Spark Tutorial

Jinlai Xu

Preparation

- Download Spark package
  - wget https://mirror.olnevhost.net/pub/apache/spark/spark-2.4.5/spark-2.4.5-bin-hadoop2.7.tgz
- Unpack
  - tar zxvf spark-2.4.5-bin-hadoop2.7.tgz
  - ln -s spark-2.4.5-bin-hadoop2.7 spark
- Install scala: sudo apt install scala
- 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

- Check HDFS and YARN services and environment configuration
  - jps
  - Add “export HADOOP_CONF_DIR=/home/student/hadoop/etc/hadoop” to the end of “~/.bashrc”
  - source ~/.bashrc
  - echo $HADOOP_CONF_DIR

```
student@CC-demo-01:~$ echo $HADOOP_CONF_DIR
/home/student/hadoop/etc/hadoop
```

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

```
val NUM_SAMPLES = 1000000
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 client \
  --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
- SimpleApp.java (pom.xml is in the package)

```
/* SimpleApp.java */
import org.apache.spark.sql.SparkSession;
import org.apache.spark.sql.Dataset;

public class SimpleApp {
  public static void main(String[] args) {
    String logFile = "README.md"; // Should be some file on your system
    SparkSession spark = SparkSession.builder().appName("Simple Application").getOrCreate();
    Dataset<String> logData = spark.read().textFile(logFile).cache();

    long numAs = logData.filter(s -> s.contains("a")).count();
    long numBs = logData.filter(s -> s.contains("b")).count();

    System.out.println("Lines with a: " + numAs + ", lines with b: " + numBs);

    spark.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
  - Check the output files
    - find .

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 1.590 s
[INFO] Finished at: 2019-03-06T19:30:29Z
[INFO] Final Memory: 22M/174M
[INFO] ------------------------------------------------------------------------
```

```
student@CC-demo-01:~/workspace$ find .
.
./src
./src/main
./src/main/java
./src/main/java/SimpleApp.java
./pom.xml
./target
./target/maven-archiver
./target/maven-archiver/pom.properties
./target/simple-project-1.0.jar
./target/maven-status
./target/maven-status/maven-compiler-plugin
./target/maven-status/maven-compiler-plugin/compile
./target/maven-status/maven-compiler-plugin/compile/default-compile
./target/maven-status/maven-compiler-plugin/compile/default-compile/inputFiles.lst
./target/maven-status/maven-compiler-plugin/compile/default-compile/createdFiles.lst
./target/classes
./target/classes/SimpleApp.class
./target/generated-sources
./target/generated-sources/annotations
```

Package and Run your Spark JAVA program

- Directory Structure:
  - pom.xml
  - src/main/java/your_program.java
  - target/your_program*.jar
- Put the example file into HDFS
  - hdfs dfs -put ~/spark/README.md .
- Run the program with spark-submit
  - ~/spark/bin/spark-submit --class "SimpleApp" \

    --master yarn \

    --deploy-mode client\

    --driver-memory 1g \

    --executor-memory 1g \

    --executor-cores 1 \

    --queue default \

    target/simple*.jar

```
2019-03-06 19:46:41 INFO  Client:54 -
        client token: N/A
        diagnostics: N/A
        ApplicationMaster host: CC-demo-02
        ApplicationMaster RPC port: 36045
        queue: default
        start time: 1551901581079
        final status: SUCCEEDED
        tracking URL: http://CC-demo-01:8088/proxy/application_1551898883682_0006/
        user: student
2019-03-06 19:46:41 INFO  Client:54 - Deleted staging directory hdfs://CC-demo-01:9000/user/student/.sparkStaging/applic
6
2019-03-06 19:46:41 INFO  ShutdownHookManager:54 - Shutdown hook called
2019-03-06 19:46:41 INFO  ShutdownHookManager:54 - Deleting directory /tmp/spark-d70a84c3-0005-49ca-9e75-f2065d08d9ad
2019-03-06 19:46:41 INFO  ShutdownHookManager:54 - Deleting directory /tmp/spark-1da29020-0949-4d07-88a9-7d11dd31b49f
```

Q & A

- How to test the program locally:
- ~/spark/bin/spark-submit --class "SimpleApp" \

```
--master local \
 --driver-memory 1g \
 --executor-memory 1g \
--executor-cores 1 \
--queue default \
target/simple*.jar
```

FAQ

- The compatibility of Spark, Hadoop and JAVA
  - From Spark Official website:
  - Spark runs on Java 8+, Python 2.7+/3.4+ and R 3.1+. For the Scala API, Spark 2.4.5 uses Scala 2.11. You will need to use a compatible Scala version (2.11.x).

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.

FAQ

- How can I run the local example in part3?
- Install Anaconda: https://www.anaconda.com/distribution/
- Download the example code (LinearRegressionExample.ipynb) and the input file (access_log) into your laptop
  - If you use terminal, run the command “jupyter notebook” in the above directory which contains the two files
  - If you use Windows or Mac, you can directly run the “Jupyter Notebook (Anaconda3)” in your Start Menu, then use “upload” to upload the above two files on the website
- Click LinearRegressionExample.ipynb on the website
- Click “Run” to run each block on the website. For some blocks, it may need several seconds to print out the results