APACHE SPARK EXECUTION MODEL

By www.HadoopExam.com

Note: These instructions should be used with the HadoopExam Apache Spark: Professional Trainings. Where it is executed and you can do hands on with trainer.

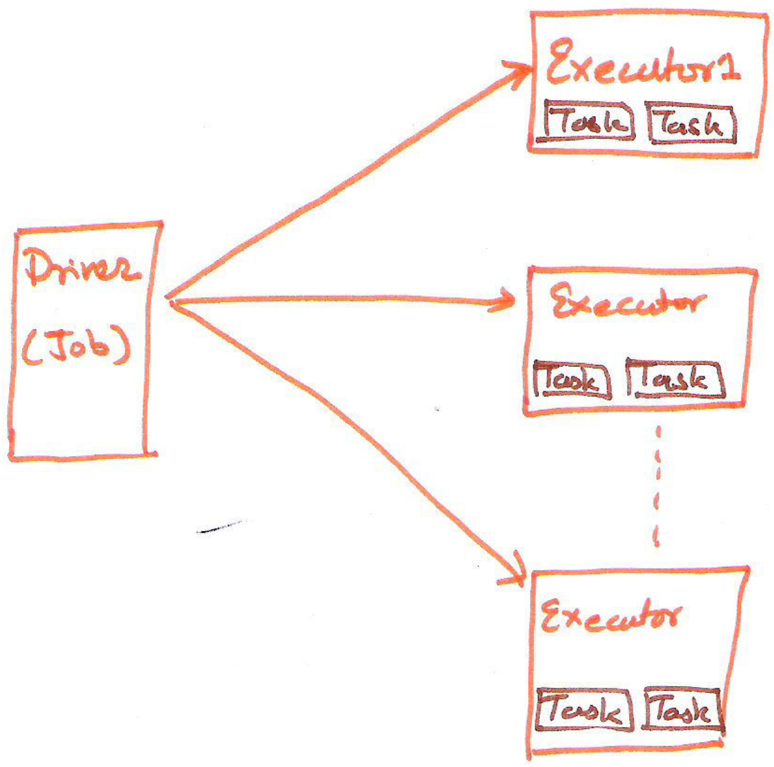
Spark Execution Model

- How spark execute your programs.

- A spark Application consist of
  - Single Driver Process
  - Set of executor process [Scattered across nodes]

- Driver:- Control high level flow of work that needs to be done.

- Executor:- Executing the control flow, in the form of tasks
  - As well as storing the data, that the user choose to cache.



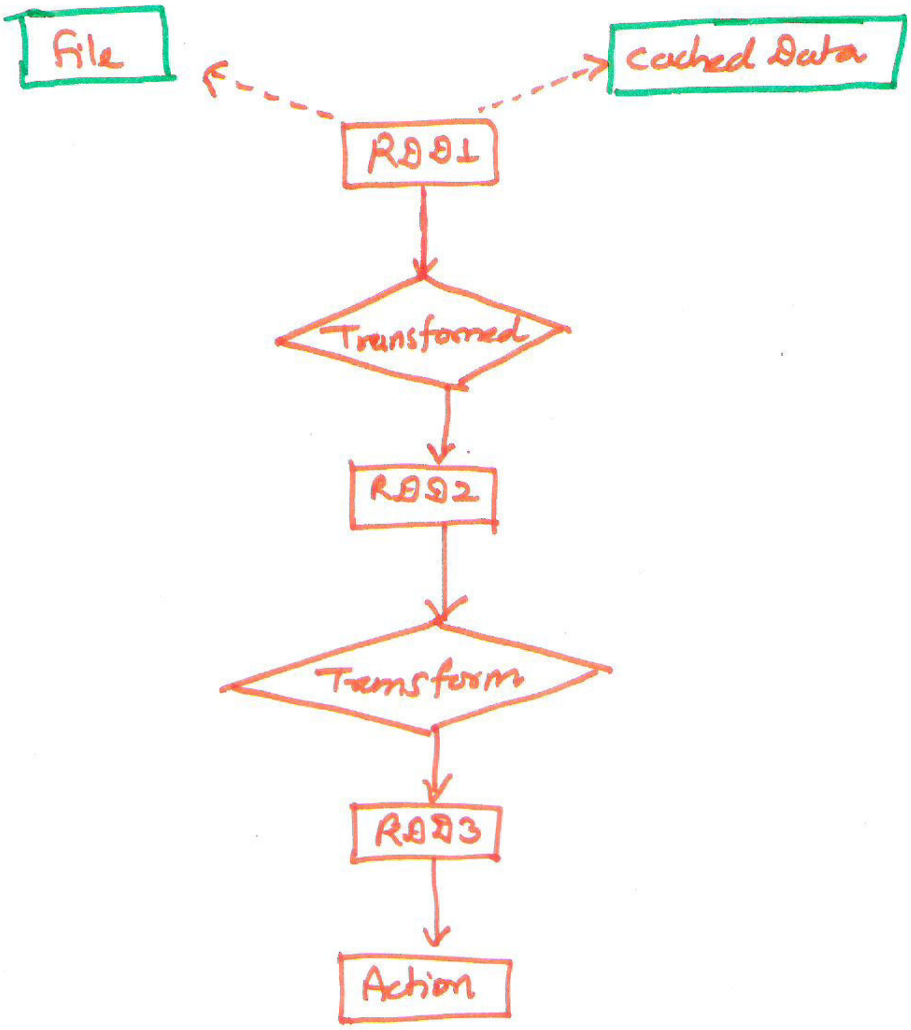
A Job contains tasks, and execute Concurrently(

- A single executor has a number of slots for running tasks, and will run many concurrently.

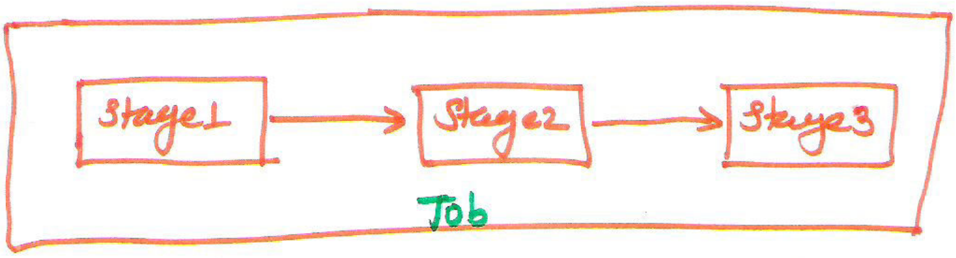
- At the top of the execution hierachy are jobs.
- Spark looks the graph of RDDs on which that action depends and formulate an execution plan.

- This plan starts with the farthest-back RDDs - that is, those that depends on no other RDDs.

- OR refrence already cached-data - and culminate in the final RDD required to produce the actions results.



- The execution plan consists of assembling the Job's transformations into stages.

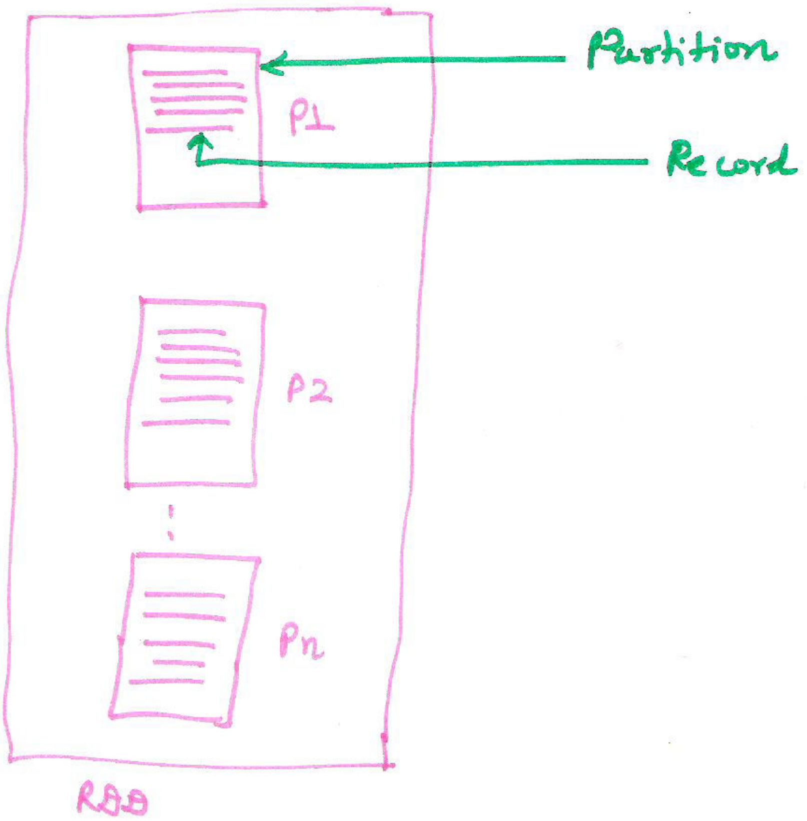


- A stage is corresponds to a collection of tasks that all ~~execution~~ execute the same code, each on a different subset of data.

- Each stage contains a sequence of transformations that can be completed without shuffling the full data.

⇒ How & what determines whether data needs to be shuffled?

- RDD comprises a fixed number of partitions.

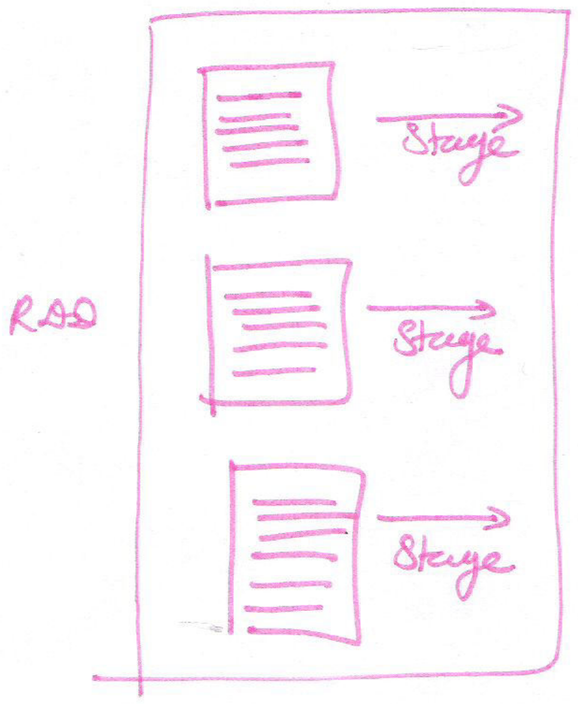


- Narrow transformation
  - map
  - filter
- Work on partition, independently, no need data shuffling
- Spark also supports transformations with wide dependencies such as
  - groupByKey()
  - reduceByKey()

- This operation bring all the same key records from all the partition together to work upon.
- Shuffle transfer data around the cluster and results in a new stage with a new set of partitions.

```
sc.textfile("hadoopExam.log")
  .map(mapfunc)
  .flatMap(...)
  .filter(...)
  .count()
```

(Shuffle not Required)



. Stage runs Parallelly on each partition in a RDD.

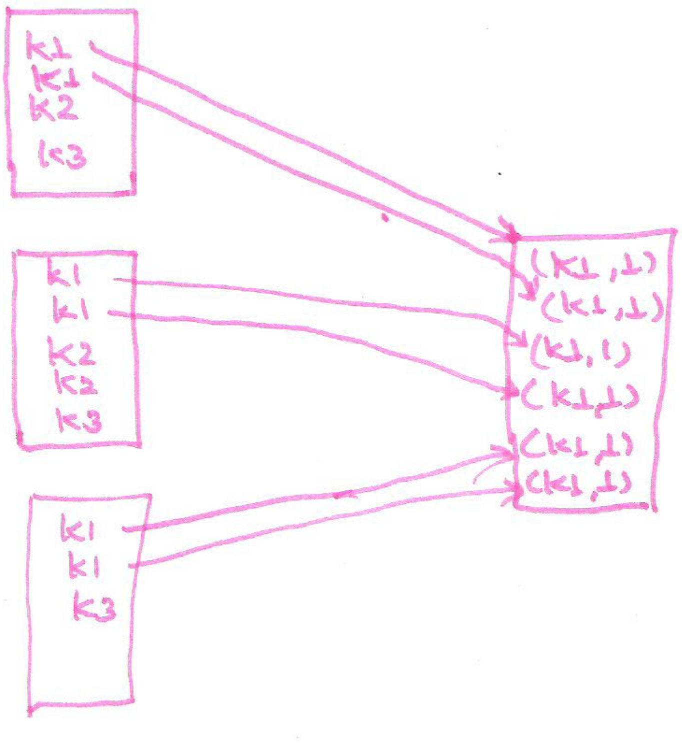
- It executes a single action, which depends on a sequence of transformations on an RDD derived from a text file.
- It would executes stage concurrently on ~~each~~ all partitions.

⇒ Because none of the output of these three operations depend on data that can come from different partitions than their input.

⇒ When shuffling Happen?

```
Val token = sc.textfile("hdfs//:hadoopExam.log")
            token.flatMap(_.split(' ')
                 .map(_,1)
                 .reduceBykey(_+_) //Word count
                 .flatMap(_._1).tocharArray
                 .map(_,1)
                 .reduceBykey(_+_) //char count
                 .collects()
```

⇒ This process will run in more than one stages.



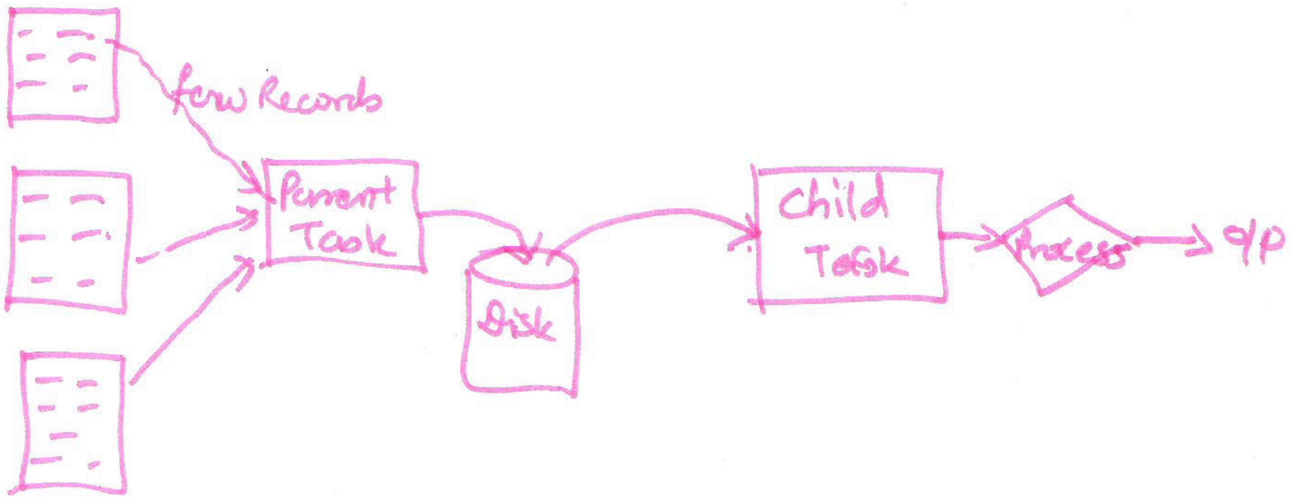
reduceBykey() require to be shuffled across partitions for getting same key

(k1,6) → stage1 (Word count)

→ stage2 (character count)

⇒ At each stage boundary data is written to disk by task in parent stage.

⇒ Then fetched over the n/w by tasks in the child stage.



⇒ Because shuffle incur heavy disk and n/w I/O, stage boundaries can be expensive and should be avoided when possible.

⇒ The number of data partitions in parent stage may ~~be~~ be different than the number of partitions in the child stage.

⇒ Transformations that may trigger a stage boundary typically accept "numPartitions" argument that determine how many partitions to split ~~this~~ the data into child stage.