Spark Fundamentals

Configuring and monitoring Spark applications

Contents

Configuring and monitoring Spark applications

This lab exercise will show you where you can specify configurations for your Spark environment. It will introduce you to some monitoring tools and methods for tuning Spark applications.

After completing this hands-on lab, you should be able to:

- Understand how to configure, monitor and tune Spark applications.

Allow 15 minutes to complete this section of lab.

1.1 Configuring Spark applications

Spark properties control most application settings and are configured separately for each application. These properties can be set directly on the SparkConf object and passed to your SparkContext. You can also set properties by providing it at runtime. For example, submitting a job using the spark-submit command and passing in arguments to that command. In this section, you will set some common properties.

__1. Note: This does not work under the new QSE v4 VM and is kept here for reference. Change the logs from INFO to ERROR, so that each time the Spark shell runs, you only get pertinent information. Open up and edit the $SPARK_HOME/conf/log4j.properties file. If that file does not exist, make a copy of the log4j.properties.template one and rename it to log4j.properties.

```
biadmin@ibmclass:~/Desktop> cat /opt/ibm/biginsights/spark/conf/log4j.properties

# Set everything to be logged to the console
log4j.rootCategory=ERROR, console
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.target=System.err
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d{yy/MM/dd HH:mm:ss} %p %c{1}:
%m%n
```

Next, launch a Spark shell with two cores. You should notice that you no longer see INFO messages on the console.

$SPARK_HOME/bin/spark-shell --master local[2]

This launches a shell that is only going to utilize two cores on the node, which is the minimum number of cores needed for parallelism. Go ahead and quit out of the Spark shell.

__2. The *spark-submit* command supports loading configurations dynamically. In fact, this is probably the more common usage of loading configurations. You have seen this used in previous lab exercises when you ran sample applications as well as the ones you developed.

__3. Additionally, the spark-submit will also read configurations options from conf/spark-defaults.conf. Let's look at it now:

cat $SPARK_HOME/conf/spark-defaults.conf

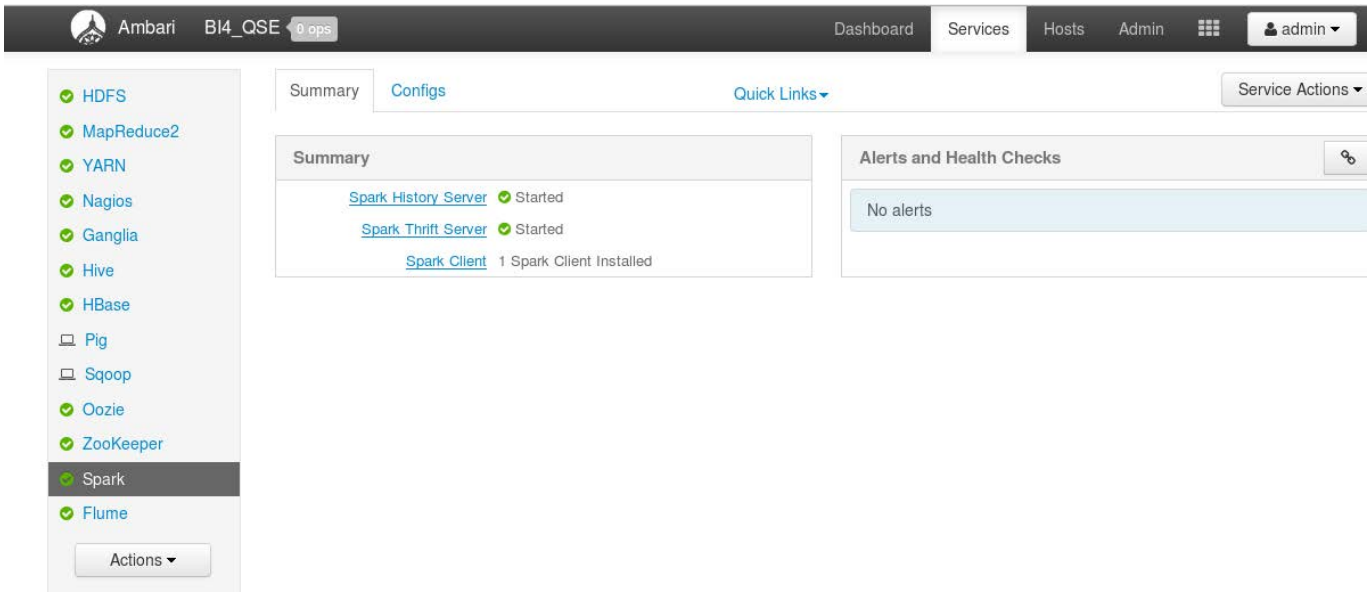
```
biadmin@ibmclass:/opt/ibm/biginsights/bin> cat /opt/ibm/biginsights/spark/conf/s
park-defaults.conf
# Default system properties included when running spark-submit.
# This is useful for setting default environmental settings.

# Example:
# spark.master                     spark://master:7077
# spark.eventLog.enabled           true
# spark.eventLog.dir               hdfs://namenode:8021/directory
# spark.serializer                 org.apache.spark.serializer.KryoSerializer
# spark.driver.memory              5g
# spark.executor.extraJavaOptions  -XX:+PrintGCDetails -Dkey=value -Dnumbers="on
e two three"
spark.master yarn-client
spark.yarn.jar    hdfs://ibmclass.localdomain.com:9000/biginsights/spark/assembl
y/target/scala-2.10/spark-assembly-1.1.0-10-hadoop2.4.1.jar
spark.eventLog.enabled    true
spark.eventLog.dir    hdfs://ibmclass.localdomain.com:9000/biginsights/logs/spar
k/historyServer
spark.yarn.historyServer.address    ibmclass.localdomain.com:18080
#spark.metrics.conf    /opt/ibm/biginsights/spark/conf/metrics.properties
```

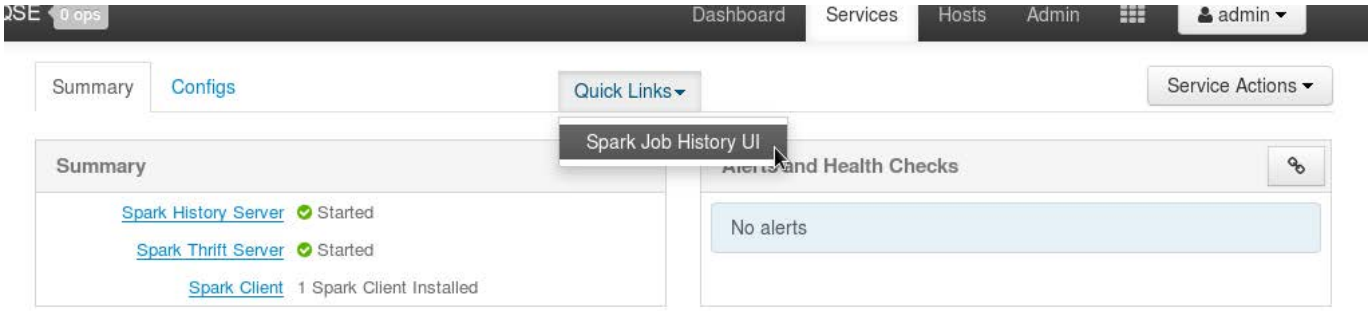
Properties set directly on the SparkConf take highest precedence, then flags passed to spark-submit or spark-shell, then options in the spark-defaults.conf file.

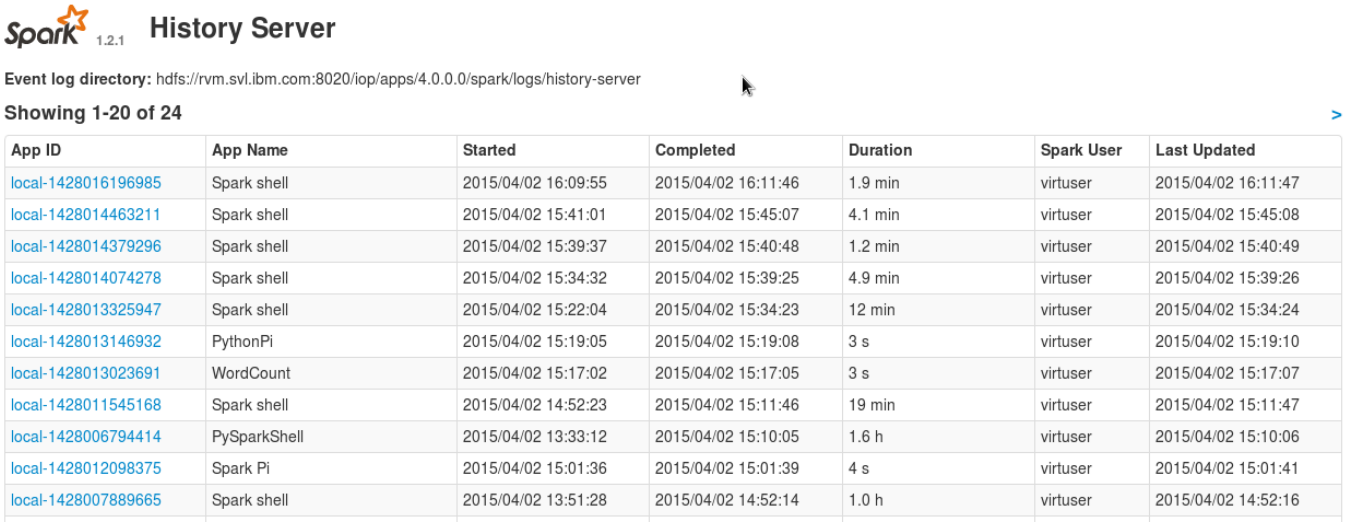
1.2 Monitoring Spark applications

__1. To view the history server, you can find the port by going to the Ambari console and accessing Spark on the Cluster Status tab:



__2. To access the history server, click on the Quick Links:





App ID	App Name	Started	Completed	Duration	Spark User	Last Updated
local-1428016196985	Spark shell	2015/04/02 16:09:55	2015/04/02 16:11:46	1.9 min	virtuser	2015/04/02 16:11:47
local-1428014463211	Spark shell	2015/04/02 15:41:01	2015/04/02 15:45:07	4.1 min	virtuser	2015/04/02 15:45:08
local-1428014379296	Spark shell	2015/04/02 15:39:37	2015/04/02 15:40:48	1.2 min	virtuser	2015/04/02 15:40:49
local-1428014074278	Spark shell	2015/04/02 15:34:32	2015/04/02 15:39:25	4.9 min	virtuser	2015/04/02 15:39:26
local-1428013325947	Spark shell	2015/04/02 15:22:04	2015/04/02 15:34:23	12 min	virtuser	2015/04/02 15:34:24
local-1428013146932	PythonPi	2015/04/02 15:19:05	2015/04/02 15:19:08	3 s	virtuser	2015/04/02 15:19:10
local-1428013023691	WordCount	2015/04/02 15:17:02	2015/04/02 15:17:05	3 s	virtuser	2015/04/02 15:17:07
local-1428011545168	Spark shell	2015/04/02 14:52:23	2015/04/02 15:11:46	19 min	virtuser	2015/04/02 15:11:47
local-1428006794414	PySparkShell	2015/04/02 13:33:12	2015/04/02 15:10:05	1.6 h	virtuser	2015/04/02 15:10:06
local-1428012098375	Spark Pi	2015/04/02 15:01:36	2015/04/02 15:01:39	4 s	virtuser	2015/04/02 15:01:41
local-1428007889665	Spark shell	2015/04/02 13:51:28	2015/04/02 14:52:14	1.0 h	virtuser	2015/04/02 14:52:16

The history server lists the applications that ran recently. Your screenshot will be different from what you see here. The app will only show up after it has finished running.

__3. Let's go through a simple example to see how caching affects the storage. Create a RDD from the README.md file.

```
val readme = sc.textFile("/tmp/README.md")
```

__4. Cache the readme RDD.

```
readme.cache()
```

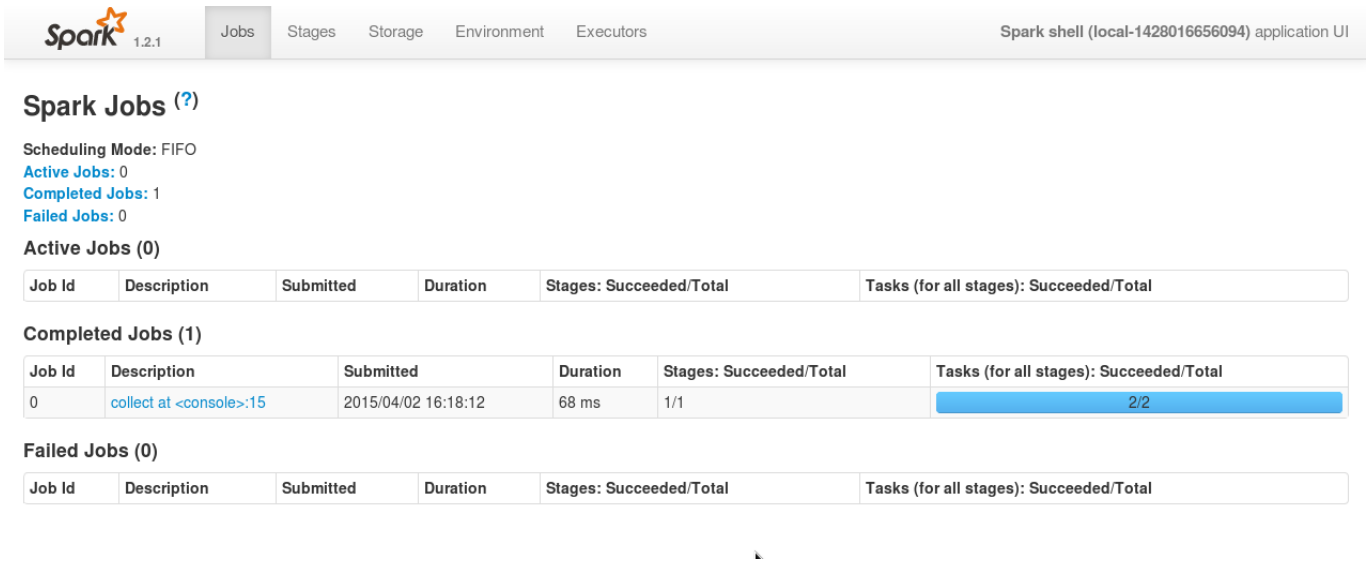
__5. Print the information out onto the console. Remember that transformations do not get applied until some action takes place. This is why we're going printing it out to the console using the collect() action.

```
readme.collect()
```

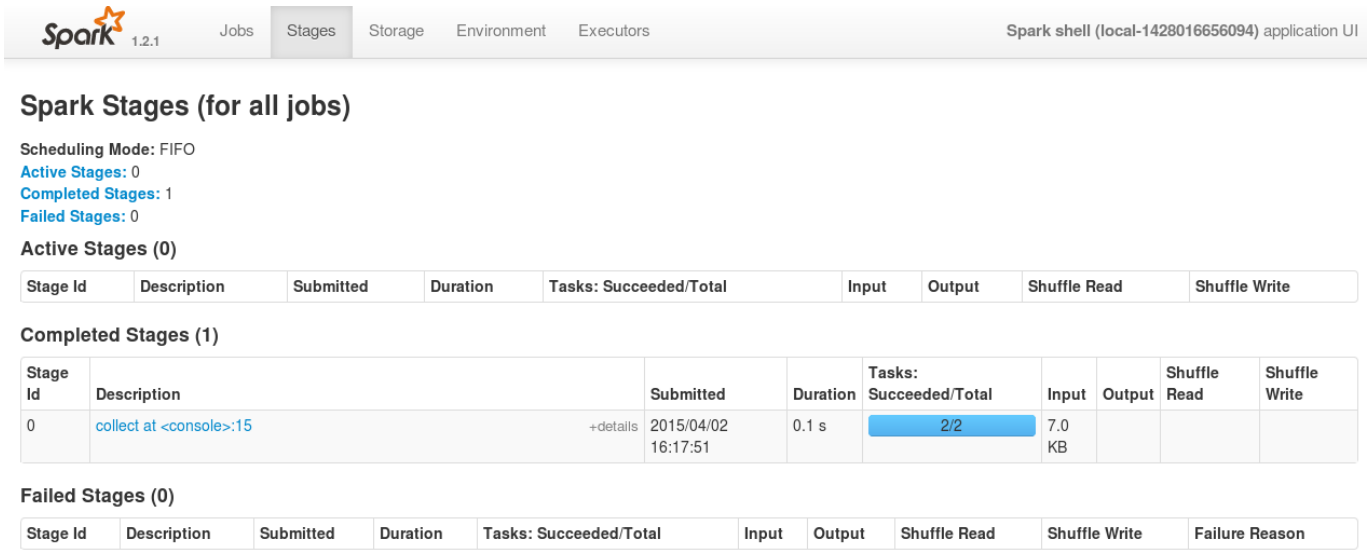
__6. Quit out of the Spark shell so the entry appears in the history server page.

__7. CTRL + D

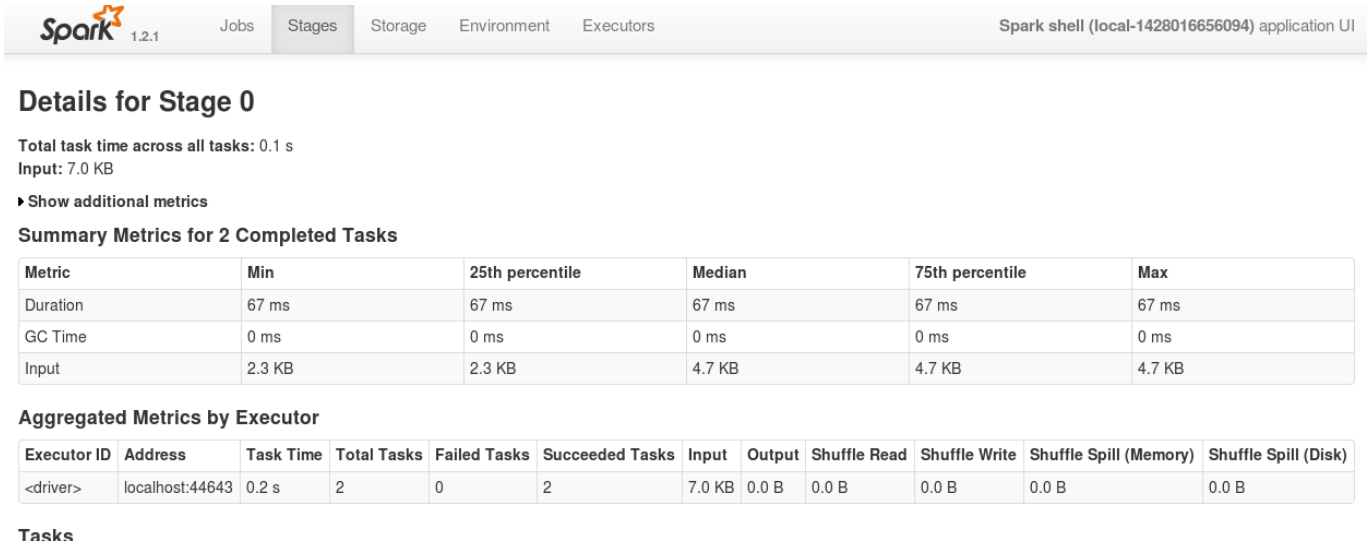
__8. Refresh the history server page to see the new listing of the Spark shell that you just used to run the simple example. Click on the link of that shell for more information.



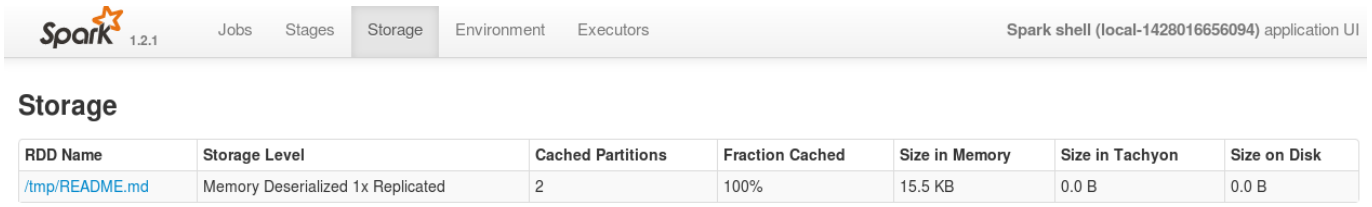
__9. Click the Stages link on the menu:



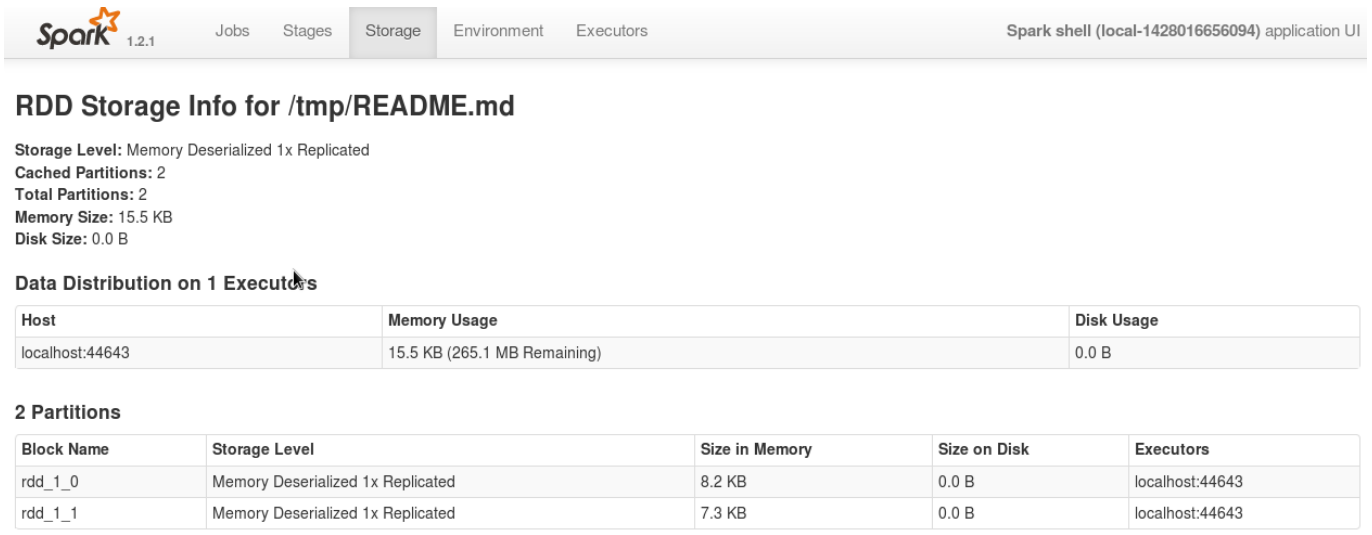
__10. Click on the collect stage to view the details of that run:



__11. There are four sections at the top: Click on the Storage section.



__12. You can see that the README.MD file was cached along with the size in memory and the number of cached partitions. f you click on the link, you can see in more detail which blocks and executors that file is distributed:



A lot of the monitoring tools can be used to help to with tuning. For example, the previous task that you just did to cache a RDD is actually a good method to use for figuring out how much memory each partition is consuming.

Summary

Having completed this exercise, you should have a basic understanding of how to specify various Spark configurations through the SparkConf object, as parameters through the spark-submit or spark-shell commands, or by specifying defaults in the spark-defaults.conf file. You also launched the history server Web UI to check out the Spark application that you ran.

NOTES

NOTES

IBM Software