Deploying and Administering Spark

Patrick Wendell
Databricks

Outline

Spark components

Cluster managers

Hardware & configuration

Linking with Spark

Monitoring and measuring

Outline

Spark components

Cluster managers

Hardware & configuration

Linking with Spark

Monitoring and measuring

Spark application

Driver program

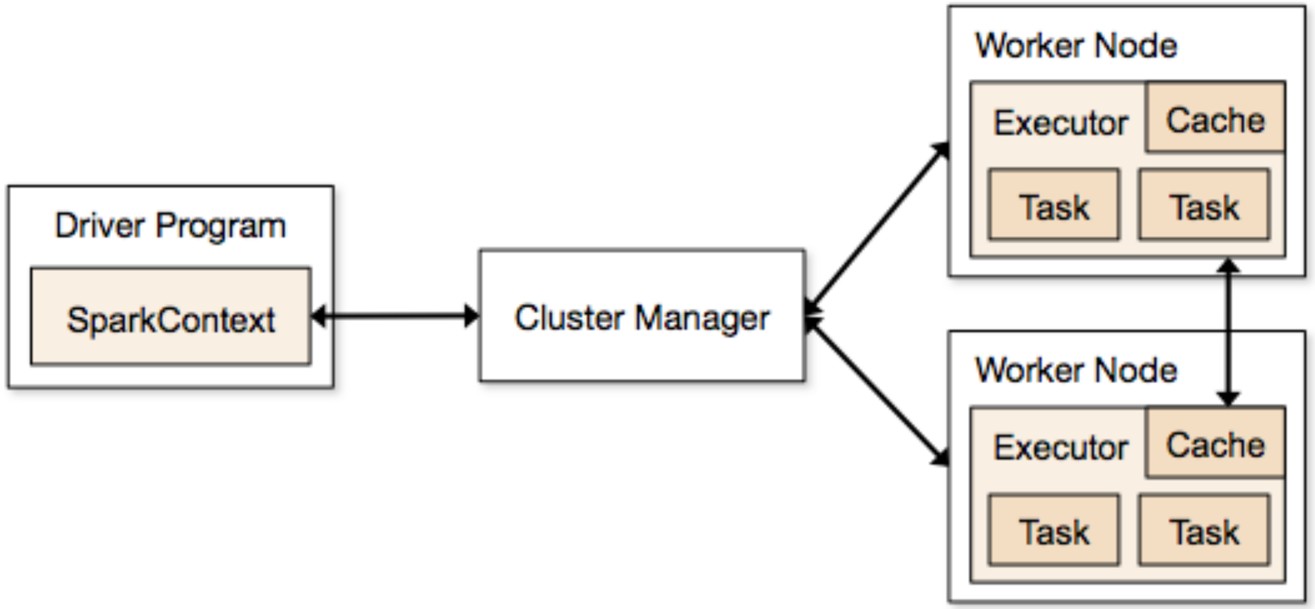
- Java program that creates a SparkContext

Executors

- Worker processes that execute tasks and store data

Cluster manager

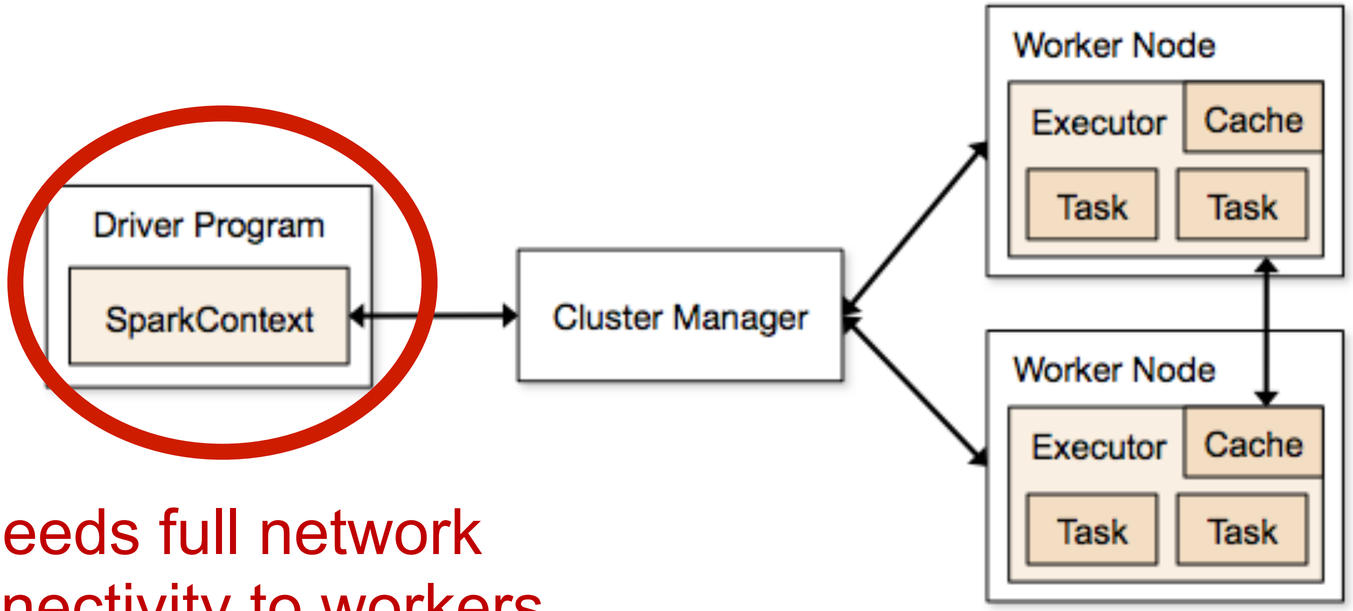
Cluster manager grants executors to a Spark application



databricks™

Driver program

Driver program decides when to launch tasks on which executor



Needs full network connectivity to workers

Types of Applications

Long lived/shared applications

- Shark
- Spark Streaming
- Job Server (Ooyala)

May do mutli-user scheduling within allocation from cluster manger

Short lived applications

- Standalone apps
- Shell sessions

databricks™

Outline

Spark components

Cluster managers

Hardware & configuration

Linking with Spark

Monitoring and measuring

Cluster Managers

Several ways to deploy Spark

1. Standalone mode (on-site)
2. Standalone mode (EC2)
3. YARN
4. Mesos
5. SIMR [not covered in this talk]

Standalone Mode

Bundled with Spark

Great for quick “dedicated” Spark cluster

H/A mode for long running applications (0.8.1+)

Standalone Mode

1. (Optional) describe amount of resources in conf/spark-env.sh

   - SPARK_WORKER_CORES

   - SPARK_WORKER_MEMORY

2. List slaves in conf/slaves

3. Copy configuration to slaves

4. Start/stop using ./bin/stop-all and ./bin/start-all

databricks™

Standalone Mode

Some support for inter-application scheduling

Set spark.cores.max to limit # of cores each application can use

EC2 Deployment

Launcher bundled with Spark

Create cluster in 5 minutes

Sizes cluster for any EC2 instance type and # of nodes

Used widely by Spark team for internal testing

databricks™

EC2 Deployment

```
./spark-ec2
  -t [instance type]
  -k [key-name]
  -i [path-to-key-file]
  -s [num-slaves]
  -r [ec2-region]
  --spot-price=[spot-price]
```

EC2 Deployment

Creates:

Spark Sandalone cluster at <ec2-master>:8080

HDFS cluster at < ec2-master >:50070

MapReduce cluster at < ec2-master >:50030

Apache Mesos

General-purpose cluster manager that can run Spark, Hadoop MR, MPI, etc

Simply pass mesos://<master-url> to SparkContext

Optional: set spark.executor.uri to a pre-built Spark package in HDFS, created by make-distribution.sh

Mesos Run Modes

Fine-grained (default):

- Apps get static memory allocations, but share CPU dynamically on each node

Coarse-grained:

- Apps get static CPU and memory allocations
- Better predictability and latency, possibly at cost of utilization

Hadoop YARN

In Spark 0.8.0:

- Runs standalone apps only, launching driver inside YARN cluster
- YARN 0.23 to 2.0.x

Coming in 0.8.1:

- Interactive shell
- YARN 2.2.x support
- Support for hosting Spark JAR in HDFS

databricks™

YARN Steps

1. Build Spark assembly JAR
2. Package your app into a JAR
3. Use the yarn.Client class

```
SPARK_JAR=<SPARK_ASSEMBLY_JAR> ./spark-class org.apache.spark.deploy.yarn.Client \
  --jar <YOUR_APP_JAR> --class <MAIN_CLASS> \
  --args <MAIN_ARGUMENTS> \
  --num-workers <N> \
  --master-memory <MASTER_MEM> \
  --worker-memory <WORKER_MEM> \
  --worker-cores <CORES_PER_WORKER>
```

More Info

http://spark.incubator.apache.org/docs/latest/cluster-overview.html

Detailed docs about each of standalone mode, Mesos, YARN, EC2

Outline

Cluster components

Deployment options

Hardware & configuration

Linking with Spark

Monitoring and measuring

Where to run Spark?

If using HDFS, run on same nodes or within LAN

1. Have dedicated (usually "beefy") nodes for Spark
2. Colocate Spark and MapReduce on shared nodes

Local Disks

Spark uses disk for writing shuffle data and paging out RDD's

Ideally have several disks per node in JBOD configuration

Set spark.local.dir with comma-separated disk locations

Memory

Recommend 8GB heap and up

Generally, more is better

For massive (>200GB) heaps you may want to increase # of executors per node (see SPARK_WORKER_INSTANCES)

Network/CPU

For in-memory workloads, network and CPU are often the bottleneck

Ideally use 10Gb Ethernet

Works well on machines with multiple cores (since parallel)

Environment-related configs

spark.executor.memory

How much memory you will ask for from cluster manager

spark.local.dir

Where spark stores shuffle files

Outline

Cluster components

Deployment options

Hardware & configuration

Linking with Spark

Monitoring and measuring

Typical Spark Application

sc = new SparkContext(<cluster-manager>…)

sc.addJar("/uber-app-jar.jar")

Created using maven or sbt assembly

sc.textFile(XX)
…reduceBy
…saveAS

databricks™

Linking with Spark

Add an ivy/maven dependency in your project on spark-core artifact

If using HDFS, add dependency on hadoop-client for your version

e.g. 1.2.0, 2.0.0-cdh4.3.1

For YARN, also add spark-yarn

Hadoop Versions

Distribution	Release	Maven Version Code
CDH	4.X.X	2.0.0-mr1-chd4.X.X
	4.X.X (YARN mode)	2.0.0-chd4.X.X
	3uX	0.20.2-cdh3uX
HDP	1.3	1.2.0
	1.2	1.1.2
	1.1	1.0.3

See Spark docs for details:
http://spark.incubator.apache.org/docs/latest/hadoop-third-party-distributions.html

databricks™

Outline

Cluster components

Deployment options

Hardware & configuration

Linking with Spark

Monitoring and measuring

Monitoring

Cluster Manager UI

Executor Logs

Spark Driver Logs

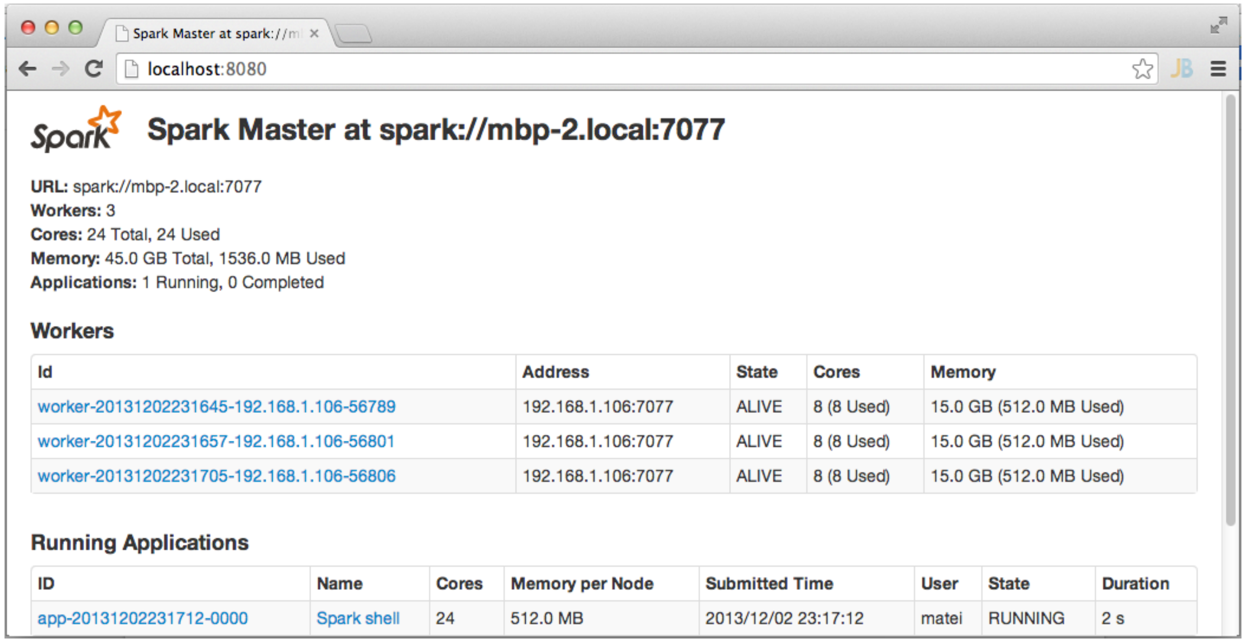
Application Web UI

Spark Metrics

Cluster Manager UI

Standalone mode: <master>:8080

Mesos, YARN have their own UIs



Spark Master at spark://mbp-2.local:7077

URL: spark://mbp-2.local:7077
Workers: 3
Cores: 24 Total, 24 Used
Memory: 45.0 GB Total, 1536.0 MB Used
Applications: 1 Running, 0 Completed

Workers

Id	Address	State	Cores	Memory
worker-20131202231645-192.168.1.106-56789	192.168.1.106:7077	ALIVE	8 (8 Used)	15.0 GB (512.0 MB Used)
worker-20131202231657-192.168.1.106-56801	192.168.1.106:7077	ALIVE	8 (8 Used)	15.0 GB (512.0 MB Used)
worker-20131202231705-192.168.1.106-56806	192.168.1.106:7077	ALIVE	8 (8 Used)	15.0 GB (512.0 MB Used)

Running Applications

ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20131202231712-0000	Spark shell	24	512.0 MB	2013/12/02 23:17:12	matei	RUNNING	2 s

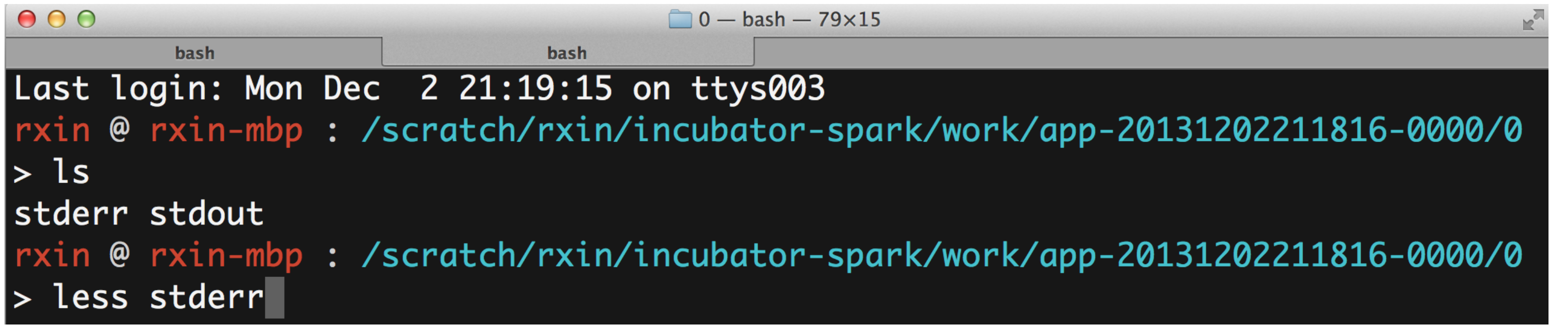
Executor Logs

Stored by cluster manager on each worker

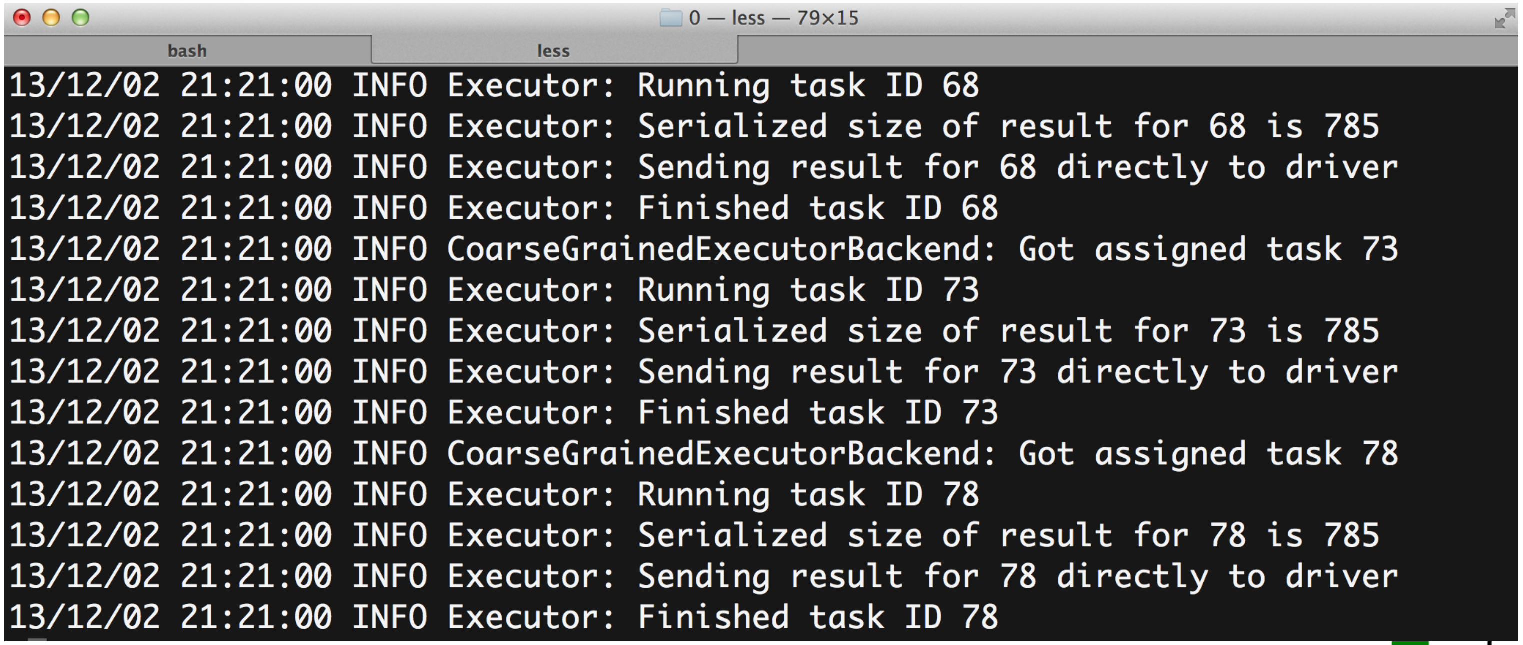
Default location in standalone mode:

/path/to/spark/work

Executor Logs


0 — bash — 79×15
bash | bash

```
Last login: Mon Dec  2 21:19:15 on ttys003
rxin @ rxin-mbp : /scratch/rxin/incubator-spark/work/app-20131202211816-0000/0
> ls
stderr stdout
rxin @ rxin-mbp : /scratch/rxin/incubator-spark/work/app-20131202211816-0000/0
> less stderr
```


0 — less — 79×15
bash | less

```
13/12/02 21:21:00 INFO Executor: Running task ID 68
13/12/02 21:21:00 INFO Executor: Serialized size of result for 68 is 785
13/12/02 21:21:00 INFO Executor: Sending result for 68 directly to driver
13/12/02 21:21:00 INFO Executor: Finished task ID 68
13/12/02 21:21:00 INFO CoarseGrainedExecutorBackend: Got assigned task 73
13/12/02 21:21:00 INFO Executor: Running task ID 73
13/12/02 21:21:00 INFO Executor: Serialized size of result for 73 is 785
13/12/02 21:21:00 INFO Executor: Sending result for 73 directly to driver
13/12/02 21:21:00 INFO Executor: Finished task ID 73
13/12/02 21:21:00 INFO CoarseGrainedExecutorBackend: Got assigned task 78
13/12/02 21:21:00 INFO Executor: Running task ID 78
13/12/02 21:21:00 INFO Executor: Serialized size of result for 78 is 785
13/12/02 21:21:00 INFO Executor: Sending result for 78 directly to driver
13/12/02 21:21:00 INFO Executor: Finished task ID 78
```

databricks™

Spark Driver Logs

Spark initializes a log4j when created

Include log4j.properties file on the classpath

See example in conf/
log4j.properties.template

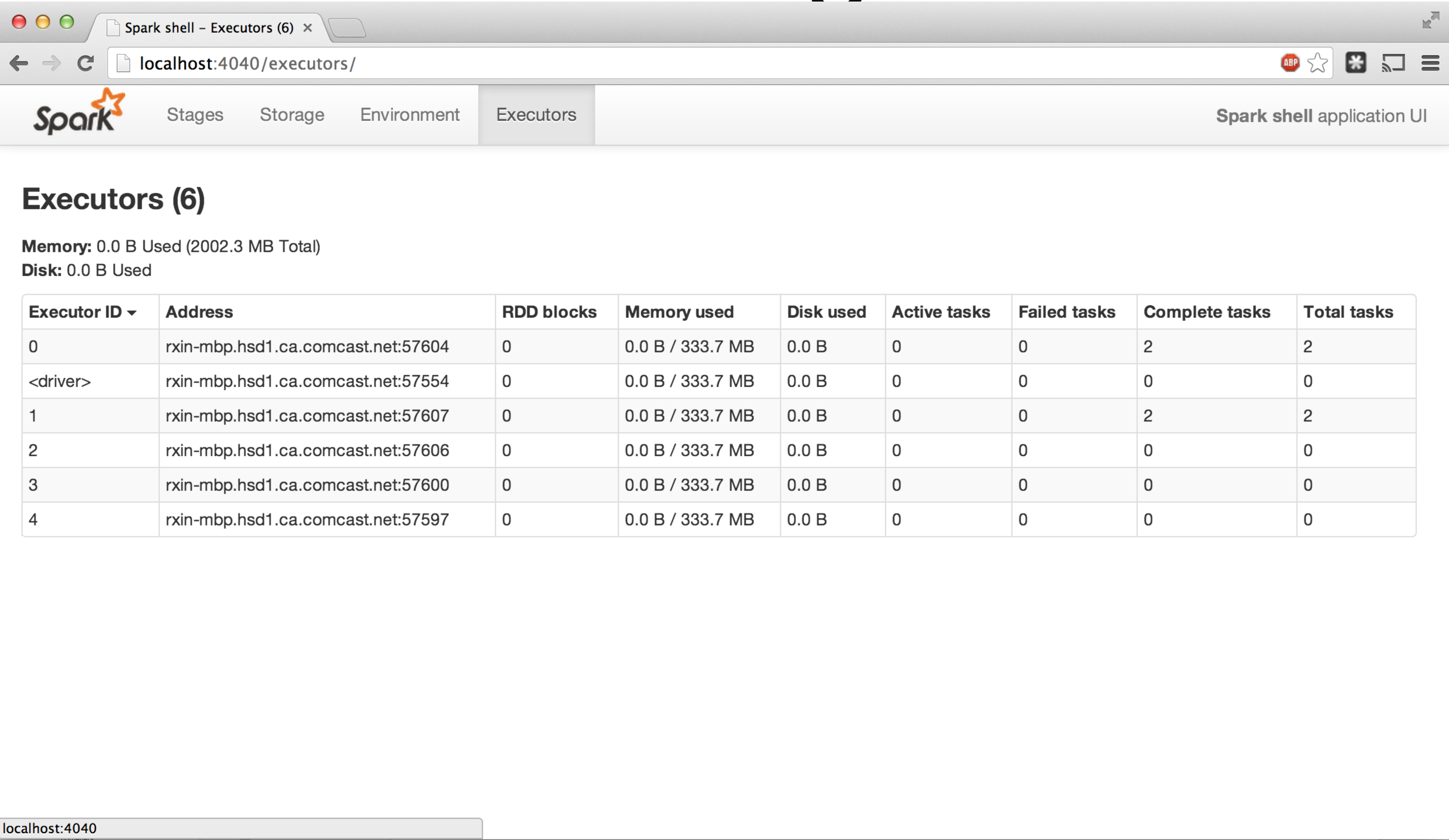
Application Web UI

http://spark-application-host:4040

(or use spark.ui.port to configure the port)

For executor / task / stage / memory status, etc

Executors Page


Spark shell – Executors (6)
localhost:4040/executors/
Spark
Stages
Storage
Environment
Executors
Spark shell application UI

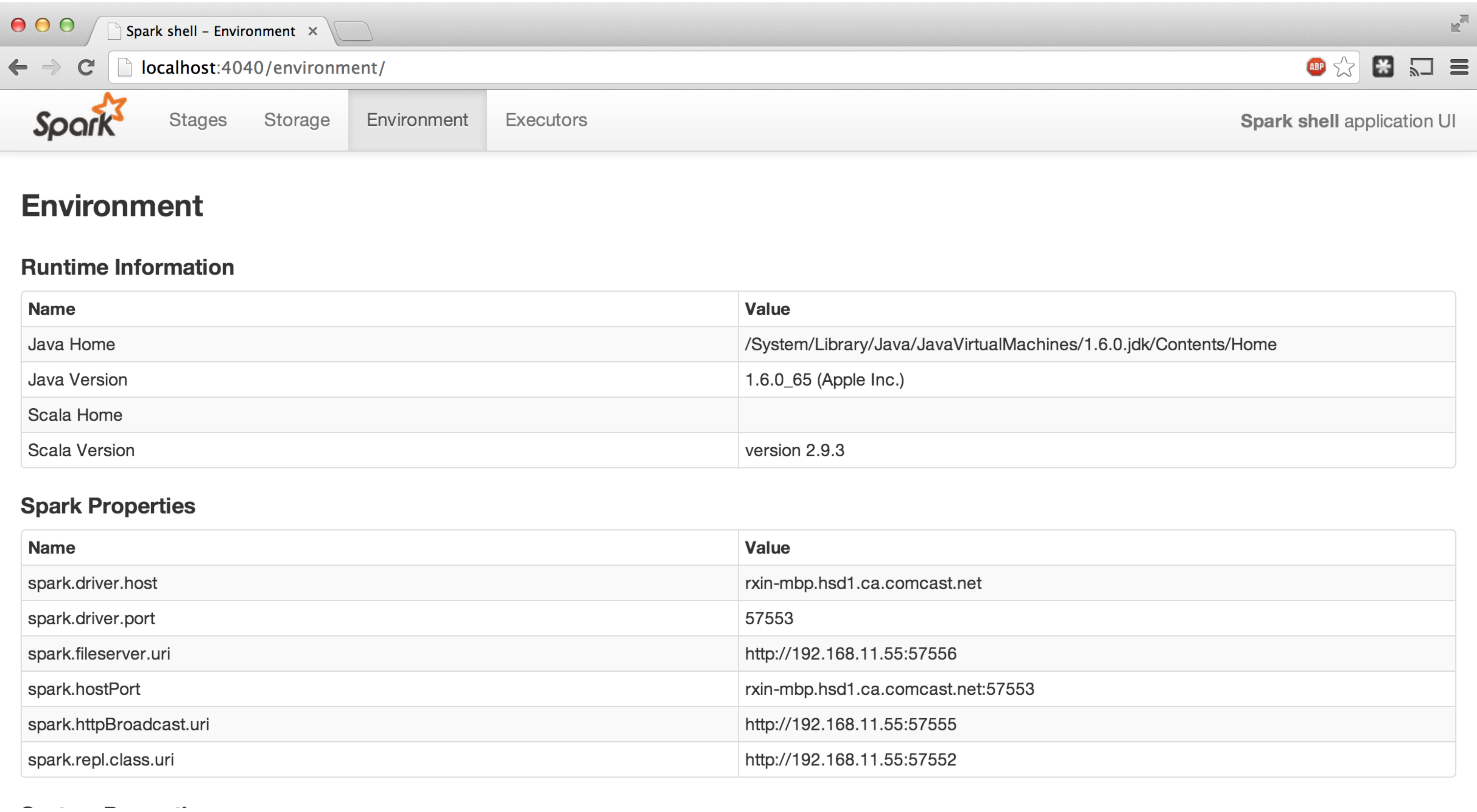
Executors (6)

Memory: 0.0 B Used (2002.3 MB Total)
Disk: 0.0 B Used

Executor ID ▾	Address	RDD blocks	Memory used	Disk used	Active tasks	Failed tasks	Complete tasks	Total tasks
0	rxin-mbp.hsd1.ca.comcast.net:57604	0	0.0 B / 333.7 MB	0.0 B	0	0	2	2
<driver>	rxin-mbp.hsd1.ca.comcast.net:57554	0	0.0 B / 333.7 MB	0.0 B	0	0	0	0
1	rxin-mbp.hsd1.ca.comcast.net:57607	0	0.0 B / 333.7 MB	0.0 B	0	0	2	2
2	rxin-mbp.hsd1.ca.comcast.net:57606	0	0.0 B / 333.7 MB	0.0 B	0	0	0	0
3	rxin-mbp.hsd1.ca.comcast.net:57600	0	0.0 B / 333.7 MB	0.0 B	0	0	0	0
4	rxin-mbp.hsd1.ca.comcast.net:57597	0	0.0 B / 333.7 MB	0.0 B	0	0	0	0

localhost:4040

Environment Page


Spark shell - Environment
localhost:4040/environment/
Spark
Stages
Storage
Environment
Executors
Spark shell application UI

Environment

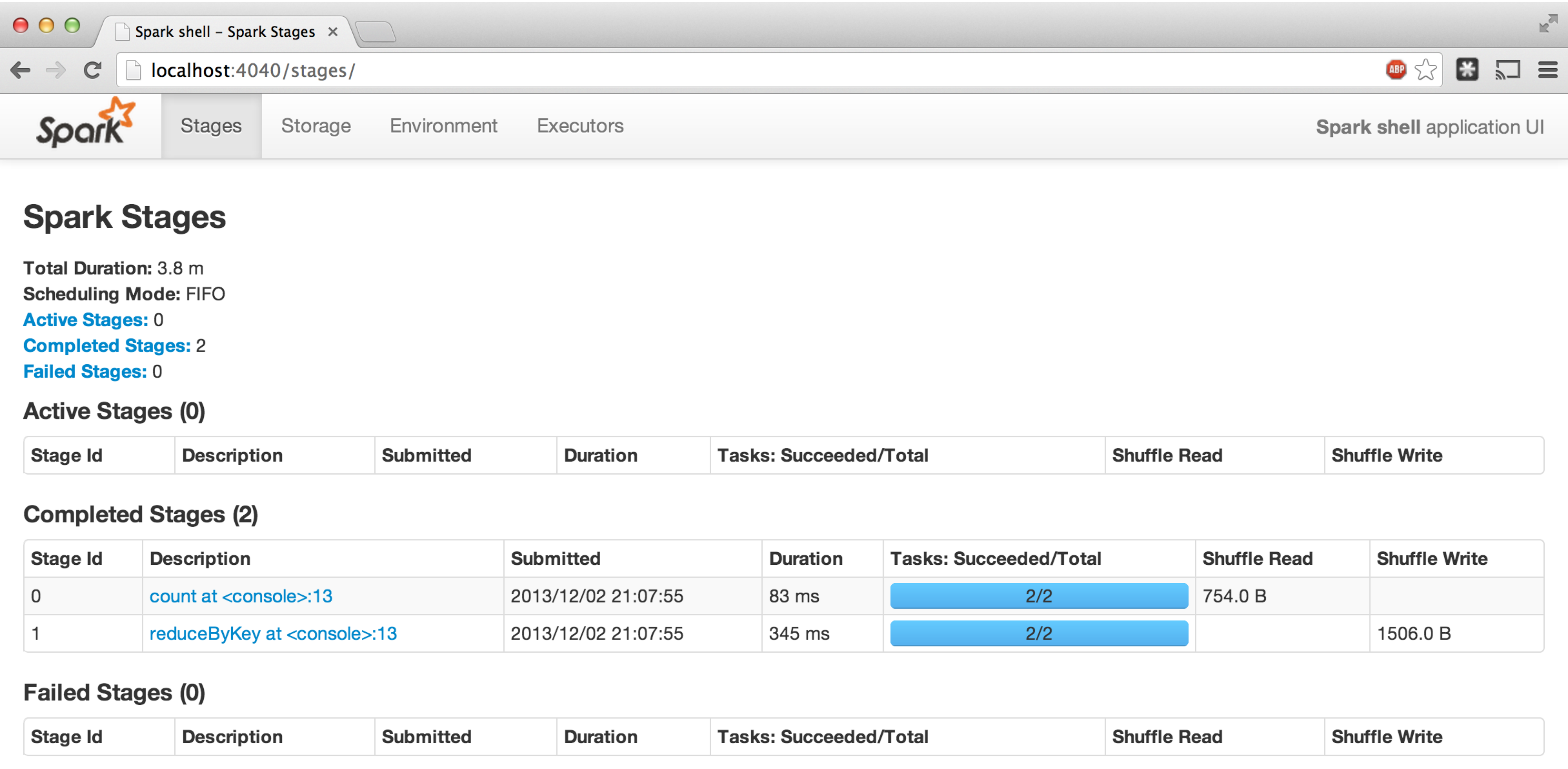
Runtime Information

Name	Value
Java Home	/System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home
Java Version	1.6.0_65 (Apple Inc.)
Scala Home	
Scala Version	version 2.9.3

Spark Properties

Name	Value
spark.driver.host	rxin-mbp.hsd1.ca.comcast.net
spark.driver.port	57553
spark.fileserver.uri	http://192.168.11.55:57556
spark.hostPort	rxin-mbp.hsd1.ca.comcast.net:57553
spark.httpBroadcast.uri	http://192.168.11.55:57555
spark.repl.class.uri	http://192.168.11.55:57552

Stage Information



Spark Stages

Total Duration: 3.8 m
Scheduling Mode: FIFO
Active Stages: 0
Completed Stages: 2
Failed Stages: 0

Active Stages (0)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Shuffle Read	Shuffle Write

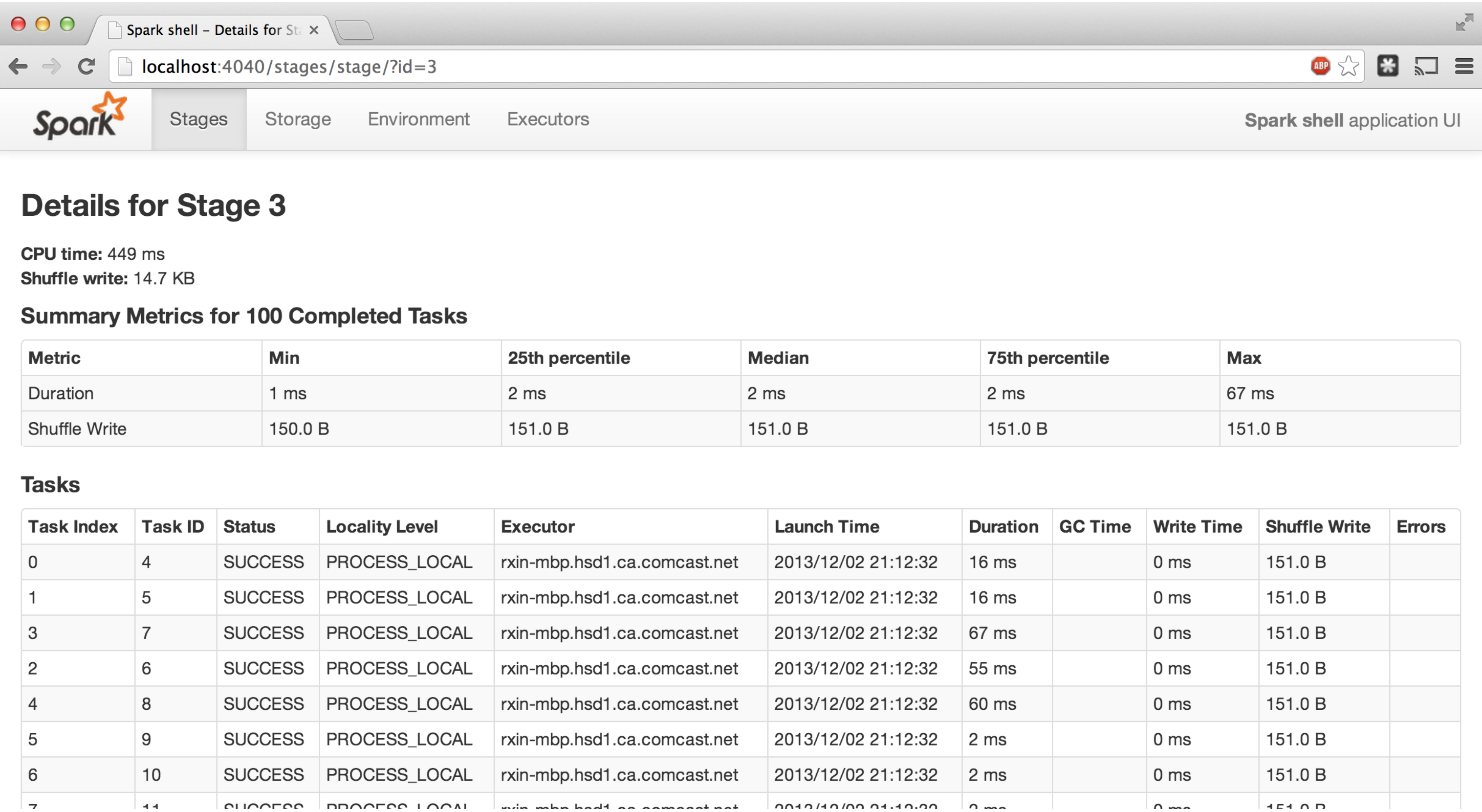
Completed Stages (2)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Shuffle Read	Shuffle Write
0	count at <console>:13	2013/12/02 21:07:55	83 ms	2/2	754.0 B	
1	reduceByKey at <console>:13	2013/12/02 21:07:55	345 ms	2/2		1506.0 B

Failed Stages (0)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Shuffle Read	Shuffle Write

Task Breakdown


Spark shell – Details for St
localhost:4040/stages/stage/?id=3
Spark
Stages
Storage
Environment
Executors
Spark shell application UI

Details for Stage 3

CPU time: 449 ms
Shuffle write: 14.7 KB

Summary Metrics for 100 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	1 ms	2 ms	2 ms	2 ms	67 ms
Shuffle Write	150.0 B	151.0 B	151.0 B	151.0 B	151.0 B

Tasks

Task Index	Task ID	Status	Locality Level	Executor	Launch Time	Duration	GC Time	Write Time	Shuffle Write	Errors
0	4	SUCCESS	PROCESS_LOCAL	rxin-mbp.hsd1.ca.comcast.net	2013/12/02 21:12:32	16 ms		0 ms	151.0 B	
1	5	SUCCESS	PROCESS_LOCAL	rxin-mbp.hsd1.ca.comcast.net	2013/12/02 21:12:32	16 ms		0 ms	151.0 B	
3	7	SUCCESS	PROCESS_LOCAL	rxin-mbp.hsd1.ca.comcast.net	2013/12/02 21:12:32	67 ms		0 ms	151.0 B	
2	6	SUCCESS	PROCESS_LOCAL	rxin-mbp.hsd1.ca.comcast.net	2013/12/02 21:12:32	55 ms		0 ms	151.0 B	
4	8	SUCCESS	PROCESS_LOCAL	rxin-mbp.hsd1.ca.comcast.net	2013/12/02 21:12:32	60 ms		0 ms	151.0 B	
5	9	SUCCESS	PROCESS_LOCAL	rxin-mbp.hsd1.ca.comcast.net	2013/12/02 21:12:32	2 ms		0 ms	151.0 B	
6	10	SUCCESS	PROCESS_LOCAL	rxin-mbp.hsd1.ca.comcast.net	2013/12/02 21:12:32	2 ms		0 ms	151.0 B	

App UI Features

Stages show where each operation originated in code

All tables sortable by task length, locations, etc

Metrics

Configurable metrics based on Coda Hale’s Metrics library

Many Spark components can report metrics (driver, executor, application)

Outputs: REST, CSV, Ganglia, JMX, JSON Servlet

Metrics

More details:
http://spark.incubator.apache.org/docs/latest/monitoring.html

More Information

Official docs:
http://spark.incubator.apache.org/docs/latest

Look for Apache Spark parcel in CDH