SPA Conference 2016

Real-World Big Data in Action

Nick Rozanski
Eoin Woods
Chris Cooper-Bland

Number 2 in an occasional series

Prerequisites

Install Java Runtime

- http://www.oracle.com/technetwork/java/javase/downloads/index.html
- For Mac El Capitain, these commands might also work:

```
$ brew tap caskroom/cask
$ brew unlink brew-cask
$ brew cask install java
```

- *I'm not entirely sure about this, I already have the JRE…*

Install Python 2.7

- https://www.python.org/downloads/
- you don't need to know Python programming for this session

Install Cygwin (Windows only)

- Provides a BASH shell to run scripts (not programs)
- Download from https://cygwin.com/install.html
- Or use the Windows 10 Linux subsystem (I've not tried this!)

Create Your Project Directory

Start a BASH Shell

- Windows: Start → Cygwin
- OS X: Applications → Terminal
- Linux: Dash → More Apps → Accessories → Terminal

Create the Project Subdirectories

```
$ mkdir -p $HOME/SPA_2016/hadoop
$ mkdir -p $HOME/SPA_2016/spark
$ mkdir -p $HOME/SPA_2016/hive
$ mkdir -p $HOME/SPA_2016/data
$ mkdir -p $HOME/SPA_2016/logs
```

Install the Big Data Software

Download Hadoop into `$HOME/SPA_2016/hadoop`

- https://www.apache.org/dyn/closer.cgi/hadoop/common/
- Select stable source, download and extract the binary tarball
- You should end up with directories `spark/bin`, `etc`, `logs`, `sbin`…
- (could also use `brew` for Mac, apt for Linux, but we won't for this demo)

Download Spark into `$HOME/SPA_2016/spark`

- https://spark.apache.org/downloads.html
- Choose the package type "pre-built for Hadoop 2.6 and later"
- Download and extract the binary tarball
- You should end up with directories `hadoop/bin`, `conf`, `logs`, `sbin`…

Download Hive into `$HOME/SPA_2016/hive`

- https://www.apache.org/dyn/closer.cgi/hive/
- Download and extract the latest binary tarball
- You should end up with directories `bin`, `conf`…

Real-World Big Data in Action

Hadoop

A Big Data Virtual Filesystem

Set Environment Variables

Create $HOME/SPA_2016/env.src

```
# set JAVA_HOME
# see next slide

export SPA_2016=$HOME/SPA_2016
# Hadoop configuration
export HADOOP_PREFIX=$SPA_2016/hadoop
export HADOOP_CONF_DIR="$HADOOP_PREFIX/etc/hadoop"
export HADOOP_HOME="$HADOOP_PREFIX"
export HADOOP_COMMON_HOME="$HADOOP_PREFIX"
export HADOOP_HDFS_HOME="$HADOOP_PREFIX"
export HADOOP_YARN_HOME="$HADOOP_PREFIX"
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"

# Spark home
export SPARK_HOME=$SPA_2016/spark

# Hive home
export HIVE_HOME=$SPA_2016/hive
```

Setting $JAVA_HOME in env.src

Windows / Cygwin

- *Don't know what this should be…*

Linux

- *I'm going to guess it's* `export JAVA_HOME=/usr/lib/jvm/java-<version>`

Mac OS X

- Run this command to find JAVA_HOME:
  `/usr/libexec/java_home`
- It is likely to be:
  `export JAVA_HOME=/Library/Java/JavaVirtualMachines/Current.jdk/Contents/Home`
- Or you can just do:
  `export JAVA_HOME="$(/usr/libexec/java_home)"`
- You may also have to run the commands:
  `cd/Library/Java/JavaVirtualMachines`
  `sudo ln -s jdk1.8.0_nn.jdk Current.jdk`

Configure Hadoop Core Settings

Edit `$HADOOP_PREFIX/etc/hadoop/core-site.xml`

- add the following lines:

```
<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://localhost:9000</value>
        <description>NameNode URI</description>
    </property>
    <property>
        <name>hadoop.proxyuser.nick.hosts</name>
        <value>*</value>
    </property>
    <property>
        <name>hadoop.proxyuser.nick.groups</name>
        <value>*</value>
    </property>
</configuration>
```

- used by the Hadoop client to access the Hadoop filesystem
- the `proxyuser` item is for accessing Hive using beeline

Configure Hadoop Site Settings

Edit `$HADOOP_PREFIX/etc/hadoop/hdfs-site.xml`

- change **/YOURHOME** to your home directory (eg `/Users/nick`)
- for Windows, use the Cygwin path, not the Windows path

```
<configuration>
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>file:///YOURHOME/SPA_2016/data/hadoop/hdfs/datanode</value>
        <description>
            Paths on the local filesystem where the DataNode stores its blocks
        </description>
    </property>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>file:///YOURHOME/SPA_2016/data/hadoop/hdfs/namenode</value>
        <description>
            Path on the local filesystem where the NameNode stores the namespace
and transaction logs
        </description>
    </property>
</configuration>
```

- tells Hadoop where to put the physical operating system files for the datanode and namenode

Initialise Hadoop Filesystem

Format the HDFS Filesystem

- Equivalent to formatting a n operating system filesystem partition
- Warning: this destroys all HDFS data!
- Do this before starting Hadoop

```
$ source $HOME/SPA_2016/env.src
$ $HADOOP_PREFIX/bin/hdfs namenode -format
```

Check it has worked

```
$ ls $SPA_2016/data/hadoop/hdfs/namenode/current
```

- These are the Hadoop physical operating system files

Start (and Stop) Hadoop Server

Start Hadoop

```
$ source $HOME/SPA_2016/env.src
$ $HADOOP_PREFIX/sbin/hadoop-daemon.sh start namenode
$ $HADOOP_PREFIX/sbin/hadoop-daemon.sh start secondarynamenode
$ $HADOOP_PREFIX/sbin/hadoop-daemon.sh start datanode
# can also do $HADOOP_PREFIX/sbin/start-dfs.sh
```

Stop Hadoop

```
$ source $HOME/SPA_2016/env.src
$ $HADOOP_PREFIX/sbin/stop-dfs.sh
```

- You may be asked for your password (the scripts use SSH)
- Ignore messages like "Unable to load native-hadoop library for your platform"

Check Hadoop is Running

Check Running Processes

```
$ jps | sort -k 2
nnnnn DataNode                ← Hadoop datanode
nnnnn NameNode                ← Hadoop namenode
nnnnn SecondaryNameNode       ← Hadoop secondary namenode
```

Check Log Files

- `$SPA_2016/hadoop/logs/hadoop-<user>-namenode-<hostname>.out`
- `$SPA_2016/hadoop/logs/hadoop-<user>-datanode-<hostname>.out`
- `$SPA_2016/hadoop/logs/hadoop-<user>-secondarynamenode-<hostname>.out`

Check Web Interfaces

- Hadoop Web UI http://localhost:50070
- Utilities → Browse Hadoop Filesystem (http://localhost:50070/explorer.html#)

Hadoop Command Line

Hadoop Command Line

- many Unix shell file manipulation commands (`ls`, `mkdir`, `rm` etc) have Hadoop equivalents using `hadoop fs -<command>`
- for example: `$HADOOP_PREFIX/bin/hadoop fs -ls /user`
- see https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/FileSystemShell.html

Create Your User Directories on Hadoop

```
$ source $HOME/SPA_2016/env.src
$ $HADOOP_PREFIX/bin/hadoop fs -mkdir -p /user/nick/load/lfb
$ $HADOOP_PREFIX/bin/hadoop fs -mkdir -p /user/nick/load/lhp
$ $HADOOP_PREFIX/bin/hadoop fs -chown -R nick /user/nick
# user identities map 1-1 from the O/S
```

- replace **nick** with your operating system user name

Exercise 1: Load Some Data into Hadoop

London Fire Brigade Reported Incidents

- Original from http://data.london.gov.uk/dataset/london-fire-brigade-incident-records
- Covers the period 2013 - 2016
- Loaded into Excel and converted into Windows CSV file

Download the data file `LFB.csv` into $SPA_2016/datasets/LFB

- see https://github.com/rozanski/bcs_spa16/tree/master/datasets/LFB

Load the data into Hadoop

- long command split over three lines:

```
$ $HADOOP_HOME/bin/hadoop fs -put \
   $SPA_2016/datasets/LFB/LFB.csv \
   hdfs://localhost:9000/user/nick/load
```

Exercise 1 continued (2 of 2)

Browse Hadoop from the Command Line

```
$ $HADOOP_HOME/bin/hadoop fs -ls /user/nick/load
Found 1 items
-rw-r--r--   3 nick supergroup   79888721 2016-05-29 10:53 /user/nick/load/
LFB.csv
```

Browse Hadoop from your Web Browser

- http://localhost:50070/explorer.html
- go to `/user/nick/load`

Real-World Big Data in Action

Spark

A Big Data Processing Engine

Configure Spark for CSV Files

Download Spark CSV Support

- Download `spark-csv` from https://spark-packages.org/package/databricks/spark-csv
- Save the latest JAR into `$SPA_2016/spark/lib`

Edit `$HOME/SPA_2016/spark/conf/spark-defaults.conf`

- Copy the file from `spark-defaults.conf.template`
- add the line:

```
spark.jars.packages com.databricks:spark-csv_2.11:1.4.0
```

- make sure the version (11.1.4.0) matches the version of the JAR you downloaded!

Edit `$HOME/SPA_2016/spark/conf/spark-env.sh`

- add the lines:

```
export HADOOP_CONF_DIR=/Users/nick/SPA_2016/hadoop/etc/hadoop
export SPARK_LOCAL_DIRS=/Users/nick/SPA_2016/data/spark
```

Start (and Stop) Spark Server

Start Hadoop if not already running

```
$ source $HOME/SPA_2016/env.src
$ $HADOOP_PREFIX/sbin/hadoop-daemon.sh start namenode
$ $HADOOP_PREFIX/sbin/hadoop-daemon.sh start secondarynamenode
$ $HADOOP_PREFIX/sbin/hadoop-daemon.sh start datanode
# can also do $HADOOP_PREFIX/sbin/start-dfs.sh
```

Start Spark Server

```
$ source $HOME/SPA_2016/env.src
$ $SPARK_HOME/sbin/start-master.sh
$ $SPARK_HOME/sbin/start-slaves.sh spark://hostname:7077
```

- where **hostname** is the host name (or IP address) of your computer

Stop Spark Server

```
$ source $HOME/SPA_2016/env.src
$ $SPARK_HOME/sbin/stop-all.sh
```

Check Hadoop and Spark Are Running

Check Running Processes

```
$ jps | sort -k 2
nnnnn DataNode                ← Hadoop datanode
nnnnn Master                  ← Spark master
nnnnn NameNode                ← Hadoop namenode
nnnnn SecondaryNameNode       ← Hadoop secondary namenode
nnnnn Worker                  ← Spark slave
```

Check Log Files

- $SPA_2016/hadoop/logs/hadoop-*<user>*-namenode-*<hostname>*.out
- $SPA_2016/hadoop/logs/hadoop-*<user>*-datanode-*<hostname>*.out
- $SPA_2016/hadoop/logs/hadoop-*<user>*-secondarynamenode-*<hostname>*.out

Check Web Interfaces

- Hadoop Web UI http://localhost:50070
- Browse Hadoop Filesystem http://localhost:50070/explorer.html#
- Spark Web UI http://localhost:8080

Pyspark

- Pyspark allows you to submit Spark commands from a Python shell
- Start Hadoop and Spark
- Start Pyspark:

```
$ $SPARK_HOME/bin/pyspark
```

- You should get the message:

```
SparkContext available as sc, HiveContext available as sqlContext.
```

- You can run any Python command at this point
- You can also call functions in the pyspark.sql library

```
$ help(sqlContext)
```

Exercise 2: Let's Do Some Data Science!

Load the LFB Data into Spark

- Enter the following command at the Pyspark prompt (*on one line, split here for readability*)

```
>>> lfb = sqlContext.read.format('com.databricks.spark.csv').
      option('header', 'true').option('inferschema', 'true').
      option('mode', 'DROPMALFORMED').load('/user/nick/load/LFB.csv')
```

Check it's Loaded

```
>>> print lfb.count()
...
322217
```

Display the data column names

```
>>> lfb.printSchema()
```

Look at Some Data

```
>>> lfb.filter(lfb.IncidentGroup == 'Special Service').limit(5).show()
```

Exercise 2 continued (2 of 2)

Incident Counts by Type

```
>>> lfb.groupBy('IncidentGroup').count().show()
```

Incident Counts by Stop Code

```
>>> lfb.groupBy('StopCodeDescription').count().show(truncate=False)
```

Most Dangerous Areas

```
>>> lfb.groupBy('Postcode_district').count(). \
        sort('count', ascending=False).show()
```

And What Happens There

```
>>> lfb.rollup('IncidentGroup','Postcode_district'). \
        count().sort('count', ascending=False).show()
```

“Frequent” Problem Areas

```
>>> for borough in sorted(lfb.freqItems(['IncGeo_BoroughName']). \
         first()[0]):
     print borough
```

Hive

A Big Data data warehousing infrastructure

Configure Hive (1 of 2)

Edit $HOME/SPA_2016/hive/conf/hive-site.xml

```
<configuration>
  <property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:derby:;databaseName=/Users/nick/SPA_2016/data/hive/
metastore_db;create=true</value>
    <description>JDBC connect string for a JDBC metastore</description>
  </property>
  <property>
    <name>hive.execution.engine</name>
    <value>spark</value>
    <description>
    Expects one of [mr, tez, spark].
    Chooses execution engine. Options are: mr (Map reduce, default), tez, spark.
While MR
    remains the default engine for historical reasons, it is itself a historical
engine
    and is deprecated in Hive 2 line. It may be removed without further warning.
    </description>
  </property>
<configuration>
```

Configure Hive (2 of 2)

Create metastore database

```
mkdir $SPA_2016/data/hive
cd $SPA_2016/data/hive
$HIVE_HOME/bin/schematool -initSchema -dbType derby
```

- You should have a directory `$SPA_2016/data/hive/metastore_db/`

Provide Spark configuration file to Hive

- Copy the Spark configuration file:
  `$SPA_2016/spark/conf/spark-defaults.conf`
  to:
  `$SPA_2016/hive/conf/spark-defaults.conf`

Start (and Stop) Hive Server

Start Hadoop if not already running

- see earlier slide

Start Spark Server if not already running

- see earlier slide

Start Hive Server

```
$ source $HOME/SPA_2016/env.src
$ nohup ./hive --service hiveserver2 2>&1 > /dev/null &
```

Stop Hive Server

```
$ killall HiveServer2
```

Beeline

- Beeline allows you to run Hive SQL queries from a command shell
- Start Hadoop, Spark and Hive if not already running
- Start Beeline:

```
$SPARK_HOME/bin/beeline -u jdbc:hive2://--color
```

- Beeline commands can span multiple lines and are terminated by a semicolon `;`
- Check your Hive databases

```
0: jdbc:hive2://> SHOW DATABASES;
+----------------+--+
| database_name  |
+----------------+--+
| default        |
+----------------+--+
```

- You have an empty Hive installation
- Exit Beeline by typing `!quit` at the prompt

Exercise 3: Data Science Using Hive

Load the LFB Data into Hive

- Run this script, which creates a Hive database called `spa_2016` and loads a table called `lfb_data`

```
$SPARK_HOME/bin/beeline -u jdbc:hive2:// --color < \
      datasets/LFB/load_external.hive
```

Check It's Loaded

- Run these commands in beeline

```
$SPARK_HOME/bin/beeline -u jdbc:hive2://
0: jdbc:hive2://> use spa_2016;
0: jdbc:hive2://> select count(*) from lfb_data;
0: jdbc:hive2://> describe lfb_data;
```

- The table should contain 322,217 rows

Exercise 3 Continued (2 of 2)

Incident Counts by Type

```
//> select incidentgroup, count(*) from lfb_data group by incidentgroup;
```

Incident Counts by Stop Code

```
//> select stopcodedescription, count(*) from lfb_data
     group by stopcodedescription;
```

Most Dangerous Areas

```
//> select postcode_district, count(*) as c
     from lfb_data group by postcode_district
     having c> 1000 order by c desc limit 10;
```

And What Happens There

```
//> select postcode_district, incidentgroup, count(*) as c
     from lfb_data group by postcode_district, incidentgroup
     having c> 1000 order by c desc;
```

"Frequent" Problem Areas

- no Hive equivalent to Spark `freqItems`

Next Steps

Some Other Big Data Tools

Spark SQL Cheat Sheet

SQL	Pyspark
`select col1, ... from mutable`	`dataFrame.select(col1, ...)`
`select count(*) from mytable`	`dataFrame.count()`
`select col1, col2, count(*) ... group by...`	`dataFrame.cube(col1, col2, ...)`
`select distinct ...`	`dataFrame.distinct()`
`select ... where ...`	`dataFrame.filter(expression)`
	`dataFrame.groupBy(col1, ...)`
`select ... limit ...`	`dataFrame.limit(n)`
`select ... order by ...`	`dataFrame.orderBy([col1, …], ascending=True|False)`

etc (MORE WORK ON THIS)

use quotes if columns are reserved words

https://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.DataFrame