W205 - Exercise 2

Daniel Balck

Application Overview

This application is a single-node instantiation of a Storm cluster for processing Twitter Streams. It consumes tweets that contain basic common words (like “a”) that twitter publishes through its Public Streaming API, counts word occurrences, and either updates or inserts these occurrence counts into a Postgres relational database.

The application uses six main software components. First, the application uses Apache Storm to do real-time message processing. Storm has excellent throughput for parsing and merging streams of data on the fly. Apache Storm consists roughly of two parts, spouts and bolts. Spouts act as intake nodes, while bolts act as processing nodes. Our spout consumes raw tweets and passes these to bolts, which parse the tweets and count word occurrences, saving them to a Postgres database.

Second, we have a PostgreSQL database that can allow the multiple write threads to interact without causing inconsistency, thanks to the databases atomic transactions. Third, we use Stream Parse, a Python implementation of the (traditionally Java) Apache Storm. While Storm still uses a JVM under the hood, Stream Parse allows bolts and spouts to be Python modules, which makes development somewhat easier.

We also use a couple Python libraries, Tweepy and Psycopg. These libraries allow for easy access to our data intake and output – Tweepy interfaces with Twitter and outputs tweets for our Storm spout, and Psycopg allows the final wordcount bolt to insert rows into Postgres.

Our sixth software component is Twitter itself. Its public APIs are second to none, and allows a massive amount of raw social network data to be available to researchers. What’s more, this data is real-time, and thus reflects what is happening all over the world at any given moment. This can be combined with other third party data to analyze reactions in social networks to current events, or, in this case, just to do simple text analysis. To use the API, an account needs to be set up and authentication tokens generated. For our project, the authentication tokens are hardcoded into our application.

The software components are thus:

- Apache Storm (http://storm.apache.org/)
- PostgreSQL (https://www.postgresql.org/)
- Stream Parse (https://github.com/Parsely/streamparse)
- Tweepy (https://github.com/tweepy/tweepy)
- Psycopg (http://initd.org/psycopg/)
- Twitter (https://dev.twitter.com/)

Directories

Directories	Notes
extweetwordcount/	Top level directory for project
extweetwordcount /topologies	Directory containing Storm Topology files (.clj)
extweetwordcount /virtualenvs	Python configuration folder for dependencies
extweetwordcount /src/spouts	Directory containing Storm spouts (for handling input). This is also a python package.
extweetwordcount /src/bolts	Directory containing Storm bolts for processing data tuples. This is also a python package.
extweetwordcount/screenshots	Directory containing screenshot deliverables

Files

Files	Notes
extweetwordcount /topologies/tweetwordcount.clj	Topology configuration file for the project. Determines the structure by which the spouts and bolts connect to form a processing pipeline.
extweetwordcount /virtualenvs/tweetwordcount.txt	Text file specifying which virtualenv dependencies should be imported and run. Contains entries for streamparse, tweepy, and redis.
extweetwordcount/src/spouts/__init__.py	Empty python file marking this directory as a python package
extweetwordcount/src/spouts/tweets.py	Python class that, using tweepy, streams tweets from Twitter via the Streaming API. This class contains attributes for accessing the API and formatting the data as tuples, which are passed on to bolts.
extweetwordcount/src/bolts/__init__.py	Empty python file marking this directory as a python package
extweetwordcount/src/bolts/parse.py	Python class that splits tweets into individual words, and emits those to a wordcount bolt.
extweetwordcount/src/bolts/wordcount.py	Python class that receives words and counts their cumulative occurance in the stream. As time passes, the list of unique words and occurances (count) of each word increase. Words, after being counted, are dropped.

extweetwordcount/readme.txt	Text file containing basic info on the project and how to run it.
extweetwordcount/config.json	A configuration file for Storm that essentially points to other configuration files (like in topologies and virtualenvs), as well as basic storm settings (e.g. max byte count)
extweetwordcount/fabfile.py	Contains before and after functions that bracket the topology, if any external set up is required. This file is note used in our project.
extweetwordcount/project.clj	Storm configuration file containing dependency jars and other JVM settings necessary for the project to execute.
extweetwordcount/init_db.py	Python script to create the tcount database and the tweetwordcount table.
extweetwordcount/reset_db.py	Python script to drop all rows from the tweetwordcount table. For debugging purposes.
extweetwordcount/setup_environment.sh	Bash shell script to install dependencies. It takes a single argument, an unmounted drive, which it will reformat and install a postgres database onto. The script also installs tweepy and psycopg, if they do not already exist.
extweetwordcount/Plot.png	Histogram of most common words
extweetwordcount/finalresults.py	Python program that outputs the occurrence of a single word provided as an argument. If no arguments are provided, it will output all words with their counts alphabetically. e.g. python finalresults.py hello
extweetwordcount/histogram.py	Python program that outputs words that fit a certain range or occurrence. Two arguments are given, separated by a comma (no space). The first is the lower bound (inclusive) of the wordcount, while the second is the inclusive upper bound. e.g. python histogram.py 15,20

Installing and Running

- The application can be run very easily. First, clone the github repository to your UBC MIDS W205 EX2-FULL machine (AMI ID: ami-d4dd4ec3). Change directories into the execise_2/extweetwordcount folder. This is the home folder for this particular project.
- Now attach a clean EBS volume to the AMI, and find the device path (e.g. /dev/xvdf). This volume will be reformatted, so please do not use this if it has valuable data. Run the setup_environment.sh script with the path to your volume, i.e. ./setup_environment.sh /dev/xvdf. This will install PostgreSQL, Tweepy, and Psycopg (Storm and Stream Parse are already installed on the AMI).
- Now initialize the database with the python script (python init_db.py). This will create the tcount database and the tweetwordcount table.
- Lastly, type “sparse run” inside the project folder to run the application. CTRL-C will stop it.
- Once a fair amount of tweets have been parsed, use the finalresults.py and histogram.py scripts to analyze the data.

Have fun!