Build Your Own Data Analytics Cluster

An Intro to the Big Data Stack

Ben Zimmer – Software Development Engineer

3M Company – SEMS Lab – Data Engineering Group

What is Big Data all about?

- New methods are required to move, store, and analyze large amounts of data (TB-PB scale)
- Build powerful systems from commodity hardware rather than specialized appliances
- Many different needs: data warehousing, stream processing, data analytics
- Frameworks and languages often use functional paradigms
- Big Data brings supercomputing to the masses!

An Engineering Problem

- Our task: replicate a scaled-down version of the Google books Ngram Viewer
- How would you go about implementing this? What languages / tools / platforms would you use?

An Engineering Problem

- To build a solution, we will use:
  - VMs in the Microsoft Azure cloud
  - Apache Cassandra
  - Apache Spark
  - R / Shiny

Apache Cassandra

- High-performance distributed database
- Unique ring architecture with no single point of failure
- Tunable consistency
- Data model is particularly suited to this problem

Apache Spark

- Framework for parallel computing
- More flexible than Hadoop
- Scala, Java, and Python APIs
- Interactive shell
- Connector library for Cassandra

R / Shiny

- R is a language for statistics and analytics, but it can do much more!
- Many built-in features that make it easy to manipulate and plot data
- Vector and string processing capabilities make it a great command line tool when working with clusters
- Shiny is an R package that allows easy creation of interactive web apps

Steps

- Prepare VMs
- Install Cassandra
- Install Spark
- Download the data and insert it into Cassandra with Spark
- Connect to Cassandra with R
- Build a user interface using R and Shiny

Prepare VMs - 1

- Create virtual machines using Azure Management Portal
  - Using the OpenLogic image (CentOS)
  - Machines come with a user called “azureuser” that has sudo permissions

- First step is disabling selinux. Log in to each machine and run:

```
setenforce 0
echo "SELINUX=disabled" > /etc/sysconfig/selinux
echo "SELINUXTYPE=targeted" >> /etc/sysconfig/selinux
```

Prepare VMs - 2

- Passwordless SSH
  - When this is configured correctly, you can interact with machines without having to type a password every time.
  - Generate a SSH key pair with `ssh-keygen`, place it in `/root/.ssh` on the master node, then append the contents of `id_rsa.pub` to `/root/.ssh/authorized_keys` on each machine.
  - You can also put `id_rsa` in `/root/.ssh/authorized_keys` on each machine; I usually don't.
  - This is highly sensitive to directory and file permissions! To debug, enable logging in the `sshd` service and watch `/var/log/messages` as you try to connect

Prepare VMs - 3

- Set up /etc/hosts
  - Give nice names to the machines in the cluster
- Choose a utility for running commands in parallel on remote machines
  - It's not difficult to write a shell script to do this (this is what I use)
  - pssh is another option
- Copy the /etc/hosts file to each of the machines

Prepare VMs - 4

- Add EPEL repository

```
wget http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
rpm -Uvh epel-release-6*.rpm
```

- Install R / RStudio Server

```
yum install R
wget http://download2.rstudio.org/rstudio-server-0.98.1062-x86_64.rpm
yum install --nogpgcheck rstudio-server-0.98.1062-x86_64.rpm
```

Prepare VMs - 5

- Install Java JDK
  - Download the latest version of JDK 1.7 from Oracle, place on each machine, then run this:

```
rpm -ivh jdk-7u51-linux-x64.rpm
alternatives --install /usr/bin/java java /usr/java/jdk1.7.0_51/bin/java 20000
alternatives --set java /usr/java/jdk1.7.0_51/bin/java
```

  - This instructs the systems to use the Oracle JDK instead of the OpenJDK JRE; Verify with `java -version`

Install Cassandra - 1

- Downloads
  - On each machine, download `apache-cassandra-2.1.0.tar.gz`, extract and place in `/opt`, add symlink
  - Place it on the master node (even though the master won't be a Cassandra node) for `cqlsh` and `cassandra-cli`

- Configuration
  - Most configuration is done in `conf/cassandra.yaml`
    - Generate tokens and set initial token for each node (http://www.geroba.com/cassandra/cassandra-token-calculator/)
    - Set authenticator / authorizer
    - Set seed node
    - Set listen address and rpc address

Install Cassandra - 2

- Start it up!
  - Start the Cassandra service on the Cassandra nodes

    ```
    bin/cassandra
    ```
  - Check logs and use nodetool to verify ring is communicating correctly

    ```
    bin/nodetool -h cass0.ed ring
    tail -f logs/system.log
    ```
  - Test it out with `bin/cqlsh`

Install Spark - 1

- Downloads
  - On each machine, download `spark-1.0.2-bin-hadoop2.tgz`, extract and place in `/opt`, add symlink
- Configuration
  - Most configuration is done in `conf/spark-env.sh`
    - Set worker `SPARK_WORKER_MEMORY` to reasonable limit
    - Set `MASTER` to Spark URL for master: spark://master.ed:7077
    - Set `SPARK_HOME` to `/opt/spark`
  - Modify `conf/slaves` to contain list of Spark worker machines
  - For easier testing, add Cassandra authentication info to `conf/spark-defaults.conf`

Install Spark - 2

- spark-cassandra-connector

```
cat > download-connector.sh <<EOF
mkdir /opt/connector
cd /opt/connector

rm *.jar

curl -o ivy-2.3.0.jar \
  'http://search.maven.org/remotecontent?filepath=org/apache/ivy/ivy/2.3.0/ivy-2.3.0.jar'
curl -o spark-cassandra-connector_2.10-1.0.0-beta1.jar \
  'http://search.maven.org/remotecontent?filepath=com/datastax/spark/spark-cassandra-connector_2.10/1.0.0-be

ivy () { java -jar ivy-2.3.0.jar -dependency \$* -retrieve "[artifact]-[revision](-[classifier]).[ext]"; }

ivy org.apache.cassandra cassandra-thrift 2.0.9
ivy com.datastax.cassandra cassandra-driver-core 2.0.3
ivy joda-time joda-time 2.3
ivy org.joda joda-convert 1.6

rm -f *-{sources,javadoc}.jar

EOF

sudo bash download-connector.sh
```

  - Use Ivy to download the connector jars and their dependencies from Maven Central

(**http://planetcassandra.org/blog/installing-the-cassandra-spark-oss-stack/)**

Install Spark - 3

- Start it up!
  - On the master, start the Spark Master server and connect workers to it
    `sbin/start-all.sh`
  - Visit `master.cass:8080` to view the Spark web UI
    - Requires setting up an SSH tunnel and adding the host names of the machines to your local hosts file

Data Loading - 1

- Use R to parse the Google ngram data index page

Data Loading - 2

- Use Spark to distribute the download and insertion of each file into Cassandra

Data Connector

- Use RCassandra to query the database

User Interface

- Build a Shiny app to replicate the ngram viewer functionality

Questions?