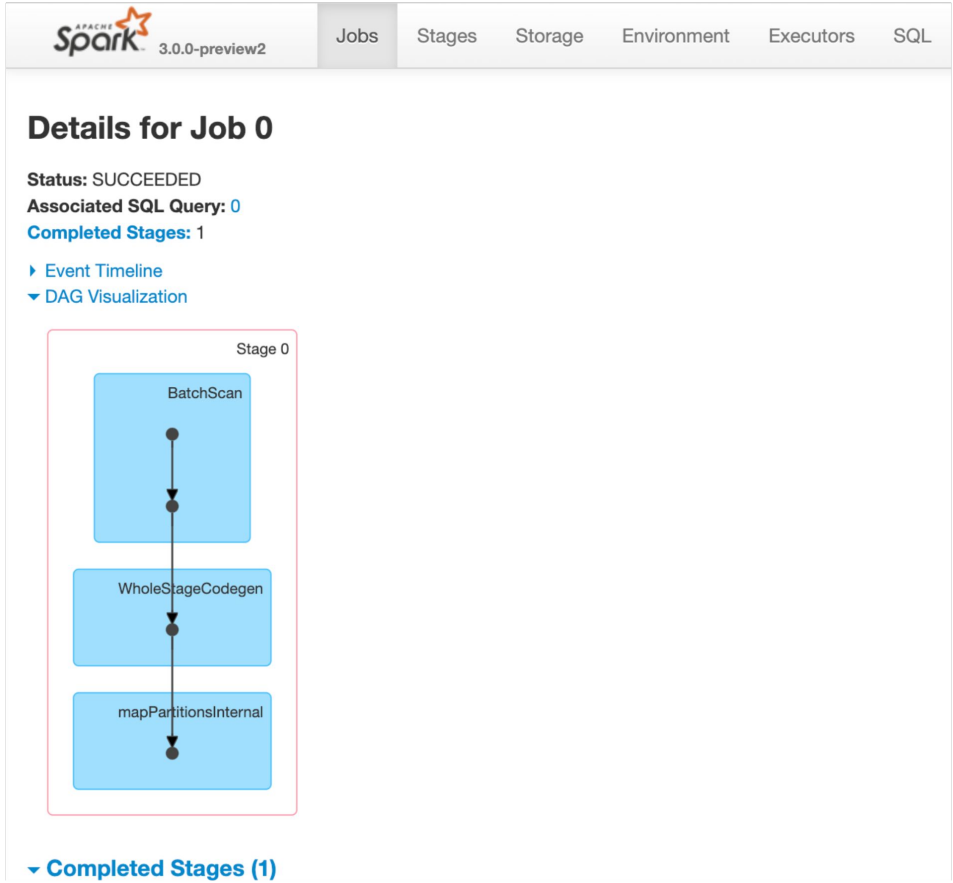

Catalog
createOrReplaceTempView()
DataFrame
Temporary View

SparkUI

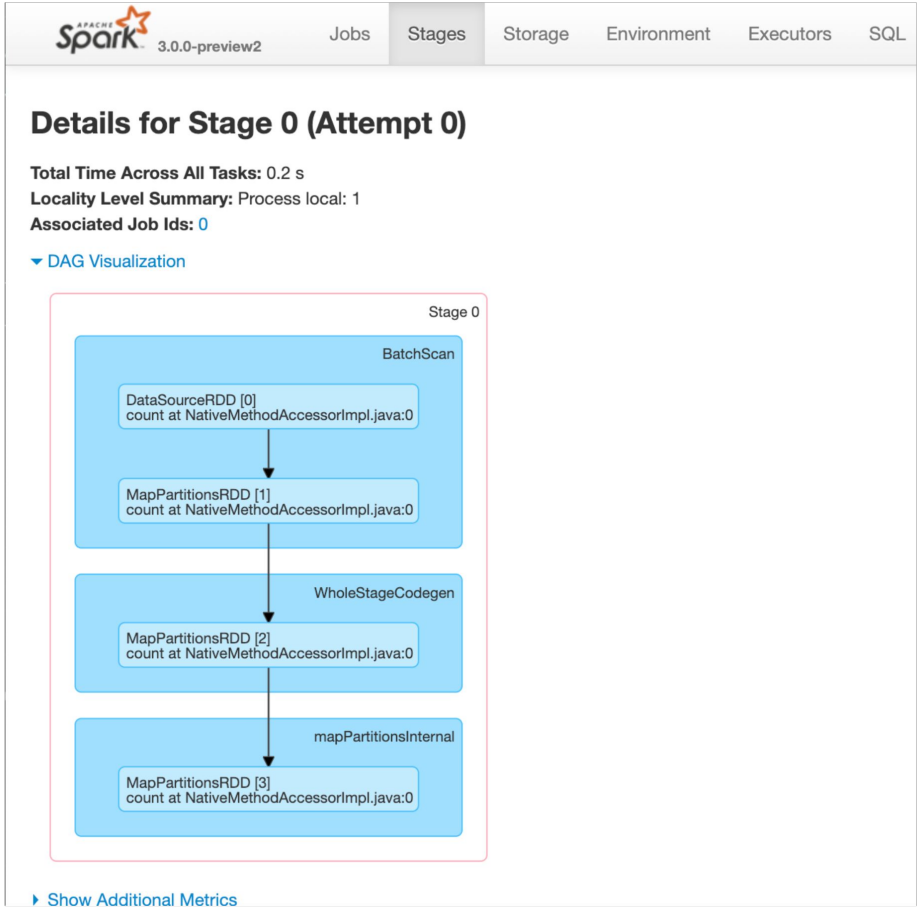
Monitoring and Tracking How an **Application** interact with **Spark**



Jobs


Spark 3.0.0-preview2
Jobs
Stages
Storage
Environment
Executors
SQL
Details for Job 0
Status: SUCCEEDED
Associated SQL Query: 0
Completed Stages: 1
Event Timeline
DAG Visualization
Stage 0
BatchScan
WholeStageCodegen
mapPartitionsInternal
Completed Stages (1)

Stages


Spark 3.0.0-preview2
Jobs
Stages
Storage
Environment
Executors
SQL
Details for Stage 0 (Attempt 0)
Total Time Across All Tasks: 0.2 s
Locality Level Summary: Process local: 1
Associated Job Ids: 0
DAG Visualization
Stage 0
BatchScan
DataSourceRDD [0]
count at NativeMethodAccessorImpl.java:0
MapPartitionsRDD [1]
count at NativeMethodAccessorImpl.java:0
WholeStageCodegen
MapPartitionsRDD [2]
count at NativeMethodAccessorImpl.java:0
mapPartitionsInternal
MapPartitionsRDD [3]
count at NativeMethodAccessorImpl.java:0
Show Additional Metrics

Environment