Big Data And Hadoop

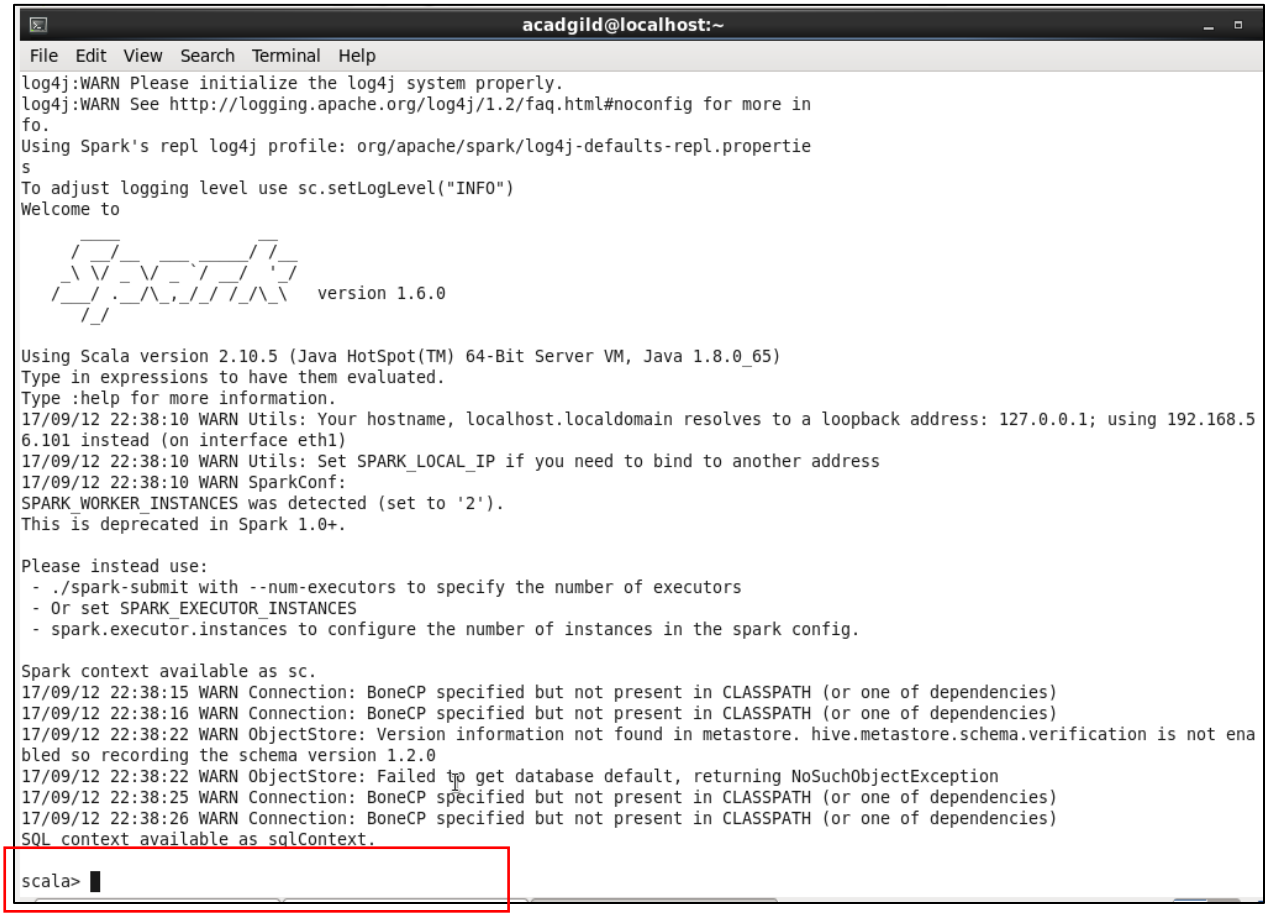
Session 13 – Assignment 2

Problem Statement:

a. Create a simple pairRDD of (1, 2), (3, 4), (3, 6).
b. Transform an RDD of ("a","b","c","d","e") to PairRDD (a,0), (b,1), (c,2), (d,3), (e,4)

Solution:

We begin with sparting the spark shell using the command **spark-shell.** The spark shell look as follows:



a. To create pairRDD of (1, 2), (3, 4), (3, 6).

Answer: Spark provides special type of operations on RDDs containing key or value pairs. These RDDs are called pair RDDs operations. Pair RDDs are a useful building block in many programming language, as they expose operations that allow you to act on each key operations in parallel or regroup data across the network.

We can create pair RDD as follows:

We create an RDD representing this data using parallelize(). Parallelized collections are created by calling SparkContext's parallelize method on an existing collection in your driver program (a Scala Seq). The elements of the collection are copied to form a distributed dataset that can be operated on in parallel.

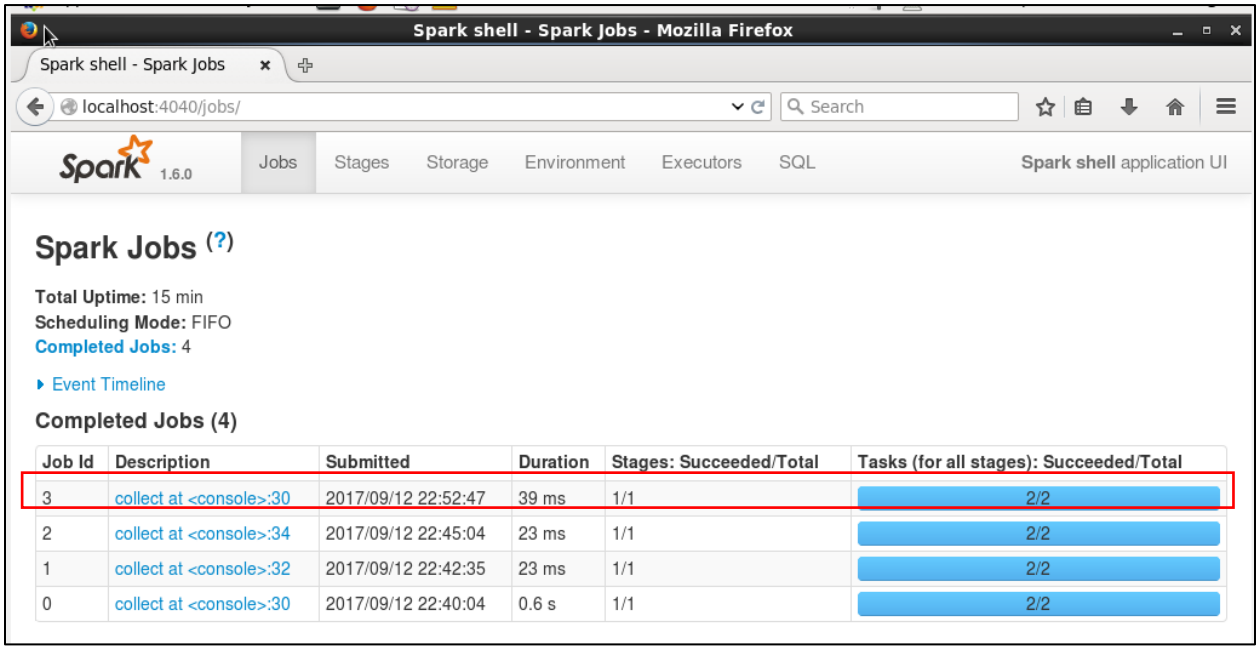
val data = sc.parallelize(Array((1, 2), (3, 4), (3, 6)))

It can be seen in the screenshot below:

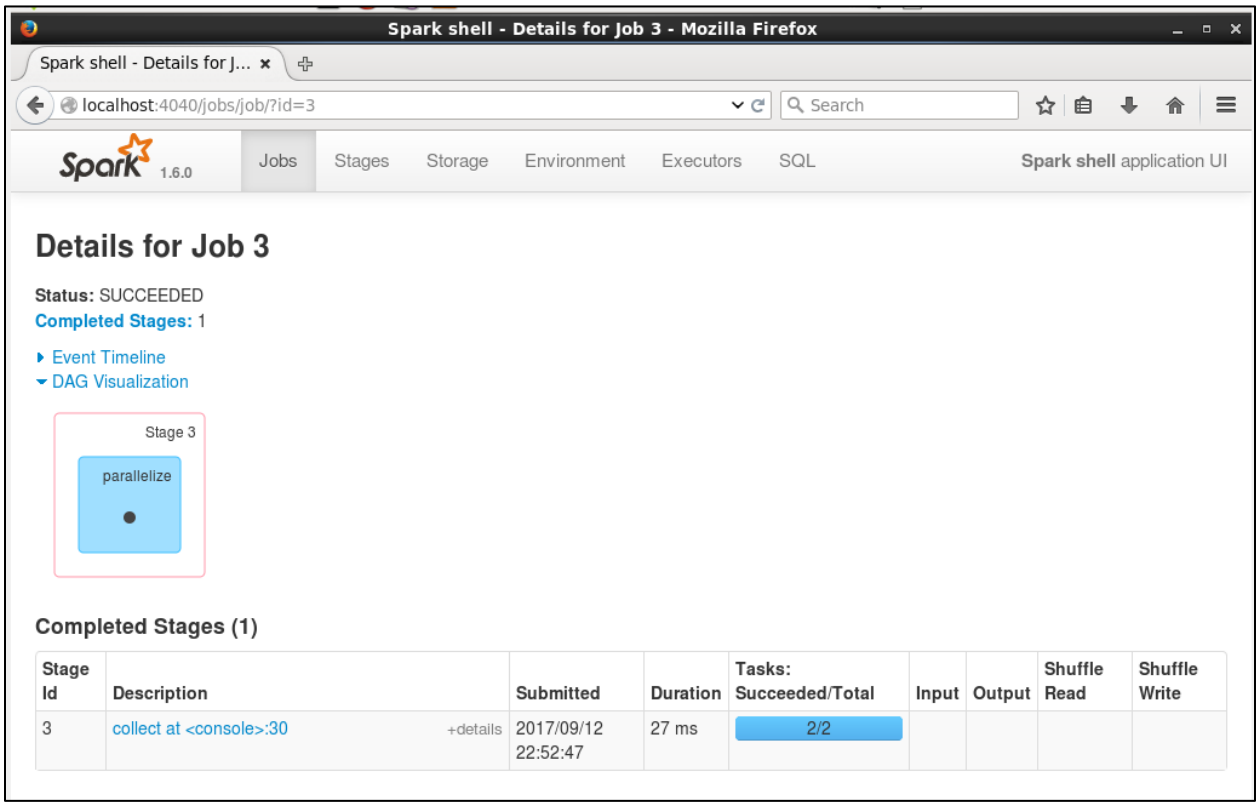
```
acadgild@localhost:~
File  Edit  View  Search  Terminal  Help
17/09/12 22:38:15 WARN Connection: BoneCP specified but not present in CLASSPATH (or one of dependencies)
17/09/12 22:38:16 WARN Connection: BoneCP specified but not present in CLASSPATH (or one of dependencies)
17/09/12 22:38:22 WARN ObjectStore: Version information not found in metastore. hive.metastore.schema.verification is not ena
bled so recording the schema version 1.2.0
17/09/12 22:38:22 WARN ObjectStore: Failed to get database default, returning NoSuchObjectException
17/09/12 22:38:25 WARN Connection: BoneCP specified but not present in CLASSPATH (or one of dependencies)
17/09/12 22:38:26 WARN Connection: BoneCP specified but not present in CLASSPATH (or one of dependencies)
SQL context available as sqlContext.

scala> val data = sc.parallelize(List(1,2,3,4,5,6,7,8,9,10));
data: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at <console>:27

scala> data.collect
res0: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

scala> val cubes = data.map(x=>x*x*x)
cubes: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[1] at map at <console>:29

scala> cubes.collect
res1: Array[Int] = Array(1, 8, 27, 64, 125, 216, 343, 512, 729, 1000)

scala> val filteredData = cubes.filter(x=>(x%2==0  && x%3==0))
filteredData: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[2] at filter at <console>:31

scala> filteredData.collect
res2: Array[Int] = Array(216)

scala> val data = sc.parallelize(Array((1,2),(3,4),(3,6));
<console>:1: error: ')' expected but ';' found.
       val data = sc.parallelize(Array((1,2),(3,4),(3,6));
                                                         ^

scala> val data = sc.parallelize(Array((1,2),(3,4),(3,6)));
data: org.apache.spark.rdd.RDD[(Int, Int)] = ParallelCollectionRDD[3] at parallelize at <console>:27

scala> data.collect
res3: Array[(Int, Int)] = Array((1,2), (3,4), (3,6))

scala>
```

On performing **collect** action, we get the output as shown in the above snapshot. Here, the RDD got created with the specified data.

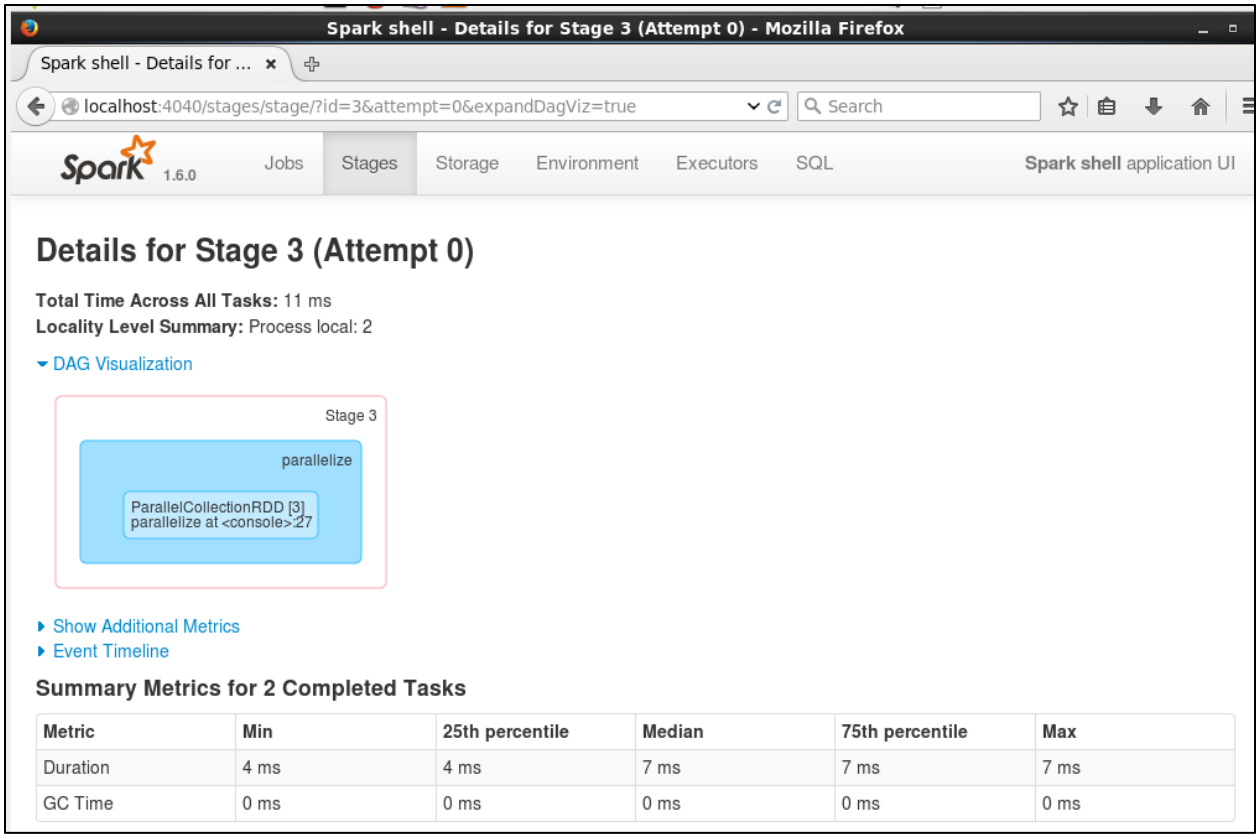
Collect being an action, its execution appears in the Spark UI as a job as follows:



The DAG representation of this job is as follows:



Detailed DAG representation is :



b. Transform an RDD of ("a","b","c","d","e") to PairRDD (a,0), (b,1), (c,2), (d,3), (e,4)
Answer: Here, the steps used are as follows:

Step1: Create RDD data using **parallelize** transformation as follows:

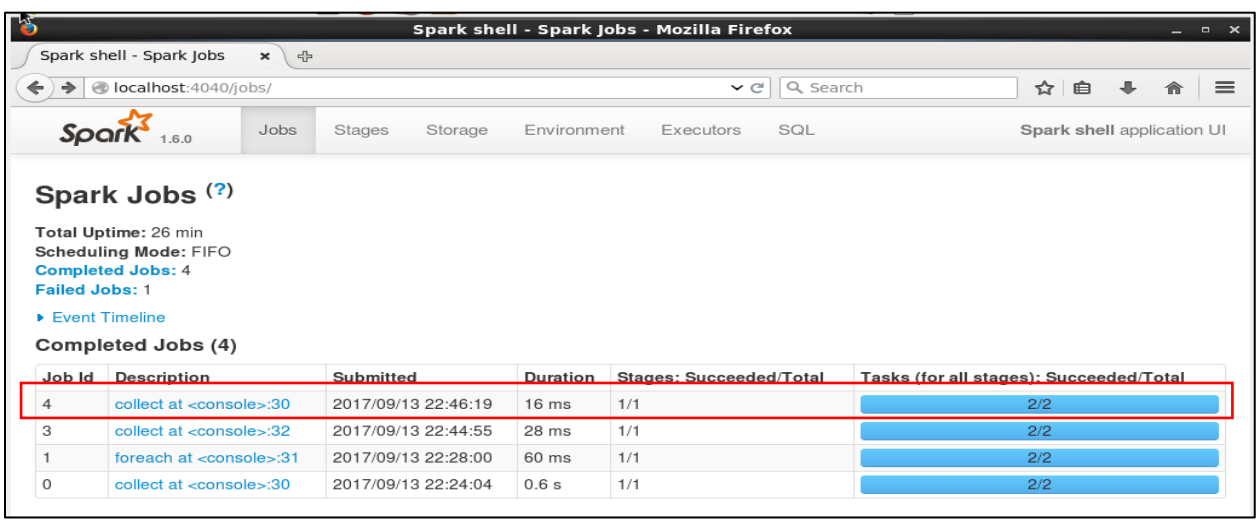
val data = sc.parallelize(Array("a","b","c","d","e"))

It can be seen in the screenshot below:

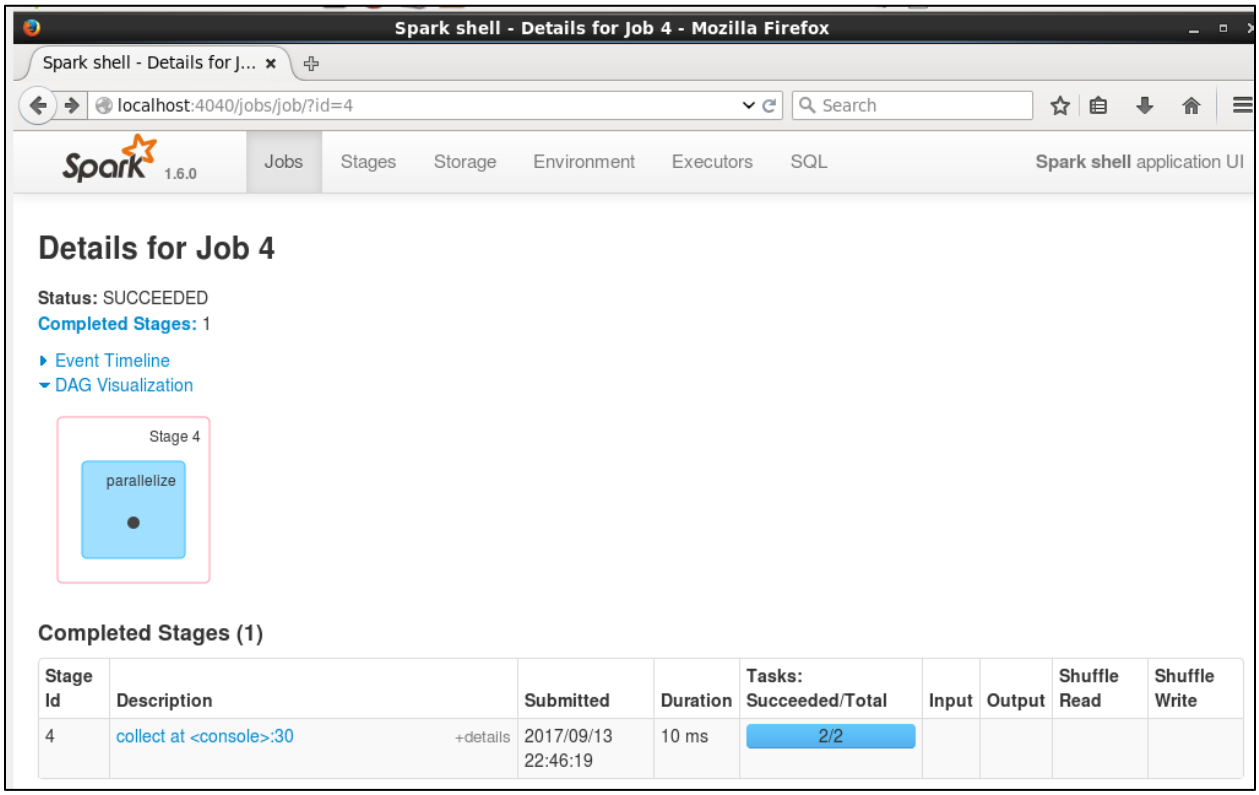
```
acadgild@localhost:~
File  Edit  View  Search  Terminal  Help
Type in expressions to have them evaluated.
Type :help for more information.
17/09/13 22:20:12 WARN Utils: Your hostname, localhost.localdomain resolves to a
 loopback address: 127.0.0.1; using 192.168.56.101 instead (on interface eth1)
17/09/13 22:20:12 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
17/09/13 22:20:12 WARN SparkConf:
SPARK_WORKER_INSTANCES was detected (set to '2').
This is deprecated in Spark 1.0+.

Please instead use:
 - ./spark-submit with --num-executors to specify the number of executors
 - Or set SPARK_EXECUTOR_INSTANCES
 - spark.executor.instances to configure the number of instances in the spark co
nfig.

Spark context available as sc.
17/09/13 22:20:19 WARN Connection: BoneCP specified but not present in CLASSPATH
 (or one of dependencies)
17/09/13 22:20:20 WARN Connection: BoneCP specified but not present in CLASSPATH
 (or one of dependencies)
17/09/13 22:20:26 WARN ObjectStore: Version information not found in metastore.
hive.metastore.schema.verification is not enabled so recording the schema versio
n 1.2.0
17/09/13 22:20:26 WARN ObjectStore: Failed to get database default, returning No
SuchObjectException
17/09/13 22:20:29 WARN Connection: BoneCP specified but not present in CLASSPATH
 (or one of dependencies)
17/09/13 22:20:30 WARN Connection: BoneCP specified but not present in CLASSPATH
 (or one of dependencies)
SQL context available as sqlContext.

scala> val data = sc.parallelize(Array("a","b","c","d","e"));
data: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[0] at parallelize at <console>:27

scala> data.collect
res0: Array[String] = Array(a, b, c, d, e)

scala>
```

On performing **collect** action, we get the output as shown in the above snapshot. Here, the RDD got created with the specified data.

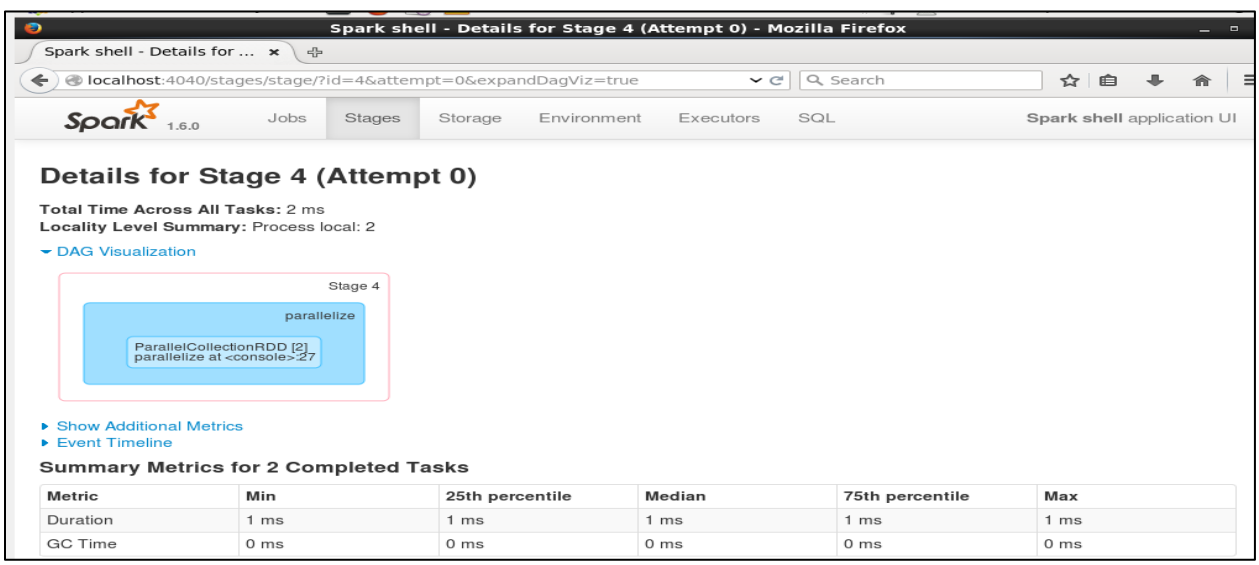
Collect being an action, its execution appears in the Spark UI as a job as follows:



DAG representation is as follows:



Detailed DAG representation is:



Step2: Transform into pair RDD. We perform the following to create the required pair RDD:

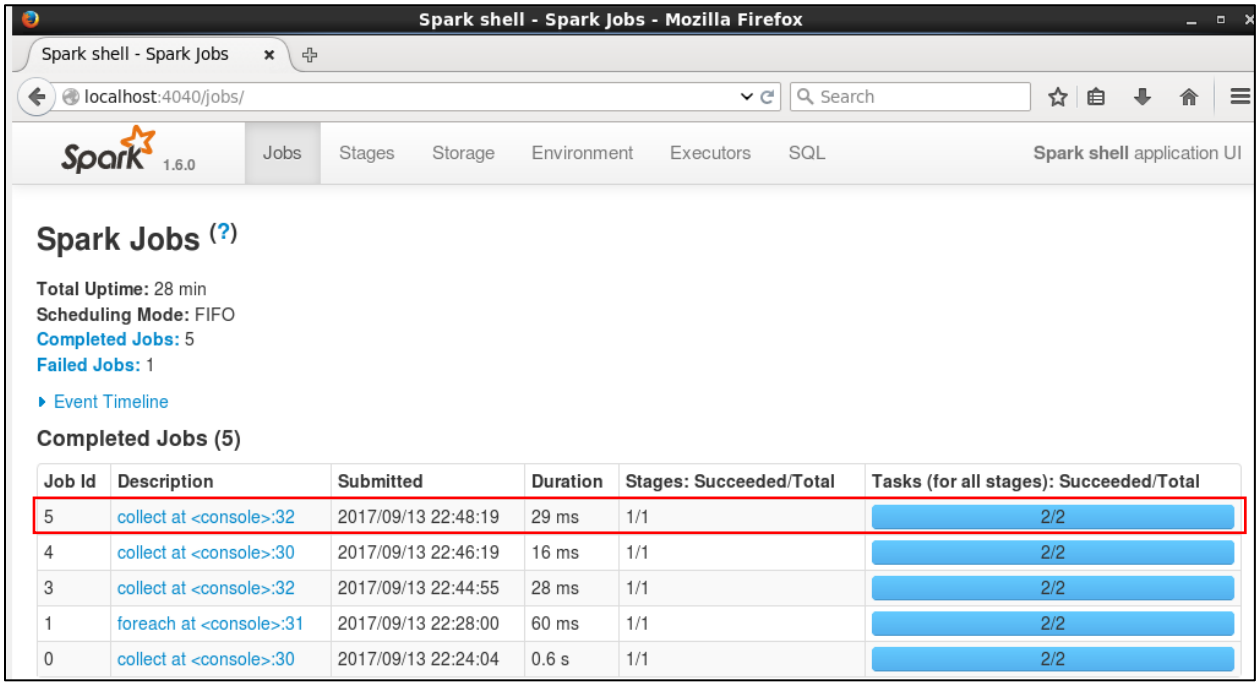
val data1 = data.map(x => (x, a.charAt(0) – ‘a’))

Here, we use the map transformation and make use of **charAt** function to get the required transformation.

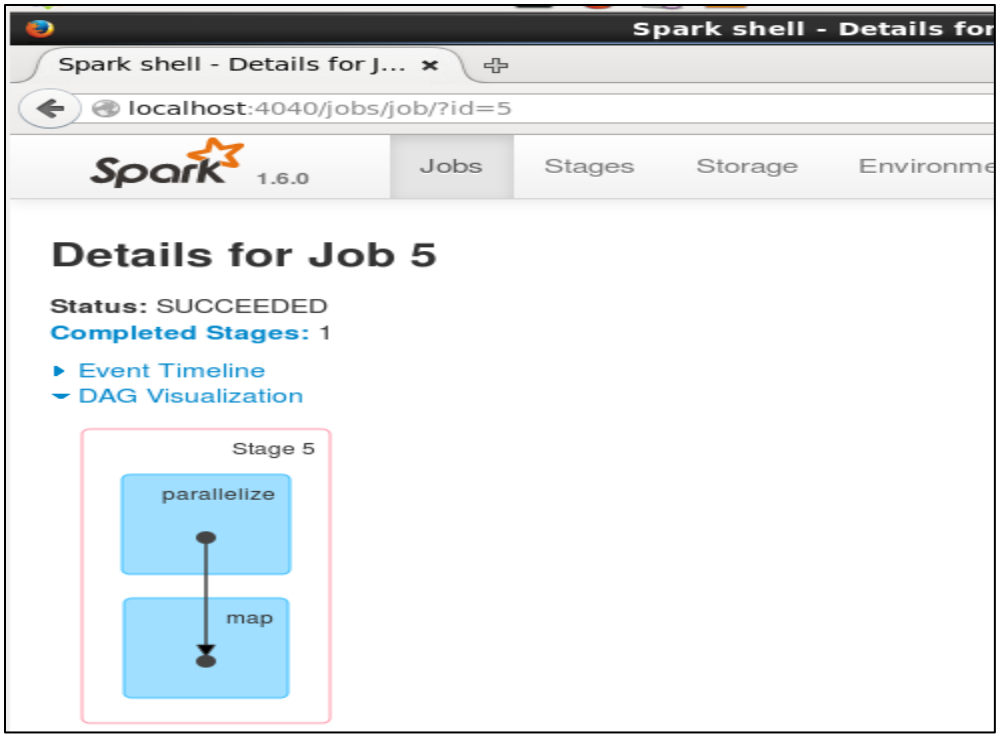
Collect method gives the output as follows:

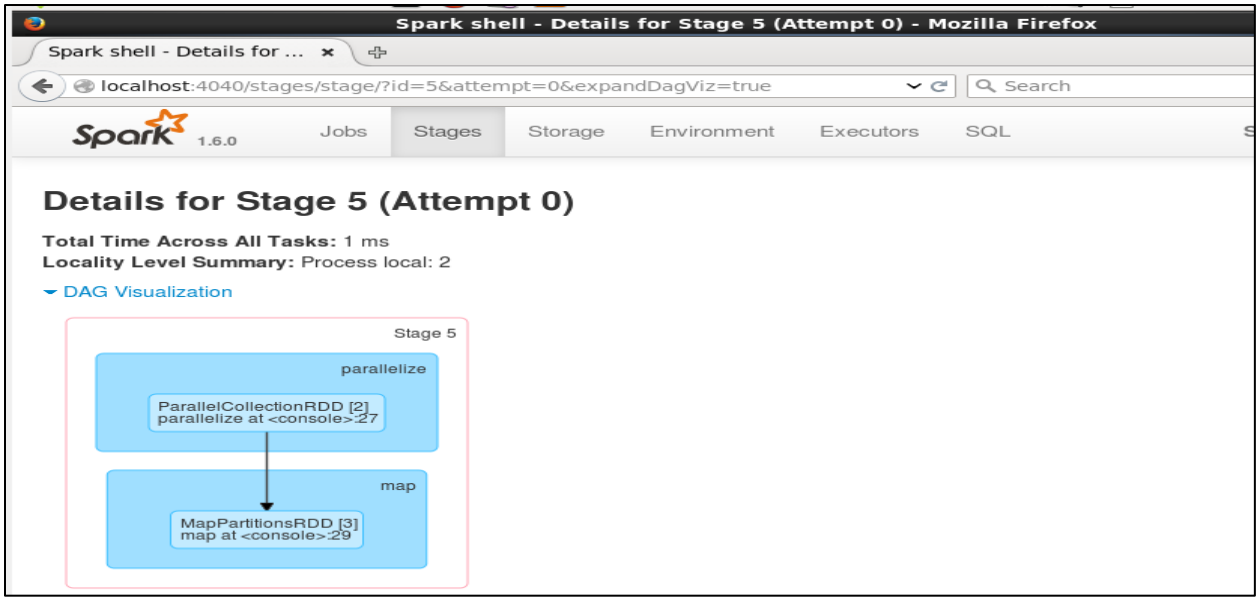
```
scala> val data1  = data.map(x=>(x,x.charAt(0)-'a') );
data1: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[1] at map at <console>:29

scala> data1.collect
res2: Array[(String, Int)] = Array((a,0), (b,1), (c,2), (d,3), (e,4))

scala>
```

Collect being an action, its execution appears in the Spark UI as a job as follows:



Its DAG representation is as follows:


Spark shell - Details for
Spark shell - Details for J...
localhost:4040/jobs/job/?id=5
Spark 1.6.0
Jobs
Stages
Storage
Environme
Details for Job 5
Status: SUCCEEDED
Completed Stages: 1
Event Timeline
DAG Visualization
Stage 5
parallelize
map


Spark shell - Details for Stage 5 (Attempt 0) - Mozilla Firefox
Spark shell - Details for ...
localhost:4040/stages/stage/?id=5&attempt=0&expandDagViz=true
Search
Spark 1.6.0
Jobs
Stages
Storage
Environment
Executors
SQL
Details for Stage 5 (Attempt 0)
Total Time Across All Tasks: 1 ms
Locality Level Summary: Process local: 2
DAG Visualization
Stage 5
parallelize
ParallelCollectionRDD [2]
parallelize at <console>:27
map
MapPartitionsRDD [3]
map at <console>:29