Unit Three Apache Spark Applications

**Spark Application Types**

Table 1 Application Domain and Context

|  |  |  |  |
| --- | --- | --- | --- |
| Sample Application | **Domain** | Component | Context |
| SimpleGraphX | **Visualisation** | GraphX, there is no Java API at this time, we will construct our own scala application | analysis, and iterative graph computation |
| JavaKMeans | **Machine learning** | MLlib | Spark’s scalable library of machine learning algorithms |
| TwitterStreamProcessor | **Processing Live Data** | Streaming, we will construct our own simple Twitter Streaming  application | Spark’s scalable library for processing of live data streams |
| JavaSparkSQL | **Relational Queries** | Spark SQL | Spark’s implementation of SQL. |

**Learning Objective 1**

To be able to describe Apache Spark Applications in a given context

**Learning Objective 2**

To be able to configure Apache Spark Applications for a given context.

**Learning Objective 3**

To be able to test, package, deploy and run Apache Spark

SimpleGraphX application

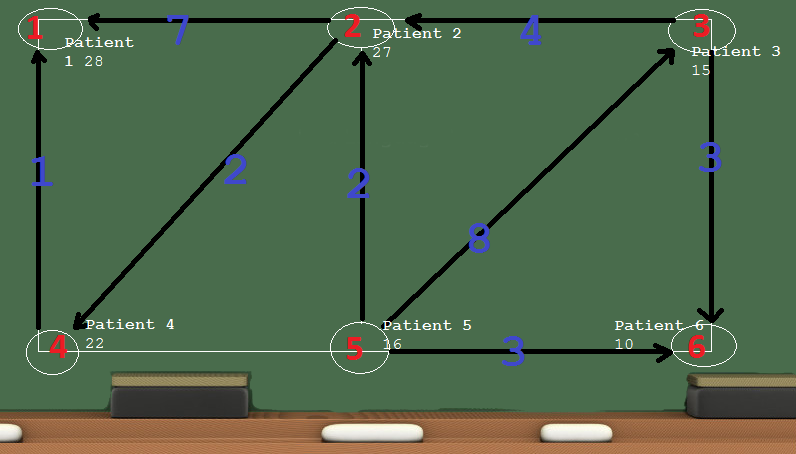
1. Construct a simple graph

Data is from a disease epidemic, Patient , days sick, exposure

|  |  |  |
| --- | --- | --- |
| patient id | days sick | exposure (id, days) |
| 1 | 28 | (4,1),(2,7) |
| 2 | 27 | (5,2),(3,4) |
| 3 | 15 | (5,8) |
| 4 | 22 | (2,2) |
| 5 | 16 |  |
| 6 | 10 | (5,3),(3,3) |

Property Graph

Query is to track the source of the epidemic

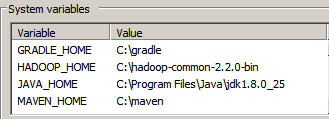


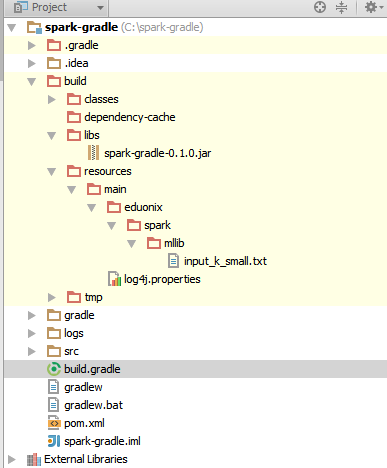
**Spark with Gradle**

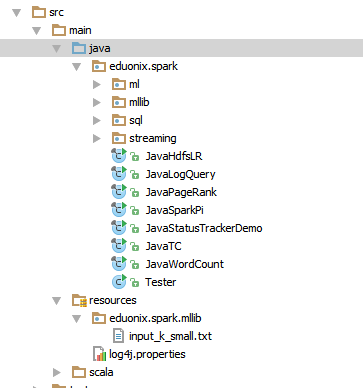
JavaKMeans Application

1. Construct a gradle Spark Projet
2. Develop a Spark Mlib application
3. Cretae a uber jar with gradle
4. run on cluster

1

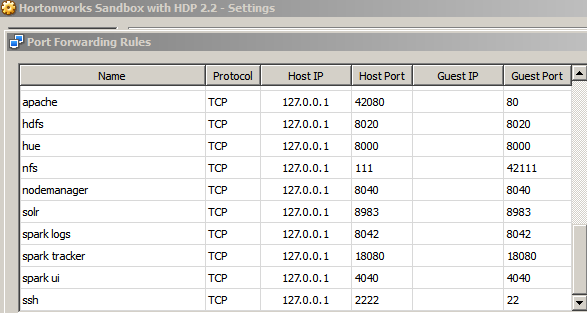






**Run on cluster**

1. set environment just as before
2. export YARN\_CONF\_DIR=/etc/hadoop/conf



b) ensure you have enabled port forwarding for port 8042

c) set up hosts file

**127.0.0.1 sandbox.hortonworks.com**

d) if you have java 8 the ensure you are compiling at java 7 source level

build.gradle

**sourceCompatibility = 1.7**

set your source level in wharver IDE you use as well

edit the **static boolean** *runOnCluster* = **true**;

in the JavaKMeans main class

e) delete the uber jar, build the project and run the fatjar plugin as before

2. create the directories and copy the input data into hdfs

hadoop fs -mkdir /tmp/data

hadoop fs ­-put input\_k\_small.txt /tmp/data

hadoop fs ­-ls /tmp/data

**./bin/spark-submit --class eduonix.spark.mllib.JavaKMeans --master yarn-cluster --num-executors 3 --driver-memory 512m --executor-memory 512m --executor-cores 1 ../spark-gradle-0.1.0.jar hdfs://sandbox.hortonworks.com:8020/tmp/data/input\_k\_small.txt 10 10 1**

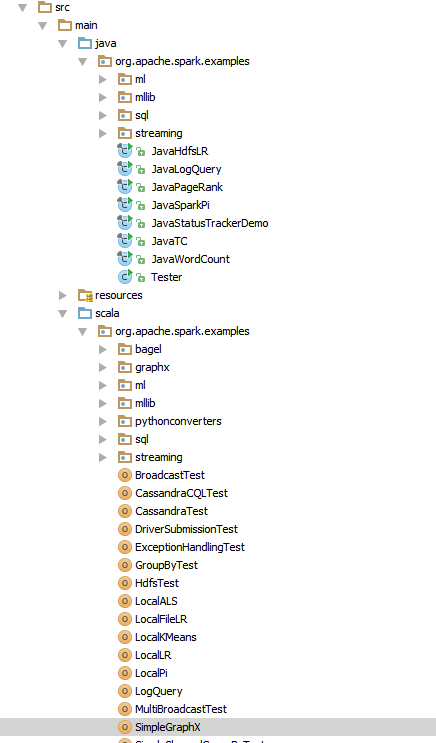
**JavaSparkSQL Application**

1. Develop a Spark SQL Java application in standalone mode
2. Run a Spark Hive User Defined Function (UDF) query from the Spark shell on Yarn

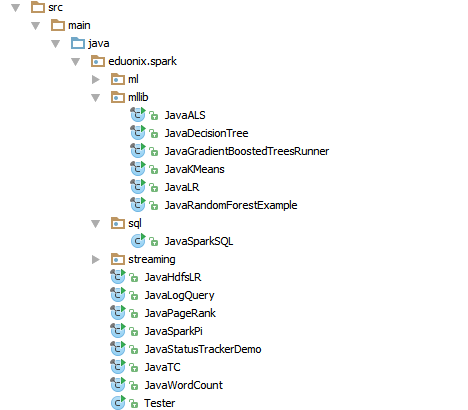
Code Review

1 Eduonix Spark source code

**JavaSparkPi SimpleGraphX**

2. Spark-gradle source code

**JavaKMeans JavaSparkSQL**



Summary Main points

1. Create 2 contexts Spark Context and SparkSQL context
2. Create initial RDD’s then transform to Schema RDD’s

with the SparkSQL context

JavaSchemaRDD teenagers3 = sqlCtx.sql(**"SELECT ……………….**

JavaSchemaRDD peopleFromJsonRDD = sqlCtx.jsonRDD( …………

1. For Schema RDD’s
2. register as table for SQL operations

peopleFromJsonRDD.registerTempTable

1. persist to file system as a Parquet file

JavaSchemaRDD parquetFile = sqlCtx.parquetFile

SparkSQL queries on Hive

1 hive-site.xml

**<configuration>  
<property>  
 <name>hive.metastore.uris</name>  
 <value>thrift://sandbox.hortonworks.com:9083</value>  
 <description>URI for client to contact metastore server</description>  
</property>  
</configuration>**

2. Launch shell

./bin/spark-shell --num-executors 2 --executor-memory 512m --master yarn-client

3. Create Hive interface via a Hive-Spark context

**val hiveContext = new org.apache.spark.sql.hive.HiveContext(sc)**

**hiveContext.hql("CREATE TABLE IF NOT EXISTS PEOPLE (name STRING, age INT)")**

**hiveContext.hql("LOAD DATA LOCAL INPATH 'examples/src/main/resources/people.txt' INTO TABLE PEOPLE ")**

**hiveContext.hql("FROM PEOPLE SELECT \*").collect().foreach(println)**