Luis Marion

MET CS 755 – Cloud Computing

Final Project – World of Warcraft Auction Stats

The project is comprised of three Map/Reduce jobs and the job runner.

AuctionsJob – the runner class. The paths were all just hardcoded for simplicities sake.

GetAuctionsMapper/GetAuctionsReducer – iterates through a newline separated list of wow servers. The file is loaded in the “servers“ directory (only needed directory to start job). There is a master list of all 288 servers that sits at the root of the eclipse project.

AuctionStatsMapper/AuctionStatsReducer – interim job that calculates the average of all of the auctions and emits them to a separate file. Average is standard sum(xi – xn)/n. Uses the itemid as the key. In hindsight, I should’ve also added the totalAuctions variable to the emitted json object (especially since I calculate it anyway).

AuctionStatsReporterMapper/AuctionStatsReporterReducer – the purpose of this job is to calculate the standard deviation, then emit the same average row from previous job, with the aggregated standard deviation data. It also emits the min/max auctions across all servers in input set. I also wish I could’ve retrieved the item-name using a different apis, however blizzard limits requests to 3000 per day (which would easily be exceeded if doing item lookups).

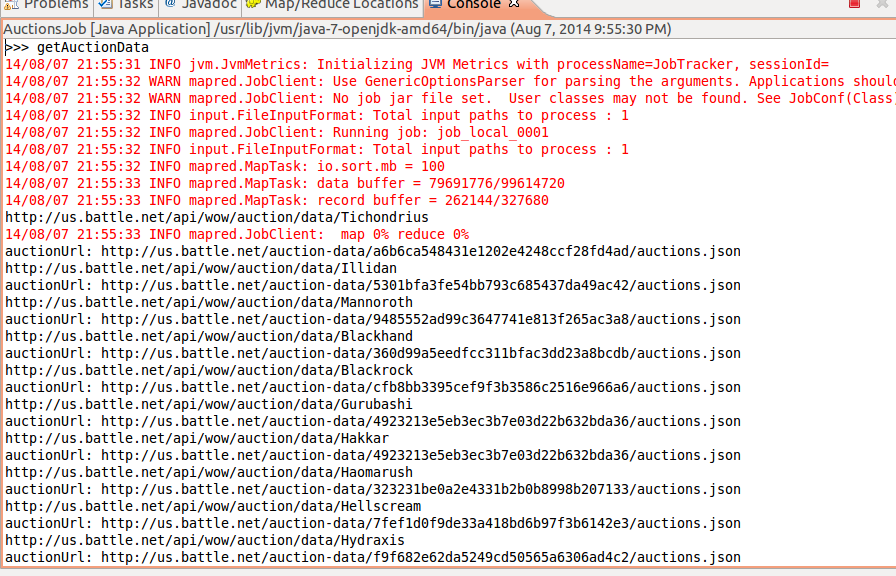
Since the standard deviation formula requires the calculation of each (xi – average)^2, I have to ensure that the average is the first row read (sequential iteration w/o using memory – by design). In order to do this I leverage the fact that hadoop will read all files from the input directory in alphabetical order. The ideal solution was to use MutlipleOutputs class from my AuctionStatsReducer and emit to two files (1 with original auctions, and another with nothing but averages – then give them filenames that would adhere to the logical ordering). However MultipleOutputs does not work with Hadoop 0.20.2 and I didn’t have the time to reconfigure both my Eclipse and “deployed” Ubuntu environment.

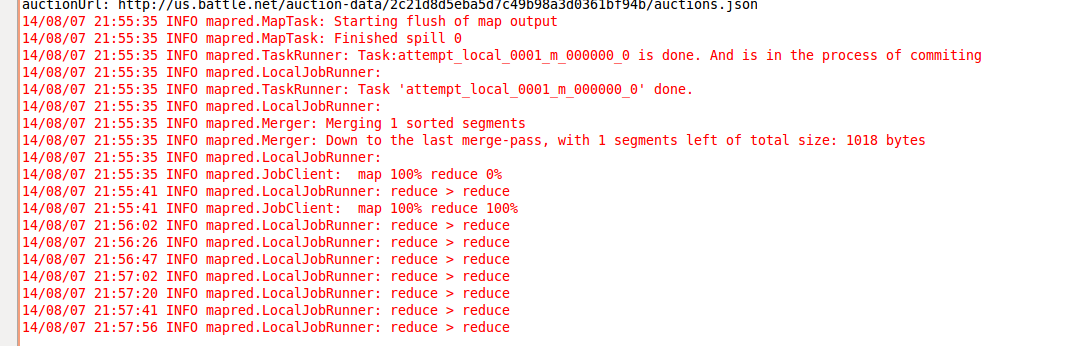
As a result, the job starts off with a wonky work-around to put the output for the average calculator (AuctionStatsReducer) and the list of auctions (GetAuctionsReducer) in the same directory. The work-around was to copyToLocal, then use FileUtil.copyMerge() to move back into the original hdfs output directory from the GetAuctionsReducer (the auctions directory). I discovered this by pure chance and personally feel like its bug in hadoop as the behavior seems inconsistent (it renames the file to the directory name if there is a naming conflict – which there is in my case – part-00000). This weird behavior works in my favor because I needed the new file to have a lower order alphabetically so it can be loaded first anyway.

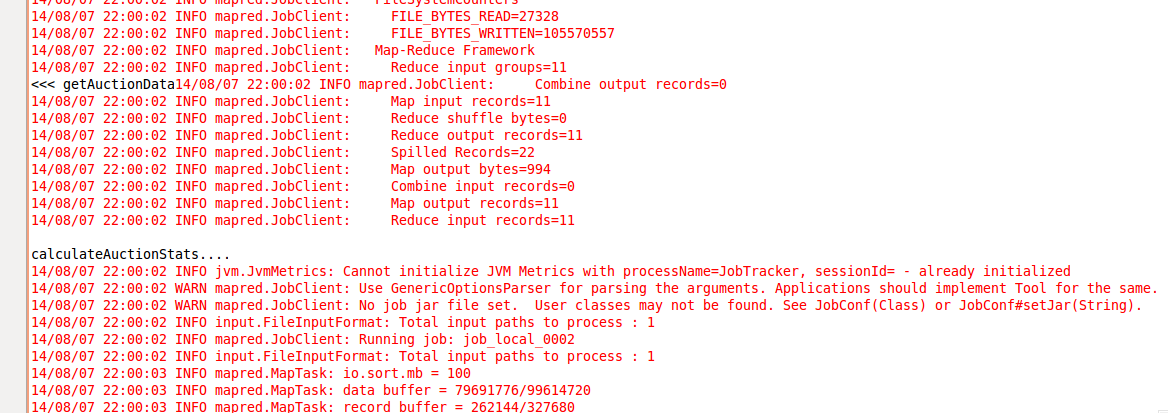
To run the job you can tweak the list of servers in the servers directory and right-click on eclipse and say “run on hadoop”.

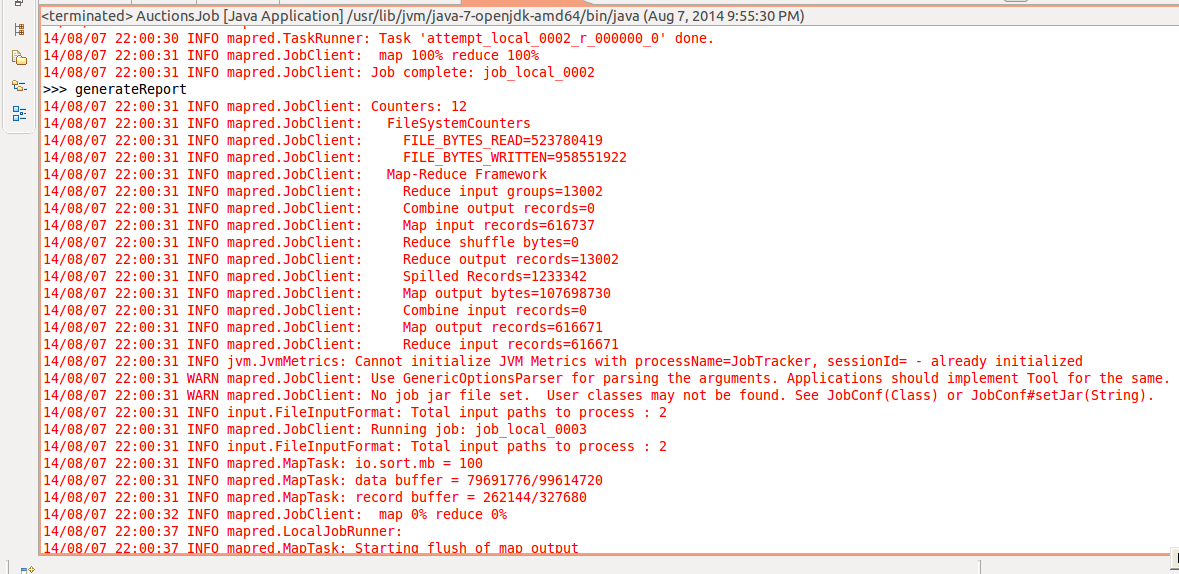
All related libraries are inside of the /lib directory. I did not attempt to run the job outside of Eclipse, so I do not have the classpath string on hand to run via command-line, but I would think it’s just a list of the jars in the lib dir (or copy/edit the .classpath file).

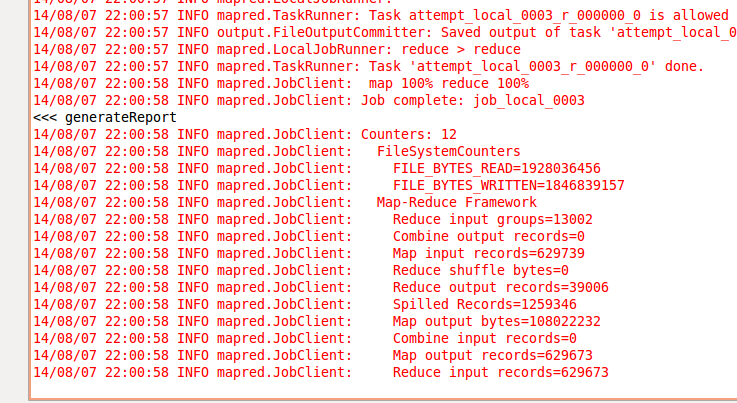
Below are some screens for a sample run.











This run took roughly 6 minutes and ran against 6 servers.

Tichondrius

Illidan

Mannoroth

Blackhand

Blackrock

Gurubashi

Hakkar

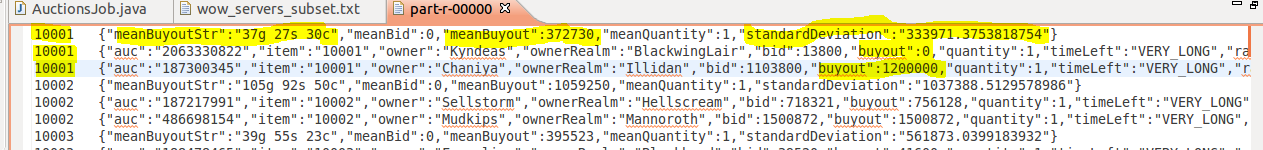
Haomarush

Hellscream

Hydraxis

Hyjal

The output looks as follows:



And follows the form AuctionMean.java, Auction.java (minimum), Auction.java (maximum) – java objects serialized to json – for all items.

**Note:** Occasionally in class I was having the web-service calls fail and bomb the job. I would remove the generated dirs (auctions, auction\_stats, output) and re-run and it would run fine. I suspect perhaps it was an internet/api issue as I wouldn’t have to change anything ...and subsequent runs would fail or pass almost on a whim. Additionally I’ve noticed it consistently behaves correctly late at night, so I suspect it could’ve been server load issues. I never got around to correcting this issue (there should be better error handling to just skip server but not fail job).

**Note 2:** Blizzard has a cap of 3000 requests per day, and since there are 288 servers each requiring 2 webservice calls, the number adds up quickly. I did not hit the cap however. Running against the full list of servers took about 1.5 hours on Tuesday back before the third job for standard deviation existed.