Hadoop Tutorials
Spark

Kacper Surdy

Prasanth Kothuri

About the tutorial

- The third session in Hadoop tutorial series
  - this time given by Kacper and Prasanth

- Session fully dedicated to Spark framework
  - Extensively discussed
  - Actively developed
  - Used in production

- Mixture of a talk and hands-on exercises

What is Spark

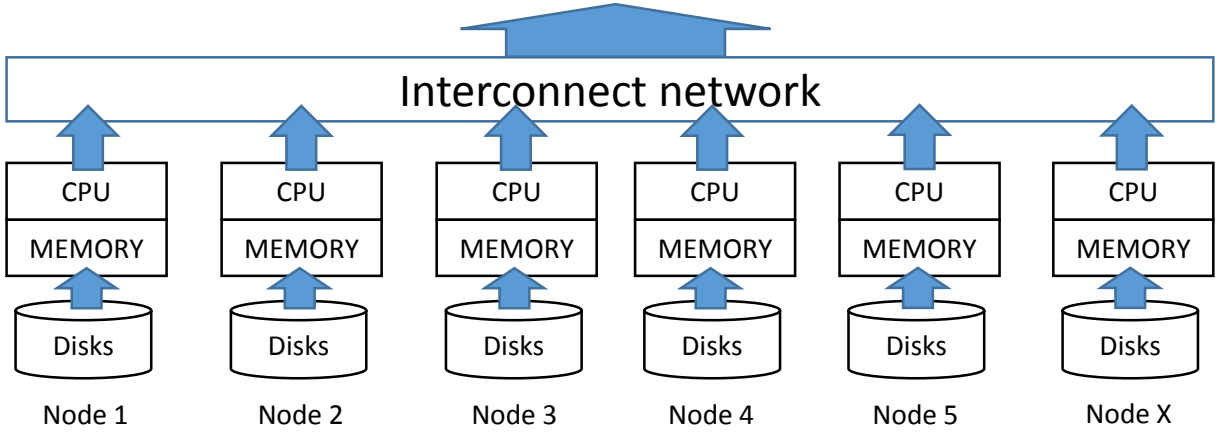
- A framework for performing distributed computations
- Scalable, applicable for processing TBs of data

- Easy programming interface
- Supports Java, Scala, Python, R
- Varied APIs: DataFrames, SQL, MLib, Streaming

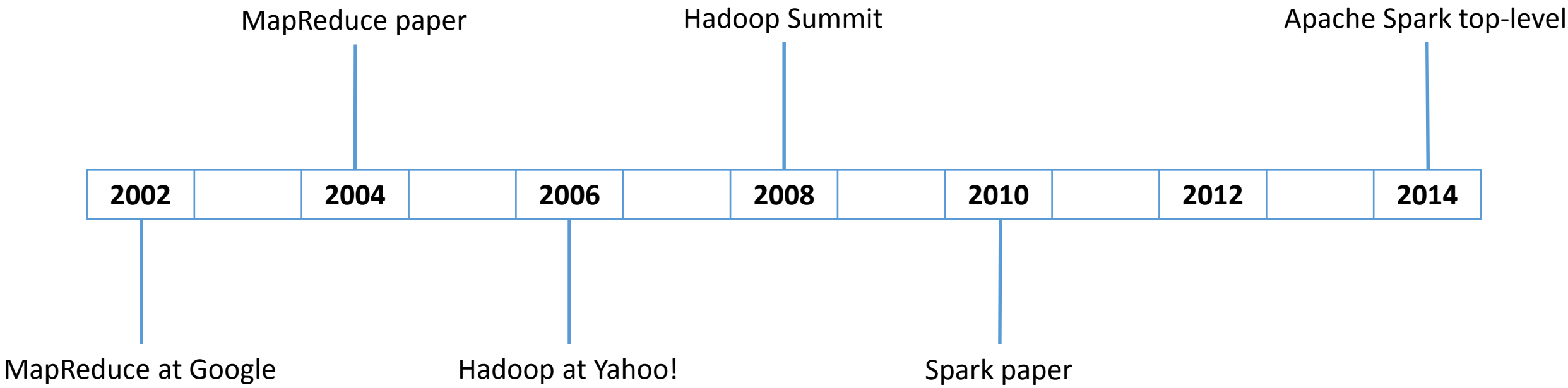
- Multiple cluster deployment modes
- Multiple data sources HDFS, Cassandra, HBase, S3

Compared to Impala

- Similar concept of the workload distribution
- Overlap in SQL functionalities
- Spark is more flexible and offers richer API
- Impala is fine tuned for SQL queries



Evolution from MapReduce



Deployment modes

Local mode

- Make use of multiple cores/CPUs with the thread-level parallelism

Cluster modes:

- Standalone
- Apache Mesos
- Hadoop YARN

  typical for hadoop clusters

with centralised resource management

How to use it

- Interactive shell
  `spark-shell`
  `pyspark`
- Job submission
  `spark-submit` (use also for python)
- Notebooks
  jupyter
  zeppelin
  SWAN (centralised CERN jupyter service)

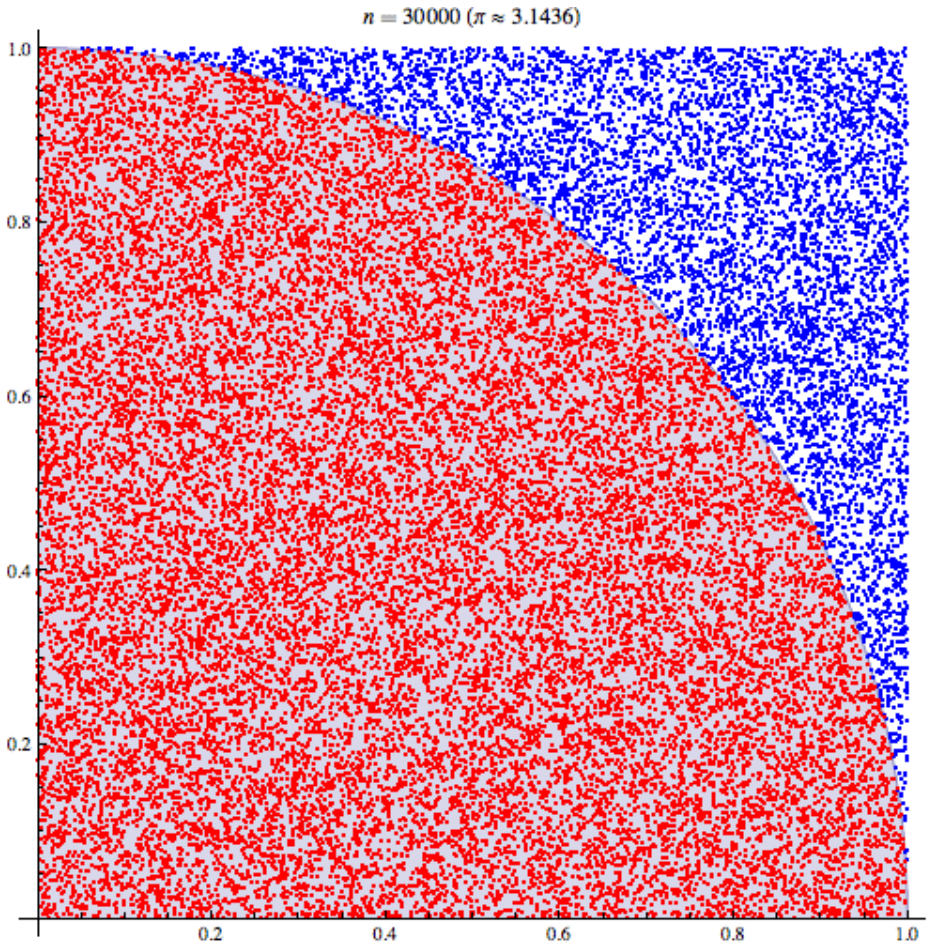
terminal

web interface

Example – Pi estimation

```
import scala.math.random

val slices = 2
val n = 100000 * slices
val rdd = sc.parallelize(1 to n, slices)
val sample = rdd.map { i =>
      val x = random
      val y = random
      if (x*x + y*y < 1) 1 else 0
}
val count = sample.reduce(_ + _)
println("Pi is roughly " + 4.0 * count / n)
```



https://en.wikipedia.org/wiki/Monte_Carlo_method

Example – Pi estimation

- Login to on of the cluster nodes `haperf1[01-12]`
- Start `spark-shell`
- Copy or retype the content of
  `/afs/cern.ch/user/k/kasurdy/public/pi.spark`

Spark concepts and terminology

Transformations, actions (1)

- Transformations define how you want to transform you data. The result of a transformation of a spark dataset is another spark dataset. They do not trigger a computation.

  examples: map, filter, union, distinct, join

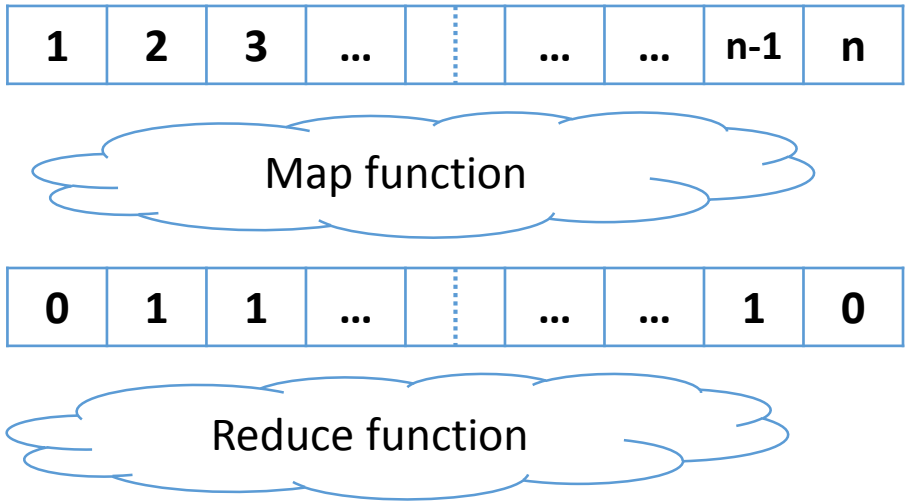
- Actions are used to collect the result from a dataset defined previously. The result is usually an array or a number. They start the computation.

  examples: reduce, collect, count, first, take

Transformations, actions (2)

```
import scala.math.random

val slices = 2
val n = 100000 * slices
val rdd = sc.parallelize(1 to n, slices)
val sample = rdd.map { i =>
      val x = random
      val y = random
      if (x*x + y*y < 1) 1 else 0
}
val count = sample.reduce(_ + _)
println("Pi is roughly " + 4.0 * count / n)
```

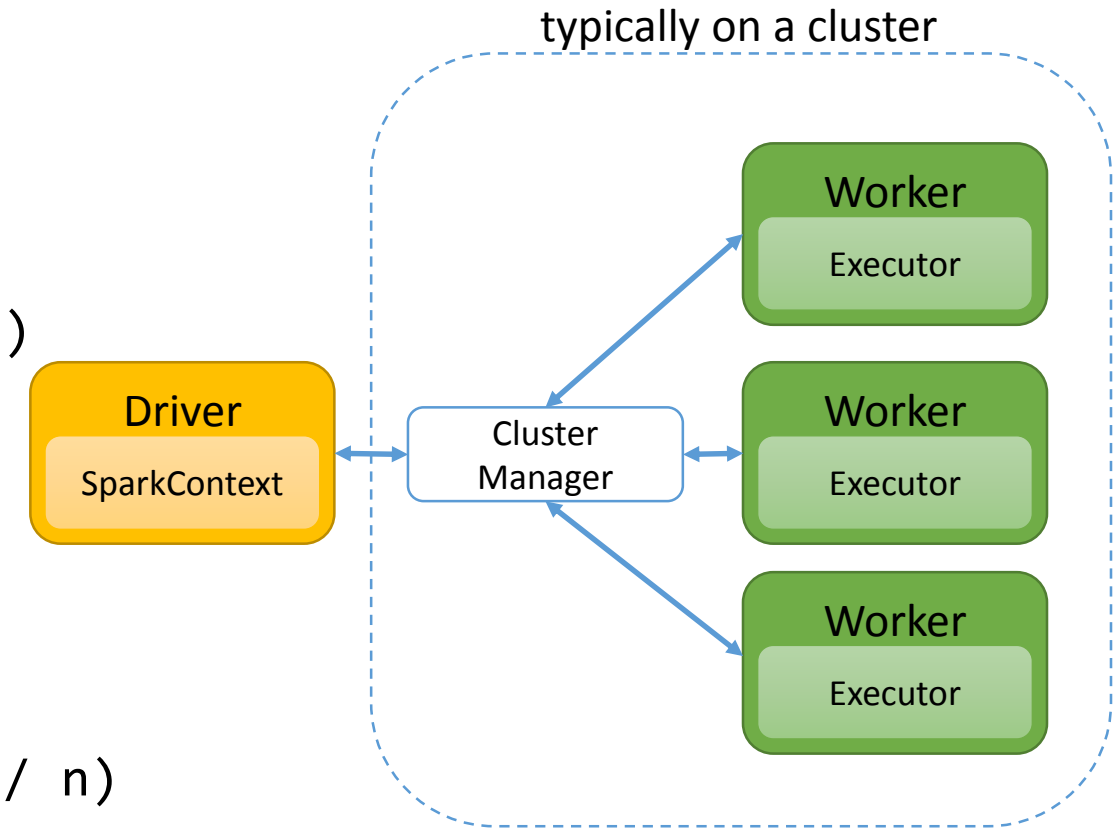


4 * count = 157018 / n = 200000

Driver, worker, executor

```
import scala.math.random

val slices = 2
val n = 100000 * slices
val rdd = sc.parallelize(1 to n, slices)
val sample = rdd.map { i =>
      val x = random
      val y = random
      if (x*x + y*y < 1) 1 else 0
}
val count = sample.reduce(_ + _)
println("Pi is roughly " + 4.0 * count / n)
```

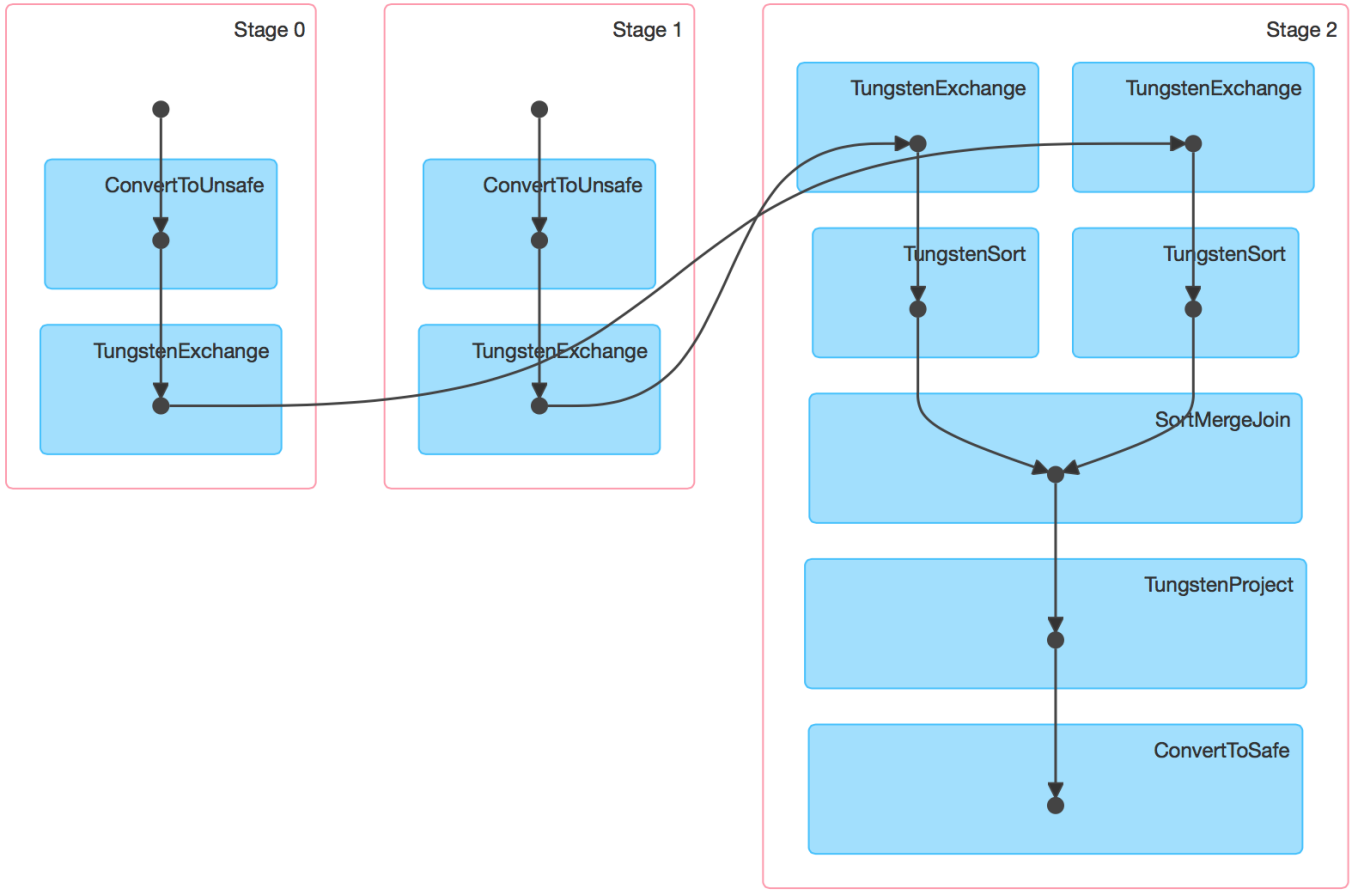


Stages, tasks

- Task - A unit of work that will be sent to one executor

- Stage - Each job gets divided into smaller sets of “tasks”
  called *stages* that depend on each other aka synchronization points

Your job as a directed acyclic graph (DAG)



https://dwtobigdata.wordpress.com/2015/09/29/etl-with-apache-spark/

Monitoring pages

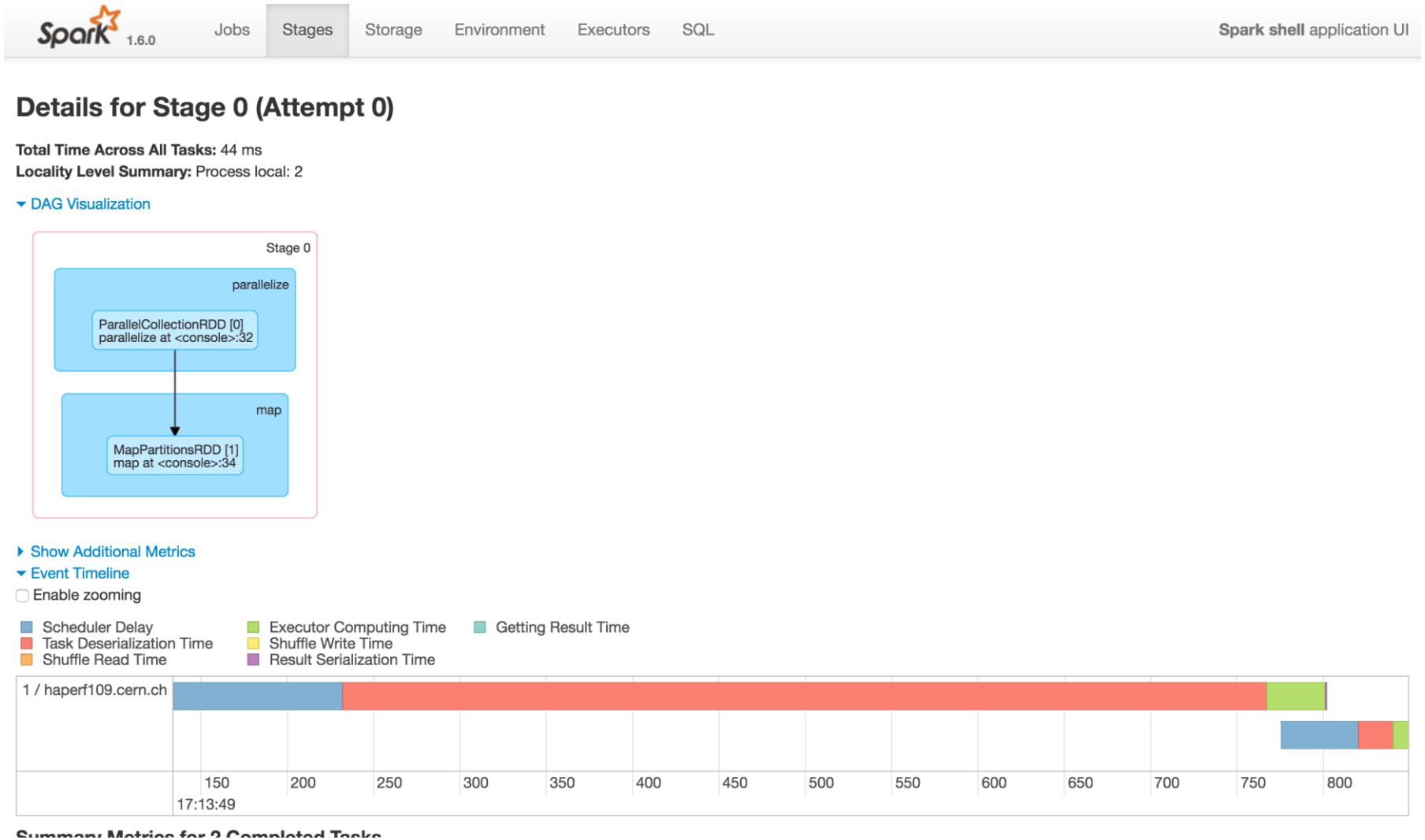
- If you can, scroll the console output up to get an application ID (e.g. application_1468968864481_0070) and open

http://haperf100.cern.ch:8088/proxy/application_XXXXXXXXXXXXX_XXXX/

- Otherwise use the link below to find you application on the list

http://haperf100.cern.ch:8088/cluster

Monitoring pages



Data APIs