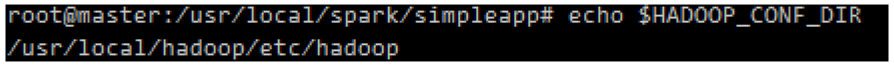
Spark Tutorial

Jinlai Xu

Preparation

- Check HDFS and YARN services and environment configuration
  - jps
  - echo $HADOOP_CONF_DIR



```
root@master:/usr/local/spark/simpleapp# echo $HADOOP_CONF_DIR
/usr/local/hadoop/etc/hadoop
```

- Download Spark package
  - wget http://apache.mirrors.pair.com/spark/spark-2.3.0/spark-2.3.0-bin-hadoop2.7.tgz
- Unpack
  - tar zxvf spark-2.3.0-bin-hadoop2.7.tgz
- Try Spark shell (local mode)
  - ./bin/spark-shell --master local[2]

```
val NUM_SAMPLES = 1000
val count = sc.parallelize(1 to NUM_SAMPLES).filter { _ =>
  val x = math.random
  val y = math.random
  x*x + y*y < 1
}.count()
println(s"Pi is roughly ${4.0 * count / NUM_SAMPLES}")
```

Run Spark Shell with YARN

- ./bin/spark-shell --master yarn --deploy-mode client

```
val NUM_SAMPLES = 1000
val count = sc.parallelize(1 to NUM_SAMPLES).filter { _ =>
  val x = math.random
  val y = math.random
  x*x + y*y < 1
}.count()
println(s"Pi is roughly ${4.0 * count / NUM_SAMPLES}")
```

Run Spark example program with YARN

-
  ```
  ./bin/spark-submit --class org.apache.spark.examples.SparkPi \
   --master yarn \
   --deploy-mode cluster \
   --driver-memory 512m \
   --executor-memory 512m \
   --executor-cores 1 \
   --queue default \
   examples/jars/spark-examples*.jar \
   10
  ```

Package and Run your Spark JAVA program

- Directory Structure:
  - pom.xml
  - src/main/java/your_program.java
- pom.xml

```
<project>
  <groupId>edu.berkeley</groupId>
  <artifactId>simple-project</artifactId>
  <modelVersion>4.0.0</modelVersion>
  <name>Simple Project</name>
  <packaging>jar</packaging>
  <version>1.0</version>
  <dependencies>
    <dependency> <!-- Spark dependency -->
      <groupId>org.apache.spark</groupId>
      <artifactId>spark-core_2.11</artifactId>
      <version>2.3.0</version>
    </dependency>
  </dependencies>
</project>
```

Package and Run your Spark JAVA program

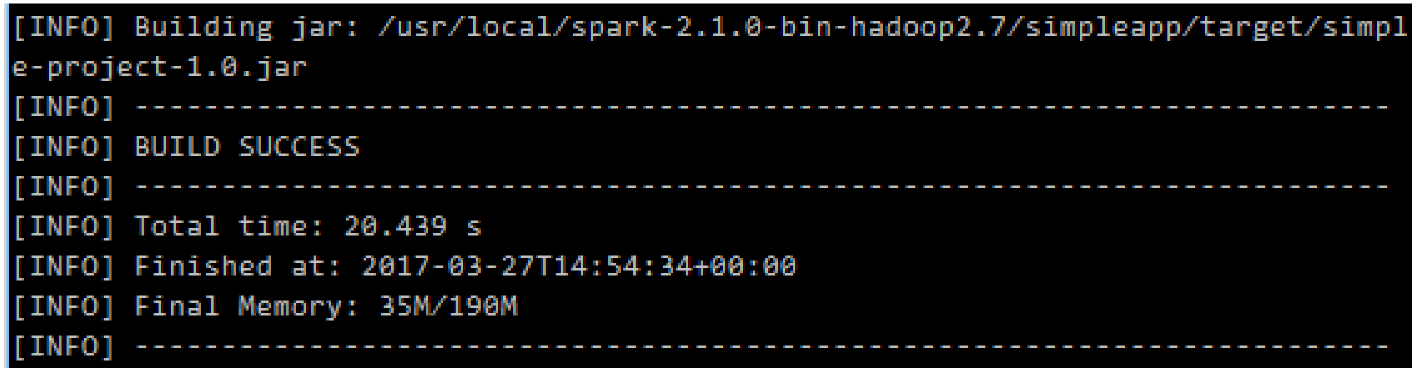
- SimpleApp.java

```java
/* SimpleApp.java */
import org.apache.spark.api.java.*;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.function.Function;

public class SimpleApp {
  public static void main(String[] args) {
    String logFile = "YOUR_SPARK_HOME/README.md"; // Should be some file on your system
    SparkConf conf = new SparkConf().setAppName("Simple Application");
    JavaSparkContext sc = new JavaSparkContext(conf);
    JavaRDD<String> logData = sc.textFile(logFile).cache();
    long numAs = logData.filter(new Function<String, Boolean>() {
      public Boolean call(String s) { return s.contains("a"); }
    }).count();
    long numBs = logData.filter(new Function<String, Boolean>() {
      public Boolean call(String s) { return s.contains("b"); }
    }).count();
    System.out.println("Lines with a: " + numAs + ", lines with b: " + numBs);
    sc.stop();
  }
}
```

Package and Run your Spark JAVA program

- Use maven to package the program:
  - Make sure you install maven on the client:
    - sudo apt-get install maven
  - Package the program:
    - mvn package



```
[INFO] Building jar: /usr/local/spark-2.1.0-bin-hadoop2.7/simpleapp/target/simpl
e-project-1.0.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 20.439 s
[INFO] Finished at: 2017-03-27T14:54:34+00:00
[INFO] Final Memory: 35M/190M
[INFO] ------------------------------------------------------------------------
```

Package and Run your Spark JAVA program

- Directory Structure:
  - pom.xml
  - src/main/java/your_program.java
  - target/your_program*.jar
- Run the program with spark-submit
  - ../bin/spark-submit --class "SimpleApp" \

    --master yarn \

    --deploy-mode cluster \

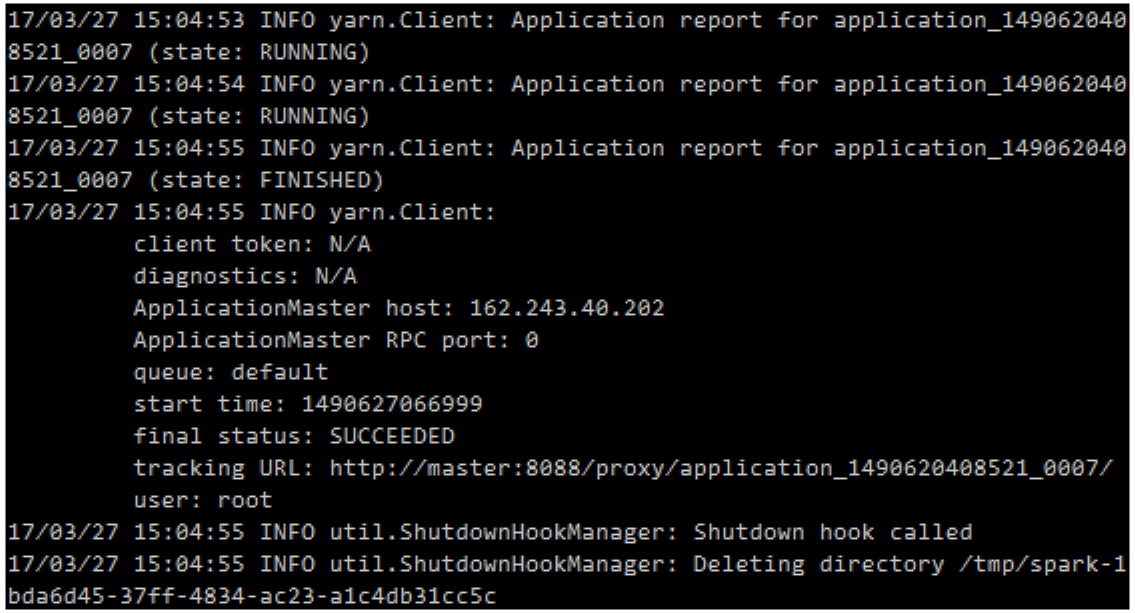
    --driver-memory 1g \

    --executor-memory 1g \

    --executor-cores 1 \

    --queue default \

    target/simple*.jar



```
17/03/27 15:04:53 INFO yarn.Client: Application report for application_149062040
8521_0007 (state: RUNNING)
17/03/27 15:04:54 INFO yarn.Client: Application report for application_149062040
8521_0007 (state: RUNNING)
17/03/27 15:04:55 INFO yarn.Client: Application report for application_149062040
8521_0007 (state: FINISHED)
17/03/27 15:04:55 INFO yarn.Client:
         client token: N/A
         diagnostics: N/A
         ApplicationMaster host: 162.243.40.202
         ApplicationMaster RPC port: 0
         queue: default
         start time: 1490627066999
         final status: SUCCEEDED
         tracking URL: http://master:8088/proxy/application_1490620408521_0007/
         user: root
17/03/27 15:04:55 INFO util.ShutdownHookManager: Shutdown hook called
17/03/27 15:04:55 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-1
bda6d45-37ff-4834-ac23-a1c4db31cc5c
```

Q & A

FAQ

- The compatibility of Spark, Hadoop and JAVA
  - From Spark Official website:
  - Spark runs on Java 7+, Python 2.6+/3.4+ and R 3.1+. For the Scala API, Spark 2.1.0 uses Scala 2.11. You will need to use a compatible Scala version (2.11.x).
  - Note that support for Java 7 and Python 2.6 are deprecated (do not mean unsupported) as of Spark 2.0.0, and support for Scala 2.10 and versions of Hadoop before 2.6 are deprecated as of Spark 2.1.0, and may be removed in Spark 2.2.0.
- The comparison in Part 3
  - Required: compare the performance of using the cached RDD feature and without using the cached RDD feature.
  - Alternative: compare the performance of above two with the same workload of using Hadoop MapReduce

FAQ

- Do I need to do any configuration for Spark
  - No, you do not. Because if you indicate to use YARN to run spark, it will automatically use the environment parameter “HADOOP_CONF_DIR” to find the Hadoop configuration files.