SPARK EXECUTION

So far, we've interacted with Spark

1. Using an REPL environment
2. Submitting a script

These are ways for the user to give instructions to Spark

Spark takes the user's instructions

It translates them into tasks that run across a computing cluster

It returns the results to the user

How does this work?

When you run a program (using PySpark or spark-submit)

Spark prints a bunch of messages

First, there are a lot of messages initializing things

```
INFO util.Utils: Successfully started service 'sparkDriver' on port 64191.
INFO slf4j.Slf4jLogger: Slf4jLogger started
INFO Remoting: Starting remoting
INFO Remoting: Remoting started; listening on addresses :[akka.tcp://sparkDriverActorSyst
INFO util.Utils: Successfully started service 'sparkDriverActorSystem' on port 64192.
INFO spark.SparkEnv: Registering MapOutputTracker
INFO spark.SparkEnv: Registering BlockManagerMaster
```

Then you can see the progress of your processing instructions

```
spark.SparkContext: Starting job: saveAsTextFile at
scheduler.DAGScheduler: Got job 0 (saveAsTextFile at
scheduler.DAGScheduler: Submitting ResultStage 0 (MapPar
cluster.YarnScheduler: Adding task set 0.0 with 2 tasks
scheduler.TaskSetManager: Starting task 0.0 in stage 0.0
scheduler.TaskSetManager: Starting task 1.0 in stage 0.0
scheduler.TaskSetManager: Finished task 1.0 in stage 0.0
scheduler.TaskSetManager: Finished task 0.0 in stage 0.0
scheduler.DAGScheduler: ResultStage 0 (saveAsTextFile at
scheduler.DAGScheduler: Job 0 finished: saveAsTextFile
```

There seem to be many components involved in the processing

```
spark.SparkContext: Starting job: saveAsTextFile at
scheduler.DAGScheduler: Got job 0 (saveAsTextFile at
scheduler.DAGScheduler: Submitting ResultStage 0 (MapPar
cluster.YarnScheduler: Adding task set 0.0 with 2 tasks
scheduler.TaskSetManager: Starting task 0.0 in stage 0.0
scheduler.TaskSetManager: Starting task 1.0 in stage 0.0
scheduler.TaskSetManager: Finished task 1.0 in stage 0.0
scheduler.TaskSetManager: Finished task 0.0 in stage 0.0
scheduler.DAGScheduler: ResultStage 0 (saveAsTextFile at
scheduler.DAGScheduler: Job 0 finished: saveAsTextFile
```

What are all of these components doing?

SparkContext

DAGScheduler

YarnScheduler

TaskSetManager

SparkContext

Every Spark application has a driver program

DAGScheduler

YarnScheduler

TaskSetManager

SparkContext

driver program

This could be

The PySpark/Scala shell

A script

A main function
(Java/Scala)

DAGScheduler

YarnScheduler

TaskSetManager

SparkContext

driver program

This program consists of instructions to Spark

Load data into RDDs

Perform operations on RDDS

DAGScheduler

YarnScheduler

TaskSetManager

driver program

SparkContext

Driver programs use a SparkContext to communicate with the Spark cluster

DAGScheduler

YarnScheduler

TaskSetManager

driver program

SparkContext

There are 2 phases involved in the driver program

Initial setup

Job Run

DAGScheduler

YarnScheduler

TaskSetManager

driver program

SparkContext

Initial setup

This happens when you launch the program (the shell or your script)

Job Run

DAGScheduler

YarnScheduler

TaskSetManager

driver program

SparkContext

Initial setup

A Job run is initiated whenever the program has a processing task

Job Run

Ex: An Action like count() or collect()

DAGScheduler

TaskScheduler

TaskSetManager

Initial setup

driver program

SparkContext

Spark needs a separate cluster manager to manage resources across the cluster

This is a plug and play component : Mesos, YARN or Spark Standalone

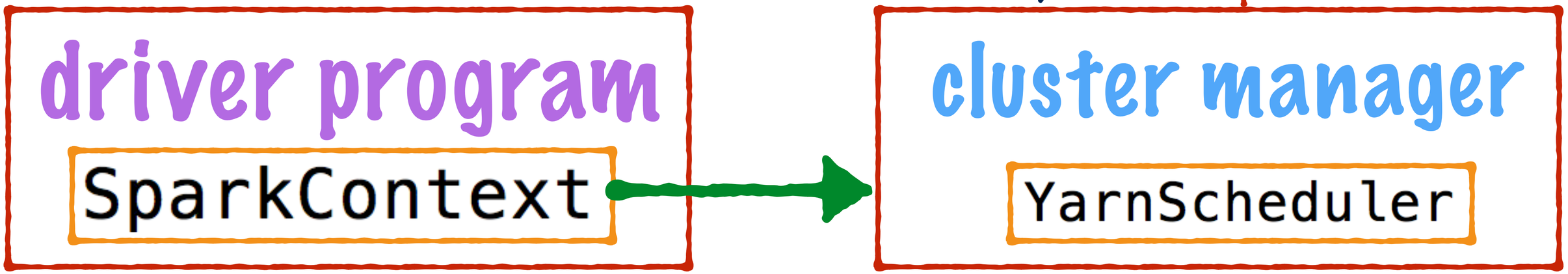
DAGScheduler

YarnScheduler

TaskSetManager

Initial setup

Mesos, YARN or Spark Standalone



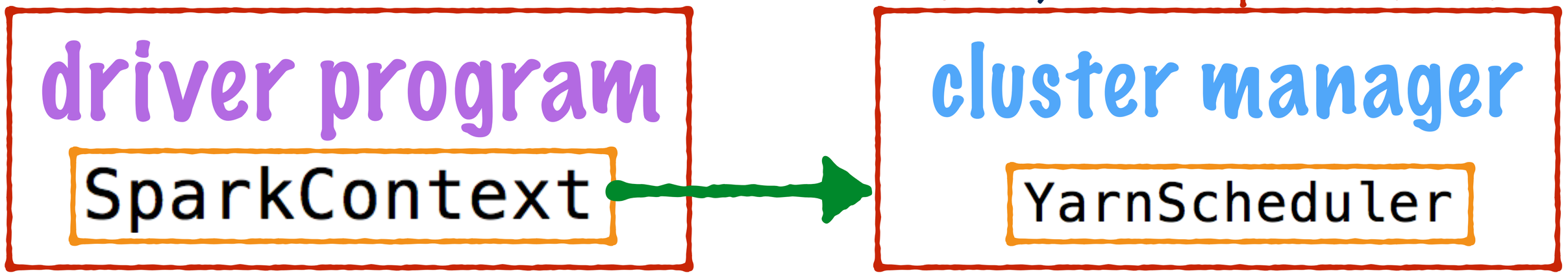
When the driver program starts

It contacts the cluster manager through the SparkContext

DAGScheduler

TaskSetManager

Initial setup



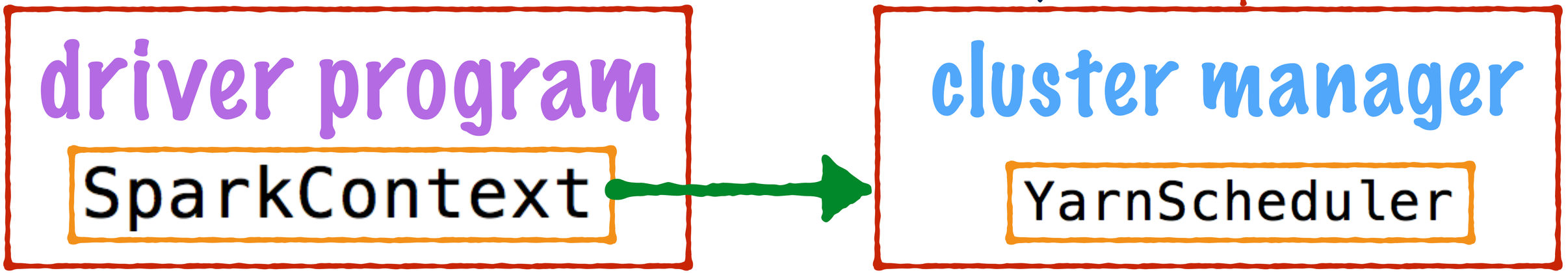
The cluster manager in turn launches Java processes on several nodes in the cluster

DAGScheduler

TaskSetManager

Initial setup

Mesos, YARN or Spark Standalone



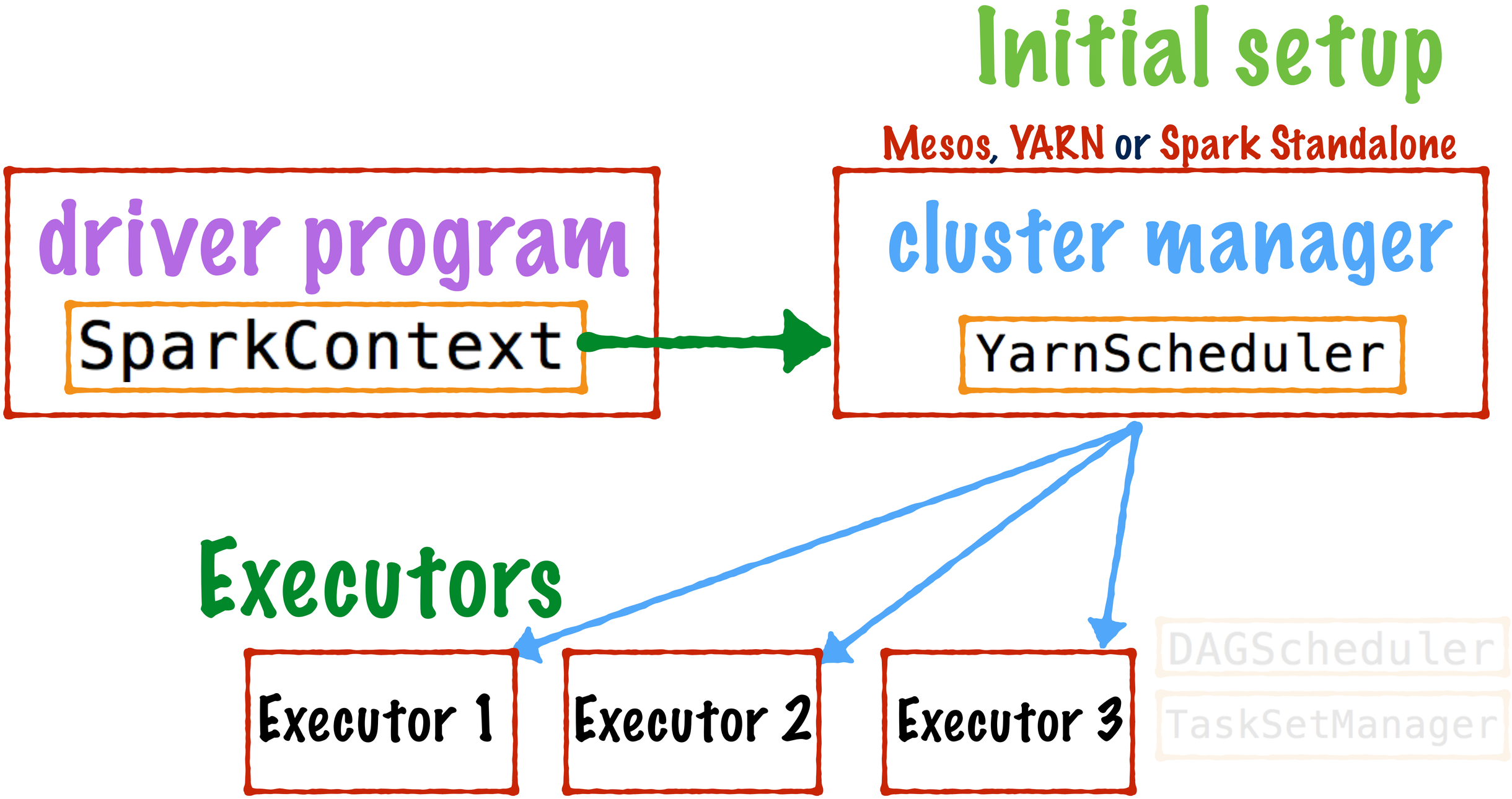
Executors

Java processes

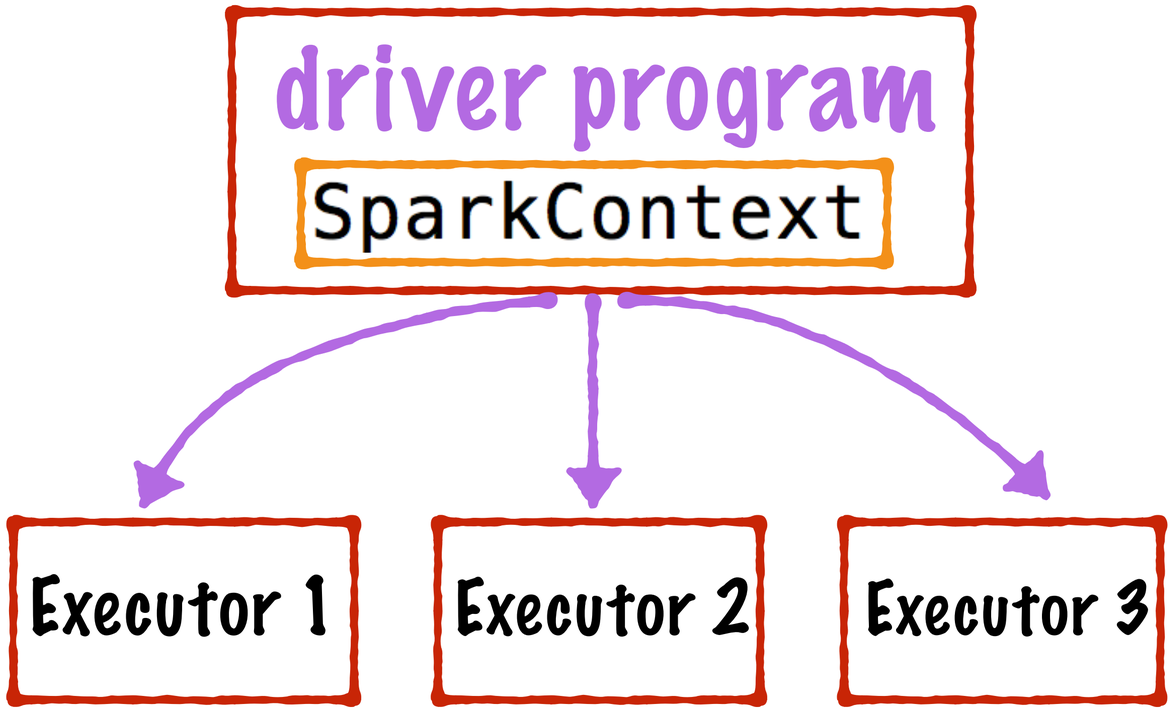
The cluster manager in turn launches on several nodes in the cluster

DAGScheduler

TaskSetManager


Initial setup
Mesos, YARN or Spark Standalone
driver program
SparkContext
cluster manager
YarnScheduler
Executors
Executor 1
Executor 2
Executor 3
DAGScheduler
TaskSetManager

Initial setup

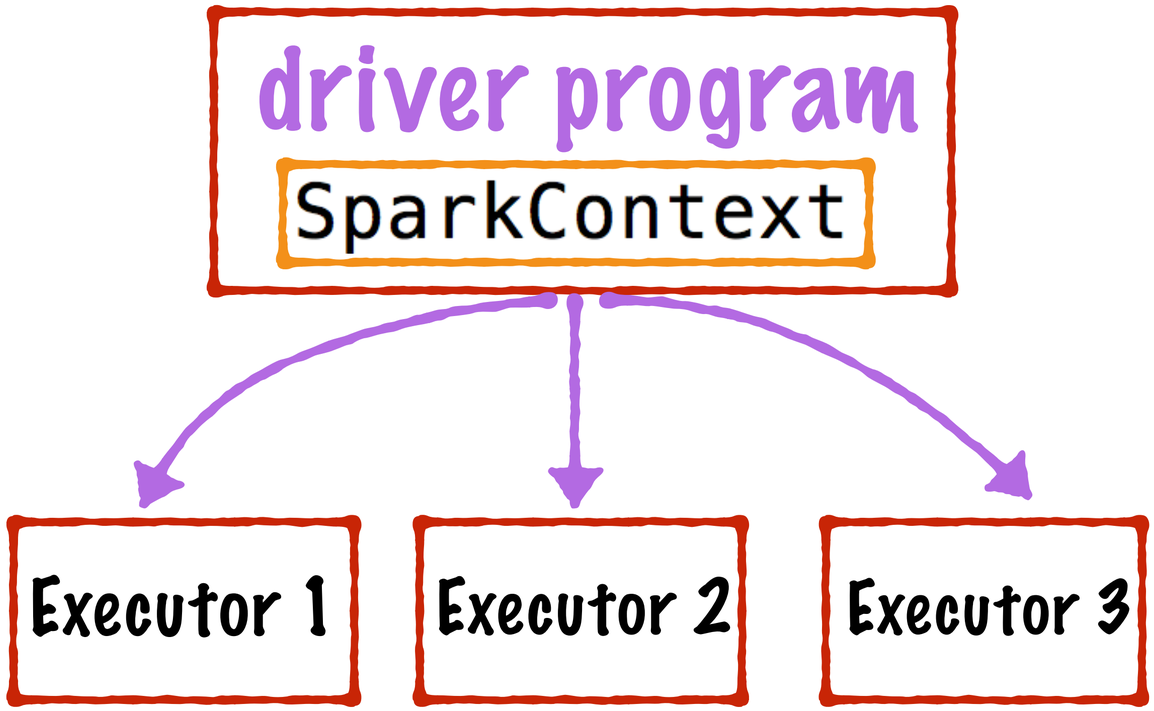


DAGScheduler

TaskSetManager

Once the executors are launched they register themselves with the driver program

Initial setup



DAGScheduler

TaskSetManager

The driver program is now ready for processing user instructions

driver program

SparkContext

Initial setup ✓

Job Run

DAGScheduler

TaskSetManager

Job Run

Whenever there is an action
(or some processing) on an RDD

Executor 1

Executor 2

Executor 3

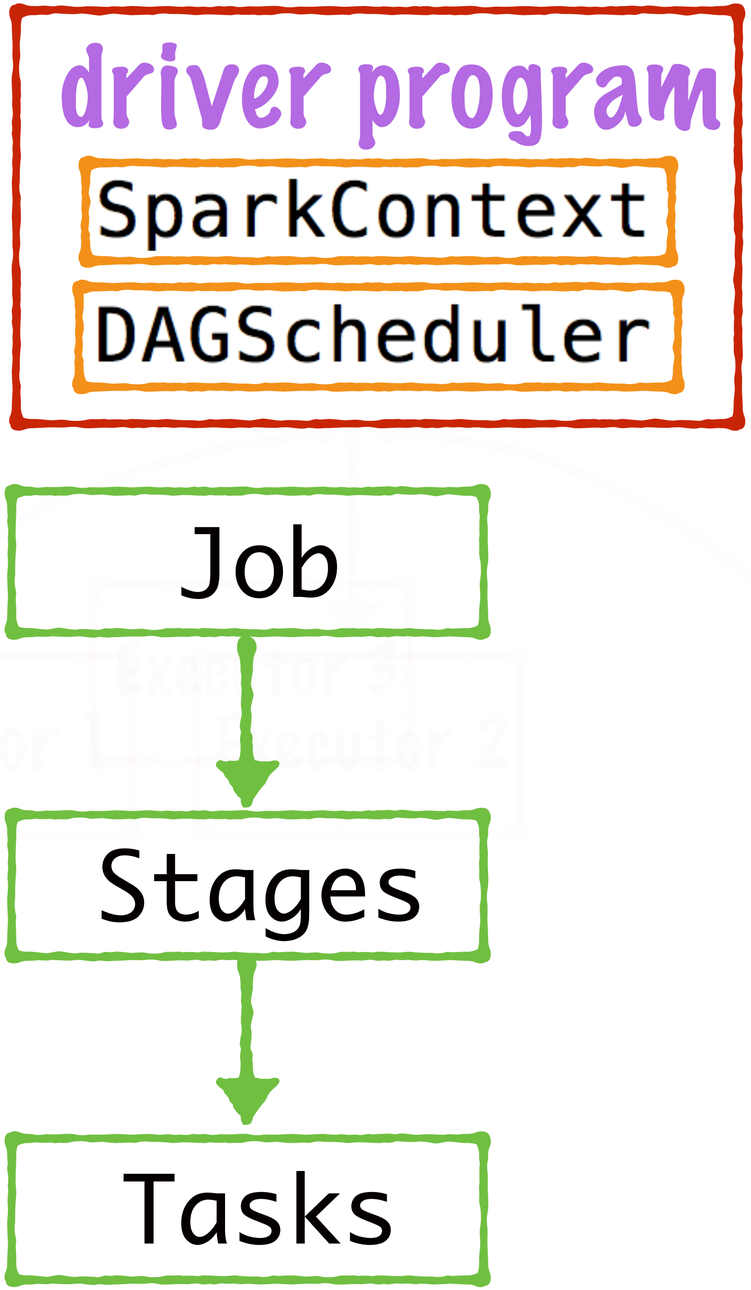
A Job is initiated

When this happens,
SparkContext passes the user
instructions on to a Scheduler

DAGScheduler

TaskSetManager

Job Run



The scheduler breaks down the Job into smaller units of work

Job Run

DAGScheduler

Job

Stages

Tasks

```
flightsParsed.map(lambda x:x.distance).reduce(lambda x,y:x+y)
```

Stage 0

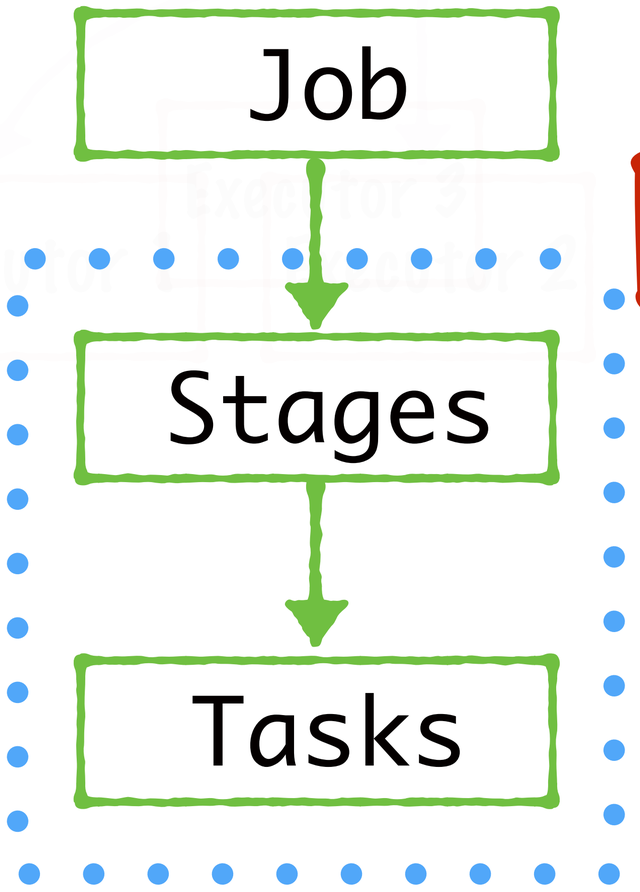
```
map(lambda x:x.distance)
```

Stage 1

```
reduce(lambda x,y:x+y)
```

Tasks on individual partitions of the RDD

Job Run



DAG

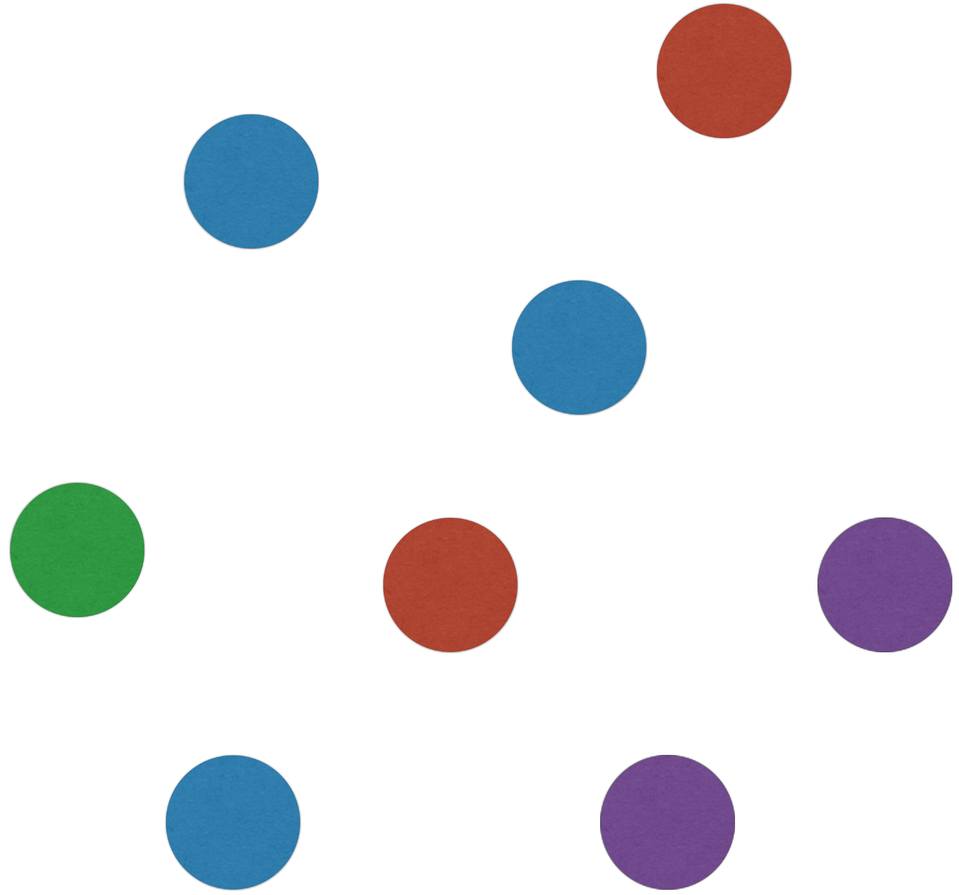
The Stages and Tasks form
a Directed Acyclic Graph

This is a standard way
to represent a workflow

Directed Acyclic Graph

In any workflow

You might have a bunch of tasks

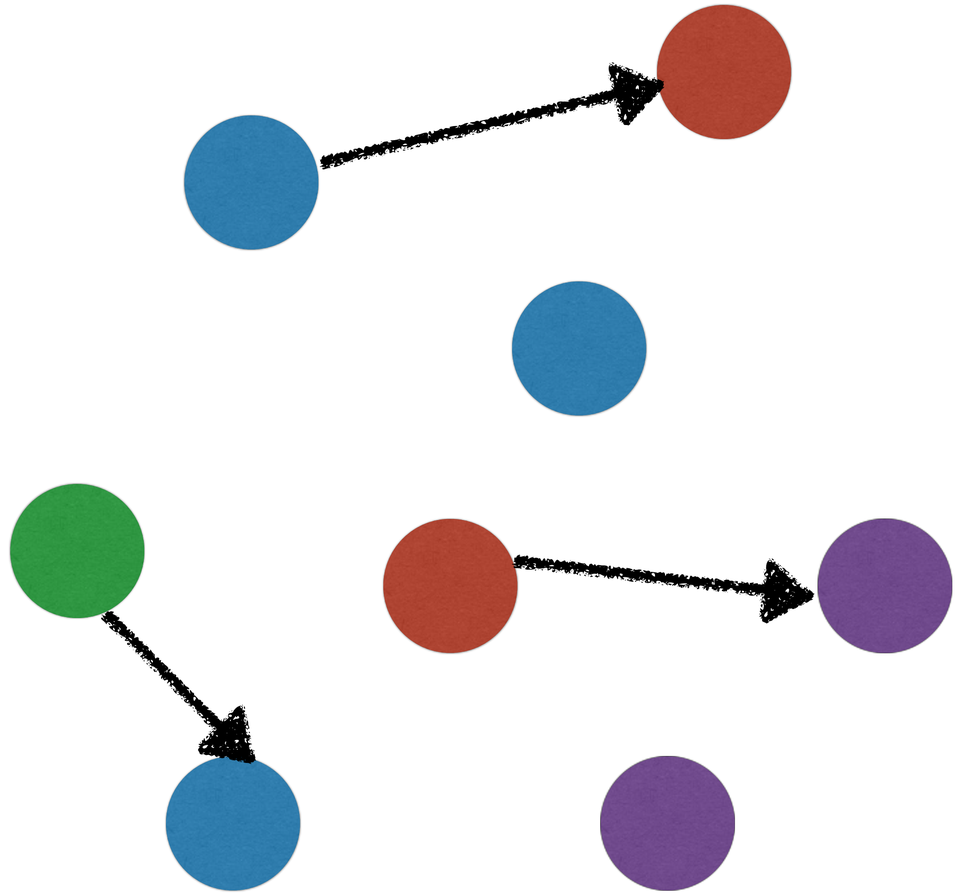


Directed Acyclic Graph

Some tasks are independent

They can be done in parallel

Ex: Applying a map operation on individual partitions of an RDD



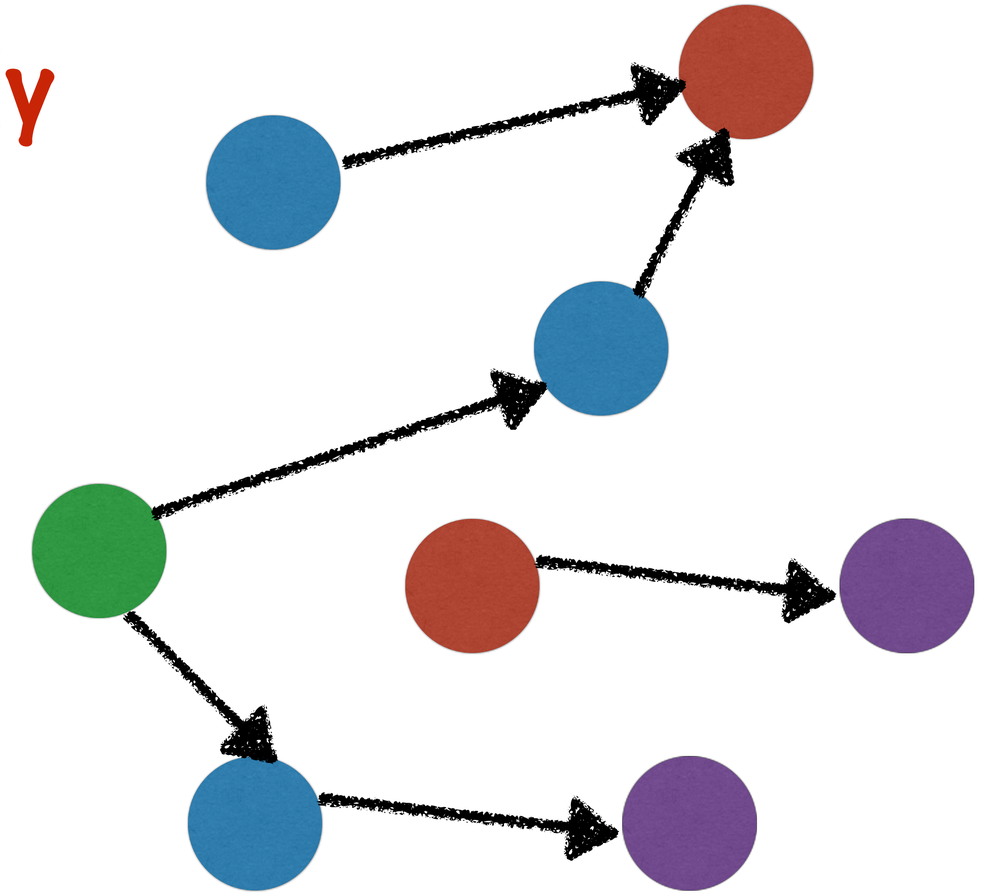
Directed Acyclic Graph

Some tasks are
dependent on each other

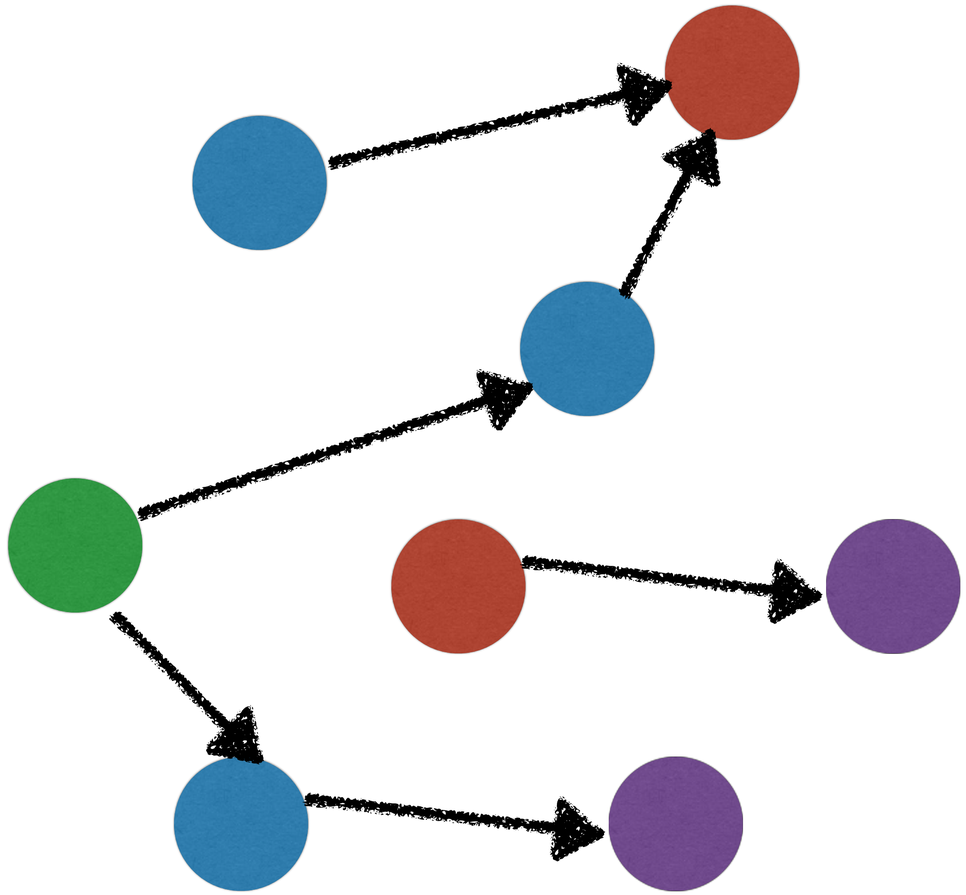
They have to be performed serially

Ex: In reduce,

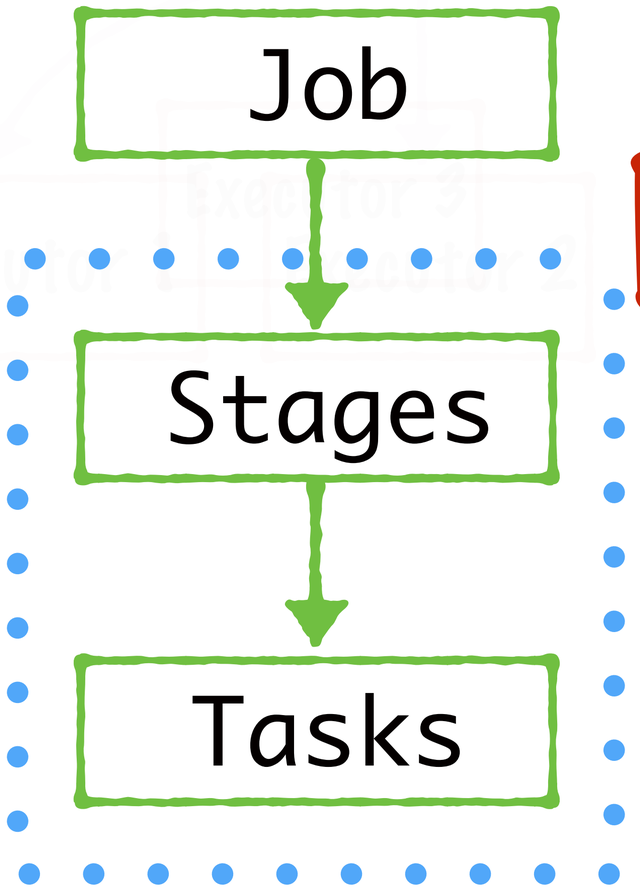
1. individual partitions have to be processed
2. Those results are combined



Directed Acyclic Graph



Job Run



DAG

The Stages and Tasks form
a Directed Acyclic Graph

This is a standard way
to represent a workflow

driver program

SparkContext

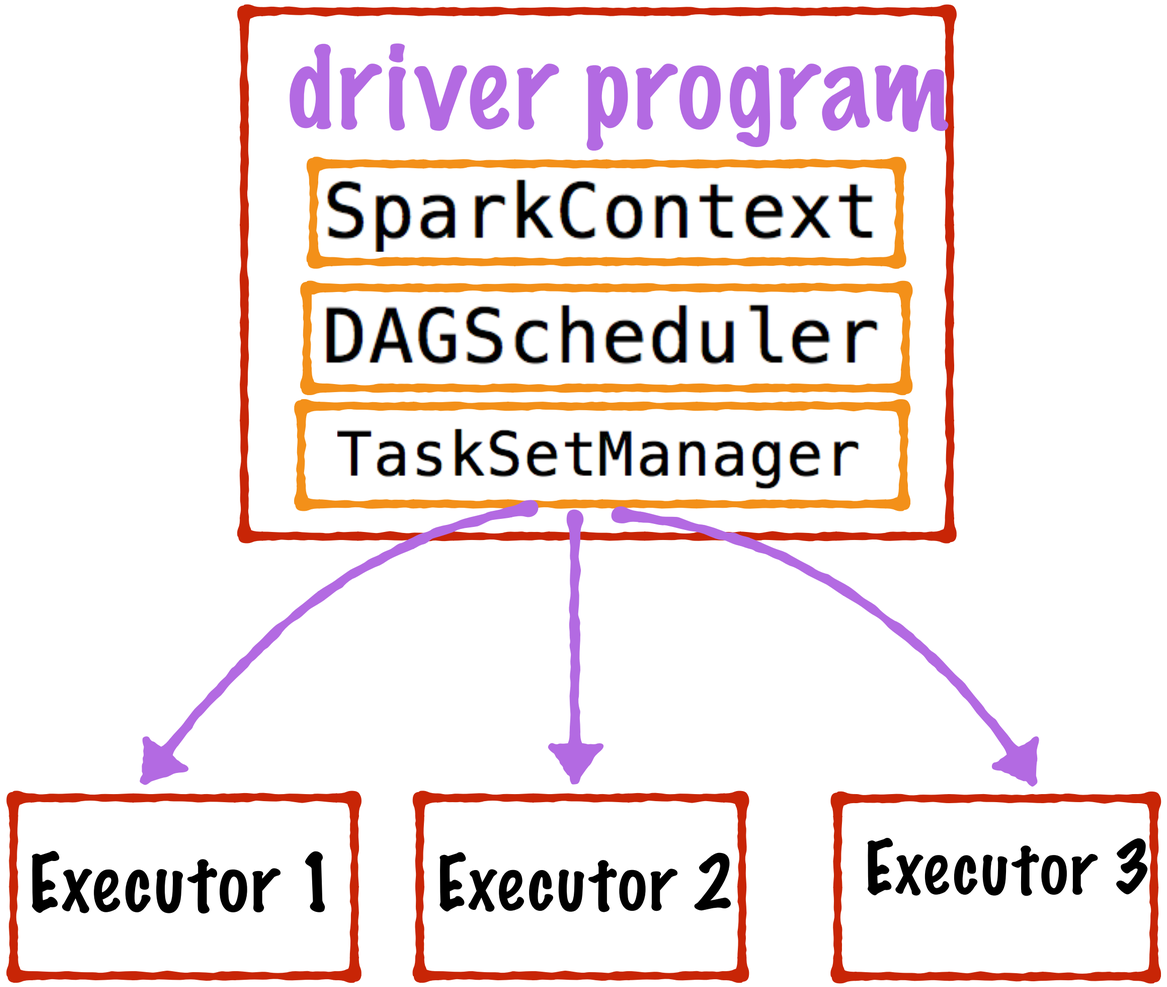
DAGScheduler

Job Run

Tasks

Tasks are the smallest unit of work in a Spark Job

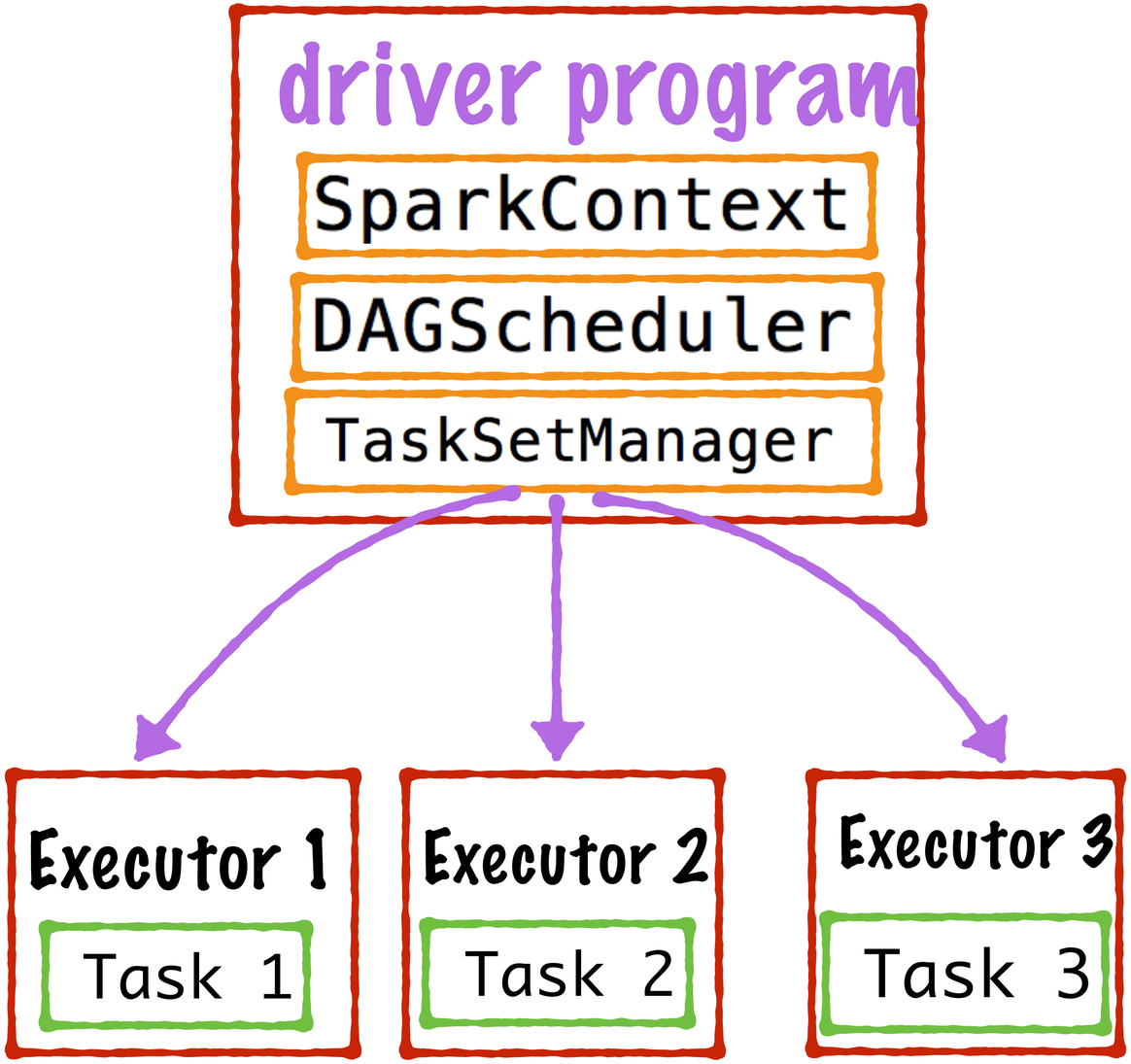
Each task is assigned to an Executor by the Task scheduler



Job Run

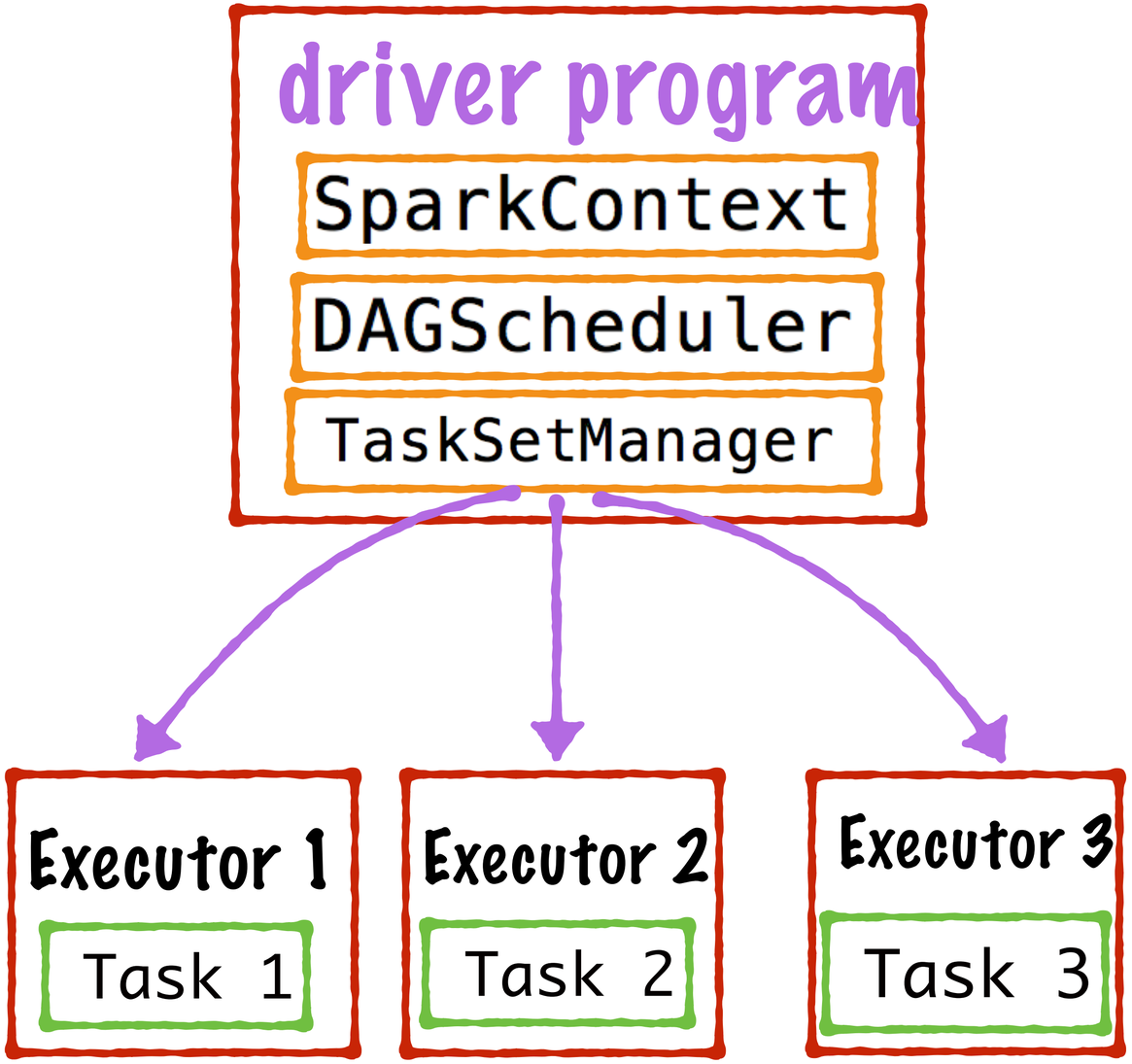
Tasks

Each task is assigned to an Executor by the Task scheduler



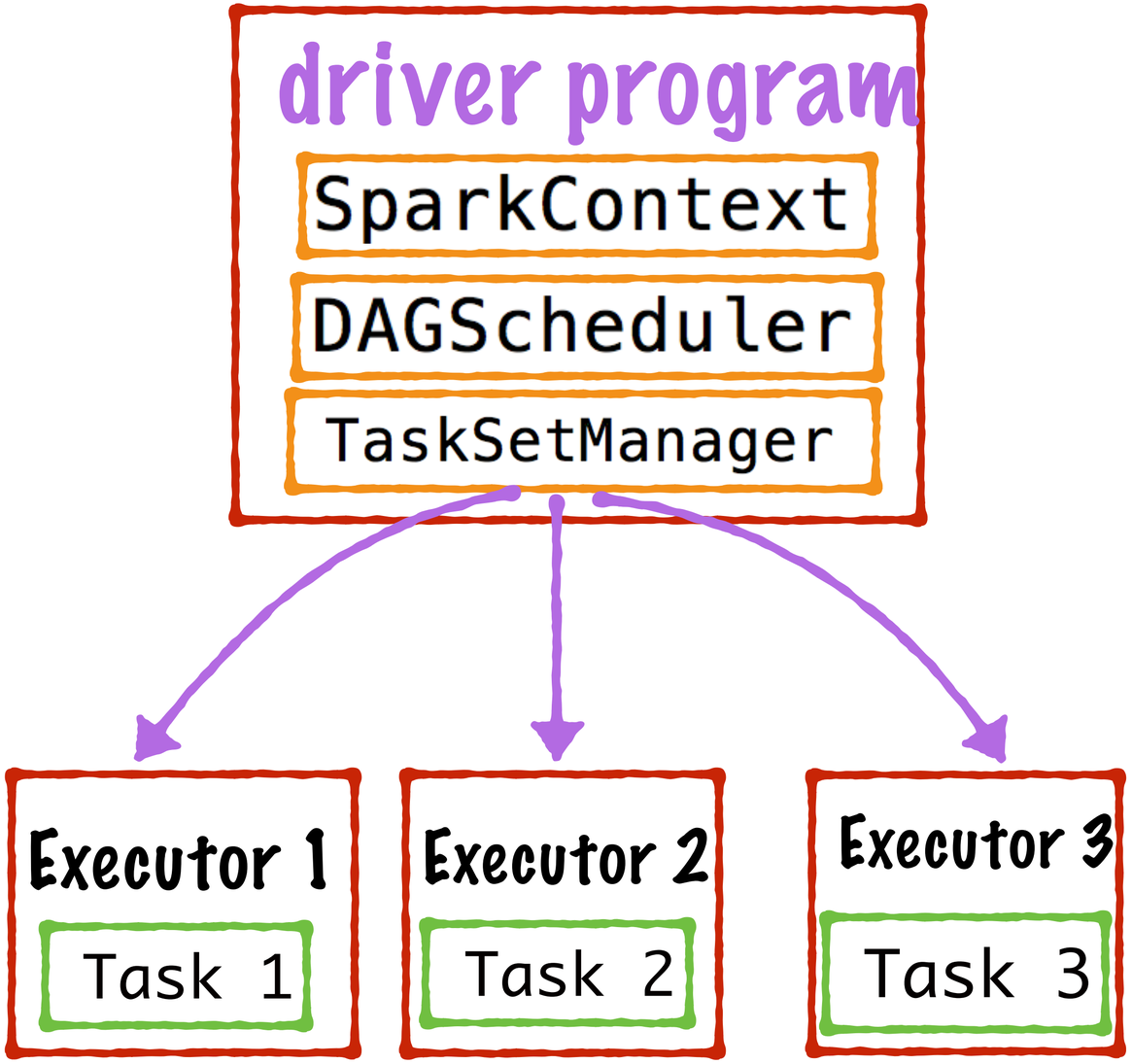
Job Run

The executors run the tasks and send updates back to the driver program



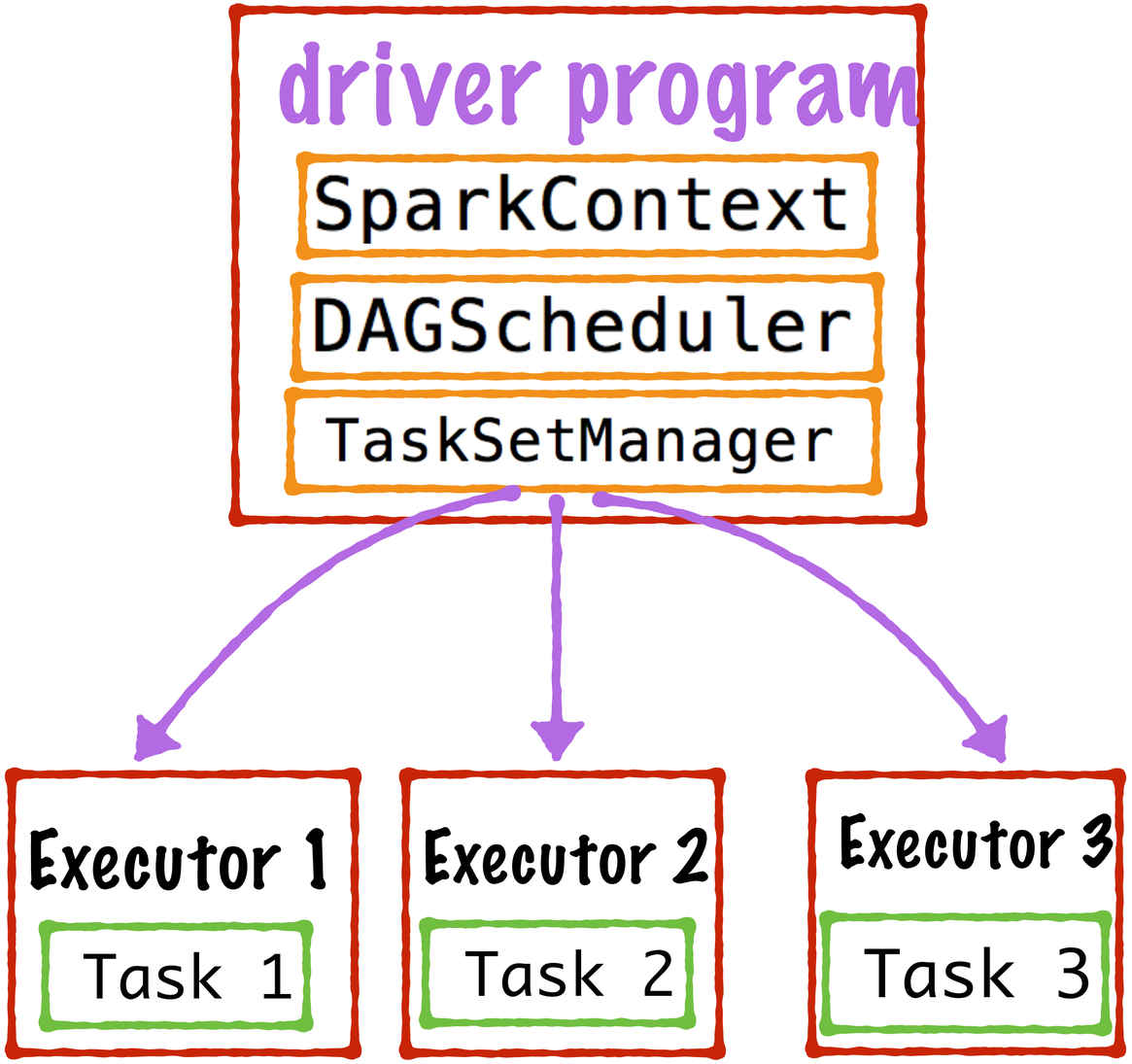
Job Run

When all the tasks in a Stage are completed, the next Stage is launched



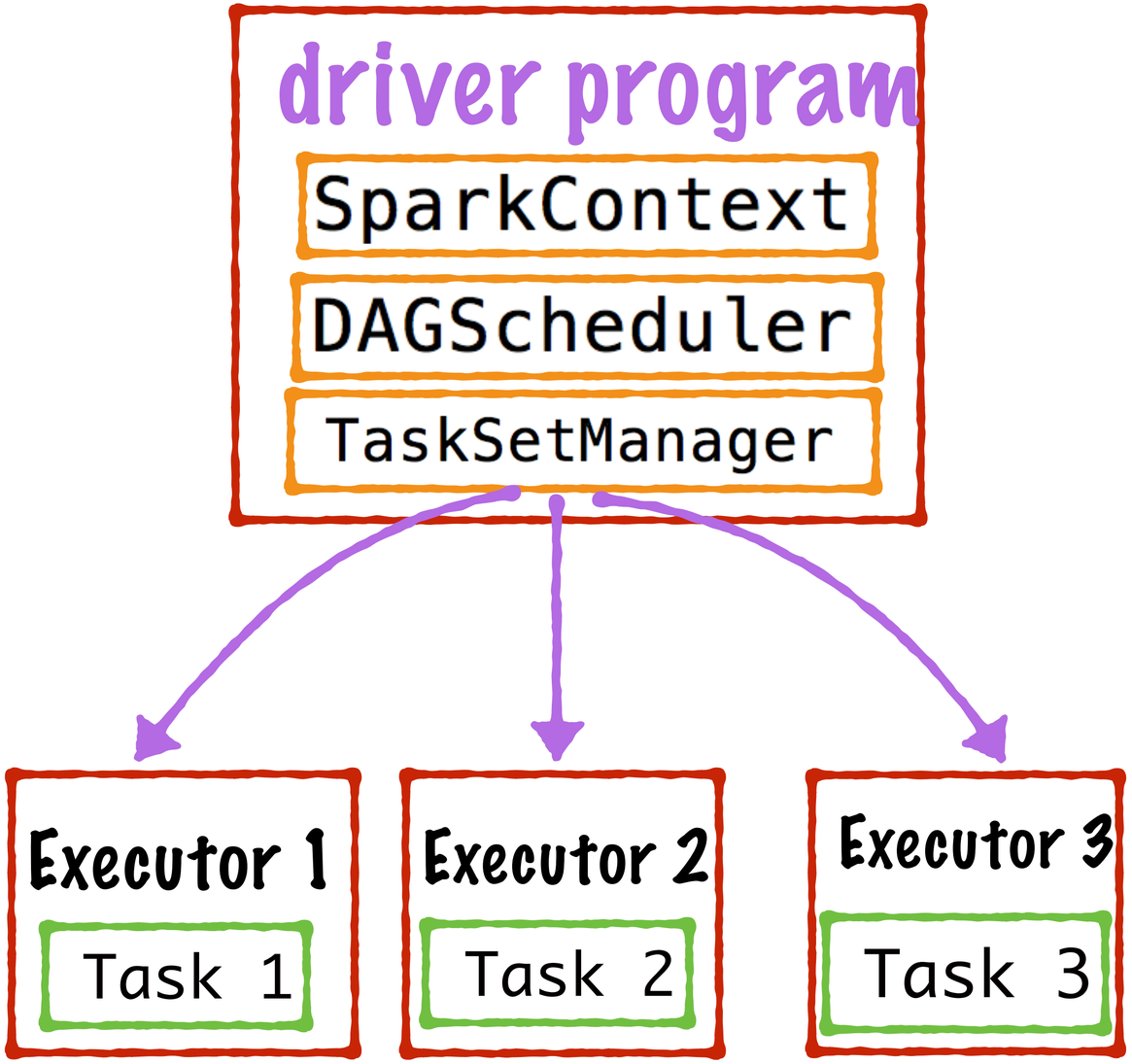
Job Run

When all the Stages are completed, the Job finishes



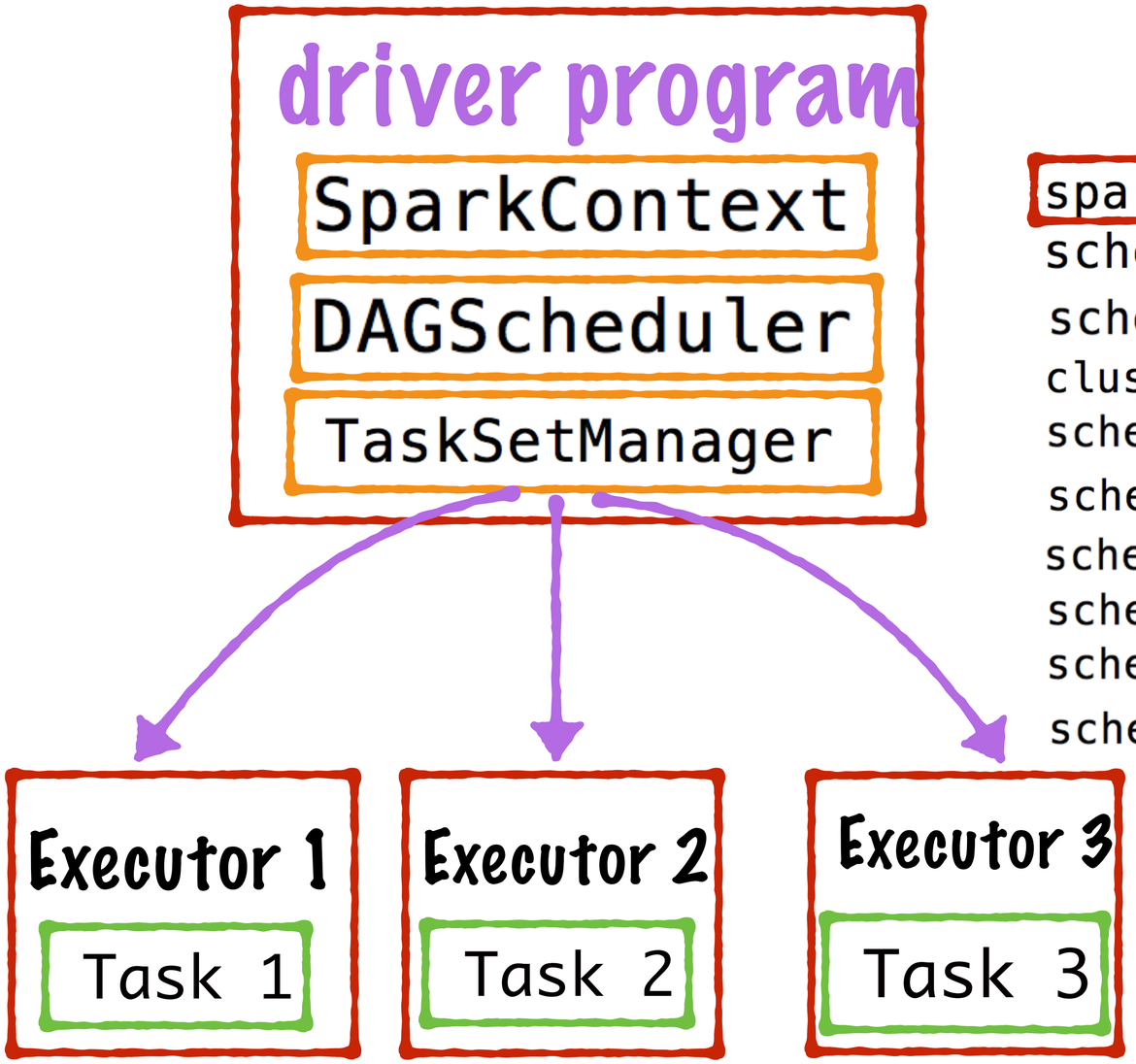
Job Run

Let's go back and look at the status messages Spark prints



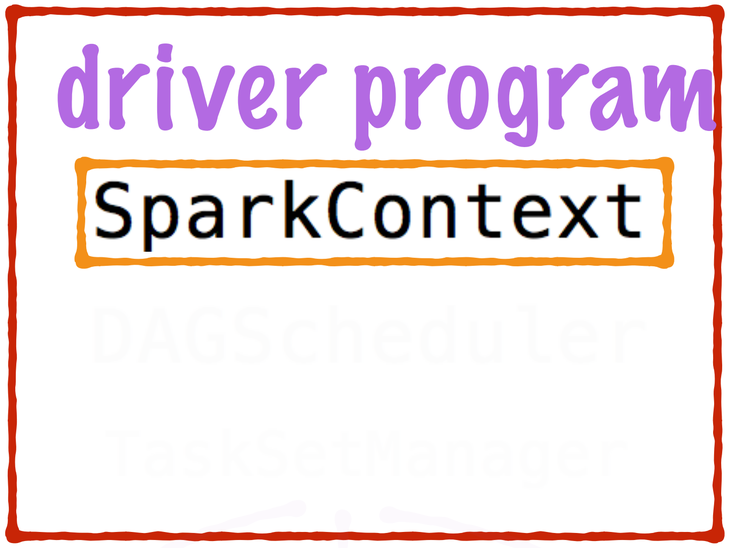
Job Run

Let's go back and look at the status messages Spark prints



Job Run

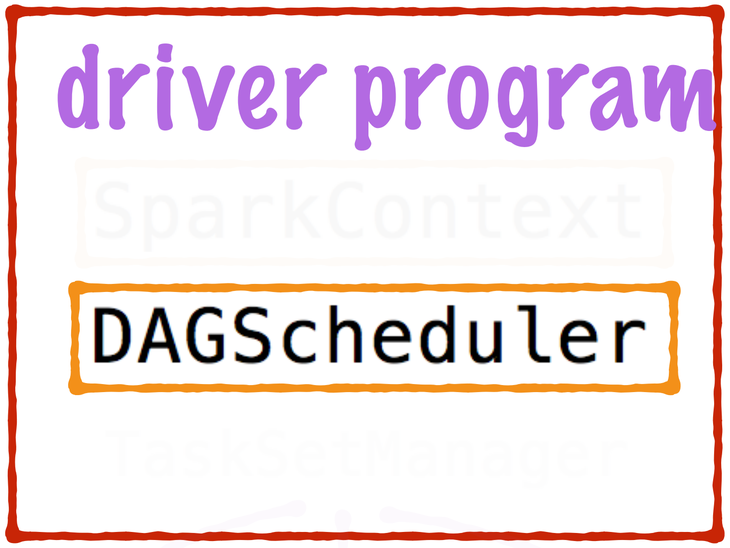
```
spark.SparkContext: Starting job: saveAsTextFile
scheduler.DAGScheduler: Got job 0 (saveAsTextFil
scheduler.DAGScheduler: Submitting ResultStage 0 (
cluster.YarnScheduler: Adding task set 0.0 with 2 tasks
scheduler.TaskSetManager: Starting task 0.0 in stage 0.
scheduler.TaskSetManager: Starting task 1.0 in stage 0.
scheduler.TaskSetManager: Finished task 1.0 in stage 0.
scheduler.TaskSetManager: Finished task 0.0 in stage 0.
scheduler.DAGScheduler: ResultStage 0 (saveAsTextFile a
scheduler.DAGScheduler: Job 0 finished: saveAsTextFile
```



Job Run

```
spark.SparkContext: Starting job: saveAsTextFile
scheduler.DAGScheduler: Got job 0 (saveAsTextFil
scheduler.DAGScheduler: Submitting ResultStage 0 (
cluster.YarnScheduler: Adding task set 0.0 with 2 tasks
scheduler.TaskSetManager: Starting task 0.0 in stage 0.
scheduler.TaskSetManager: Starting task 1.0 in stage 0.
scheduler.TaskSetManager: Finished task 1.0 in stage 0.
scheduler.TaskSetManager: Finished task 0.0 in stage 0.
scheduler.DAGScheduler: ResultStage 0 (saveAsTextFile a
scheduler.DAGScheduler: Job 0 finished: saveAsTextFile
```

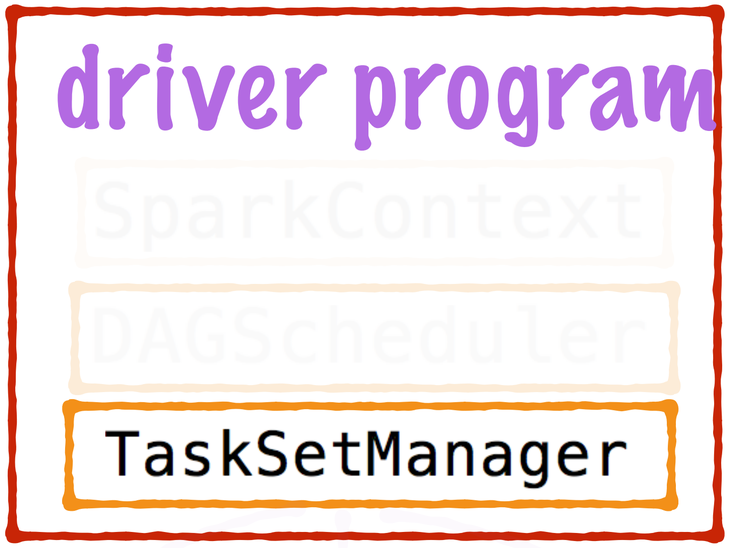
SparkContext starts the Job



Job Run

```
spark.SparkContext: Starting job: saveAsTextFile
scheduler.DAGScheduler: Got job 0 (saveAsTextFil
scheduler.DAGScheduler: Submitting ResultStage 0 (
cluster.YarnScheduler: Adding task set 0.0 with 2 tasks
scheduler.TaskSetManager: Starting task 0.0 in stage 0.
scheduler.TaskSetManager: Starting task 1.0 in stage 0.
scheduler.TaskSetManager: Finished task 1.0 in stage 0.
scheduler.TaskSetManager: Finished task 0.0 in stage 0.
scheduler.DAGScheduler: ResultStage 0 (saveAsTextFile a
scheduler.DAGScheduler: Job 0 finished: saveAsTextFile
```

DAGScheduler takes the Job and constructs a DAG of Stages and tasks



Job Run

```
spark.SparkContext: Starting job: saveAsTextFile
scheduler.DAGScheduler: Got job 0 (saveAsTextFil
scheduler.DAGScheduler: Submitting ResultStage 0 (
cluster.YarnScheduler: Adding task set 0.0 with 2 tasks
scheduler.TaskSetManager: Starting task 0.0 in stage 0.
scheduler.TaskSetManager: Starting task 1.0 in stage 0.
scheduler.TaskSetManager: Finished task 1.0 in stage 0.
scheduler.TaskSetManager: Finished task 0.0 in stage 0.
scheduler.DAGScheduler: ResultStage 0 (saveAsTextFile a
scheduler.DAGScheduler: Job 0 finished: saveAsTextFile
```

TaskSetManager actually schedules and keeps track of the tasks