Real Time Search and Analytics on Big Data - Real Time Demo

Introduction

This demo contains two applications: 1. A Java application that indexes stock data into Solr. (stocks-indexer) 2. An HTML web page that uses JavaScript and AJAX to generate a graph from data in Solr. (query-app) The interesting part about this demo is that we are indexing data in a streaming fashion and querying on it in real time. As the data is being indexed, we query on that data in real time and display it in a graph. Essentially, this demo is meant to replicate ticker like data being indexed and queried on in real time.

Prerequisites

- An Environment with Solr Installed (The instructor should provide a cluster on EC2)
- An Environment with HBase Installed (The instructor should provide a cluster on EC2)

Running the Exercise

The exercise contains two main projects, the stocks-indexer project and the query-app project.

The stocks-indexer project actually contains two MapReduce applications, the first named HBaseStocksLoader and the second named StocksIndexer. We run the HBaseStocksLoader to stage the data and then run the StocksIndexer to index the data in Solr.

The query-app is a simple HTML/AJAX web application named realtimedemo.html that queries Solr on a given interval.

Therefore, we must run the applications in this order: 1) Run HBaseStocksLoader to stage the data 2) Run StocksIndexer to load data in Solr 3) As StocksIndexer is running, use realtimedemo.html to query Solr and view the newly indexed data in real time

Sorting the Stock Data (and why we are using HBase)

If you take a look at the sample data ($PROJECT_HOME/data) you will find the stock data in CSV format sorted by latest date. If we were to write a Java application that took this data in line by line, we would get the latest date first and then fill in the history (which is not what we want).

I resolved this issue in a bit of a roundabout way. Since the Think Big Academy already has an HBase course using this same stock data, I reused one of that code to sort the data in HBase. HBase provides lexicographically sorted rowkeys, so with the rowkey STOCK_SYMBOLIDATE, we get stock data ordered in the way we want it. This probably is not the best reason for incorporating HBase into our architecture, but it did make developing this course much easier for me. And hey, you get two free HBase MapReduce examples out of it!

StocksHBaseLoader Purpose

As mentioned in the previous section we need to load HBase to sort the data. This is a MapReduce application that does just that. It scans over CSV records in HDFS and loads them into HBase.

StocksIndexer Purpose

This application uses MapReduce to scan over records in HBase and index them in Solr in a time-series fashion. This way, if we're querying the data in Solr, we will see 'new' records being populated in real time.

Building the stocks-indexer jar

The real-time-stocks-indexer–1.0-SNAPSHOT.jar is already provided in the directory for this exercise, however if you want to build the jar yourself you can use the following Maven command and should see similar results. The Maven command will create the jar in the ./target directory.

```
$ cd ~/code/ryantabora/RealTimeSearchAndAnalytics/07-real-time-demo/stocks-indexer/
$ ls -l
total 85464
-rw-r--r--  1 ryantabora  staff      2491 Jan 22 16:36 pom.xml
-rw-r--r--  1 ryantabora  staff  43749464 Jan 25 15:44 real-time-stocks-indexer-1.0-SNAPSHOT.jar
drwxr-xr-x  4 ryantabora  staff       136 Jan 22 15:38 src
$ mvn clean package
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Real Time Stocks Indexer 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.4.1:clean (default-clean) @ real-time-stocks-indexer ---
[INFO] Deleting /Users/ryantabora/Code/ryantabora/RealTimeSearchAndAnalytics/07-real-time-demo/stocks-indexer/target
[INFO]
[INFO] --- maven-resources-plugin:2.5:resources (default-resources) @ real-time-stocks-indexer ---
[debug] execute contextualize
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/ryantabora/Code/ryantabora/RealTimeSearchAndAnalytics/07-real-time-demo/stocks-indexer/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:2.3.2:compile (default-compile) @ real-time-stocks-indexer ---
[INFO] Compiling 6 source files to /Users/ryantabora/Code/ryantabora/RealTimeSearchAndAnalytics/07-real-time-demo/stocks-indexer/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.5:testResources (default-testResources) @ real-time-stocks-indexer ---
[debug] execute contextualize
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/ryantabora/Code/ryantabora/RealTimeSearchAndAnalytics/07-real-time-demo/stocks-indexer/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:2.3.2:testCompile (default-testCompile) @ real-time-stocks-indexer ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.10:test (default-test) @ real-time-stocks-indexer ---
[INFO] No tests to run.
[INFO] Surefire report directory: /Users/ryantabora/Code/ryantabora/RealTimeSearchAndAnalytics/07-real-time-demo/stocks-indexer/target/surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------

Results :

Tests run: 0, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-jar-plugin:2.3.2:jar (default-jar) @ real-time-stocks-indexer ---
[INFO] Building jar: /Users/ryantabora/Code/ryantabora/RealTimeSearchAndAnalytics/07-real-time-demo/stocks-indexer/target/real-time-stocks-indexer-1.0-SNAPSHOT.jar
[INFO]
[INFO] --- maven-shade-plugin:2.0:shade (default) @ real-time-stocks-indexer ---
[INFO] Including junit:junit:jar:4.10 in the shaded jar.
[INFO] Including org.hamcrest:hamcrest-core:jar:1.1 in the shaded jar.
[INFO] Including org.apache.hbase:hbase:jar:0.94.1 in the shaded jar.
.
.
.
[WARNING] We have a duplicate org/hamcrest/BaseDescription.class in /Users/ryantabora/.m2/repository/org/hamcrest/hamcrest-core/1.1/hamcrest-core-1.1.jar
[WARNING] We have a duplicate org/hamcrest/BaseMatcher.class in /Users/ryantabora/.m2/repository/org/hamcrest/hamcrest-core/1.1/hamcrest-core-1.1.jar
[WARNING] We have a duplicate org/hamcrest/core/AllOf.class in /Users/ryantabora/.m2/repository/org/hamcrest/hamcrest-core/1.1/hamcrest-core-1.1.jar
[WARNING] We have a duplicate org/hamcrest/core/AnyOf.class in /Users/ryantabora/.
.
.
.
[INFO] Replacing original artifact with shaded artifact.
[INFO] Replacing /Users/ryantabora/Code/ryantabora/RealTimeSearchAndAnalytics/07-real-time-demo/stocks-indexer/target/real-time-stocks-indexer-1.0-SNAPSHOT.jar with /Us
[INFO] Dependency-reduced POM written at: /Users/ryantabora/Code/ryantabora/RealTimeSearchAndAnalytics/07-real-time-demo/stocks-indexer/dependency-reduced-pom.xml
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 18.925s
[INFO] Finished at: Fri Jan 25 15:39:53 PST 2013
[INFO] Final Memory: 17M/81M
[INFO] ------------------------------------------------------------------------
```

Staging the Data in HBase

To stage the data in HBase, we must run the StocksHBaseLoader. We can do this with the jar we just built. We must provide the StocksHBaseLoader with a properties file, an input path to the CSV files in HDFS, and an output directory in HDFS for the job. For example:

```
hadoop jar ./target/real-time-stocks-indexer-1.0-SNAPSHOT.jar com.ryantabora.tutorial.StocksHBaseLoader ../tutorial.properties /tinydata /mytestoutput
```

This job will create the STOCKS table if it does not exist, and then populate it with records it finds in the tinydata directory in HDFS. The output should look something like this:

Indexing Solr

Now that the data is staged in HBase, we can index it in Solr. We will load a stock symbol at a time. The StocksIndexer requires the tutorial.properties file path as well as the stock symbol that you wish to index. For example, this call loads all of the stock data we have for AAME.

```
hadoop jar ./target/real-time-stocks-indexer-1.0-SNAPSHOT.jar com.ryantabora.tutorial.StocksIndexer ../tutorial.properties AAME
```

Remember that as soon as we start loading this data in Solr we want to immediately run the web application so we can query the data as it is being indexed.

The Query Web Application

Running the web application is as simple as pointing your browser to the URL in which the page is hosted to. You can enter any stock symbol you would like to query and the application will generate the queries to Solr. All you need to do is sit back and watch the graph be created!