P4

Ian Davis

W002icd

CEG3900

03/10/2017

Github URL: https://github.com/IanDavis1995/P4.git

Task 1

Summary

To develop this java program, I used the java 8 builtin Stream and nio packages. I created three functions, filterLogFile, readLogFile and writeLogFile. readLogFile returns a Stream of Strings with each line of text from the input file. writeLogFile writes a Stream of Strings as each of line of a given output file. filterLogFile calls both of these routines, filtering out the Stream of input lines to find the relevant log messages and pull out the information required.

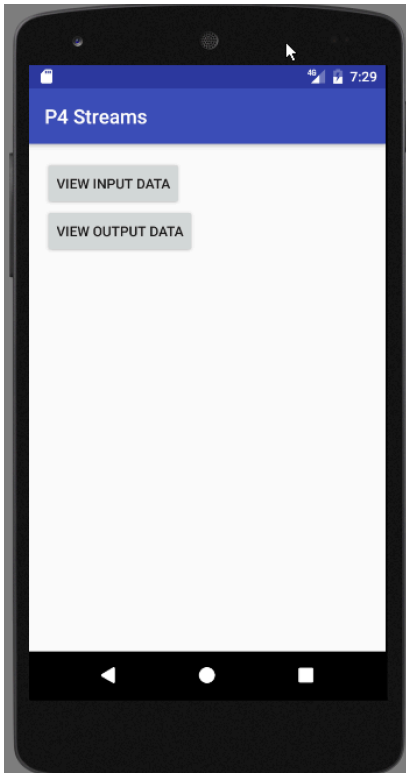
Experience Report

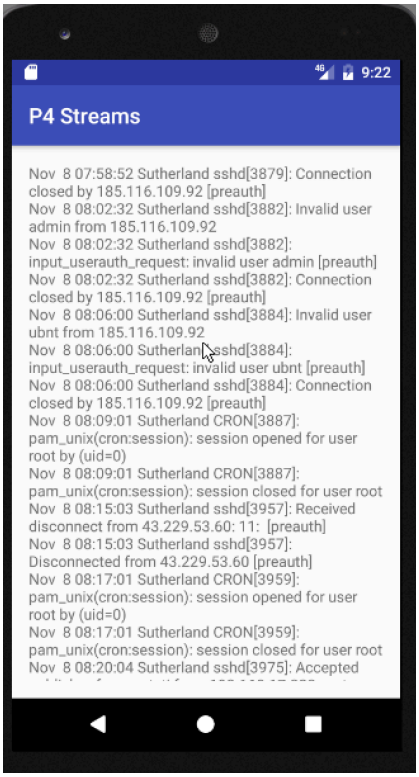
Using streams is a much simpler way to write code that performs operations on large sets of data. They are a lot more expressive when it comes to working with sets of data, and serve to make much more efficient routines.

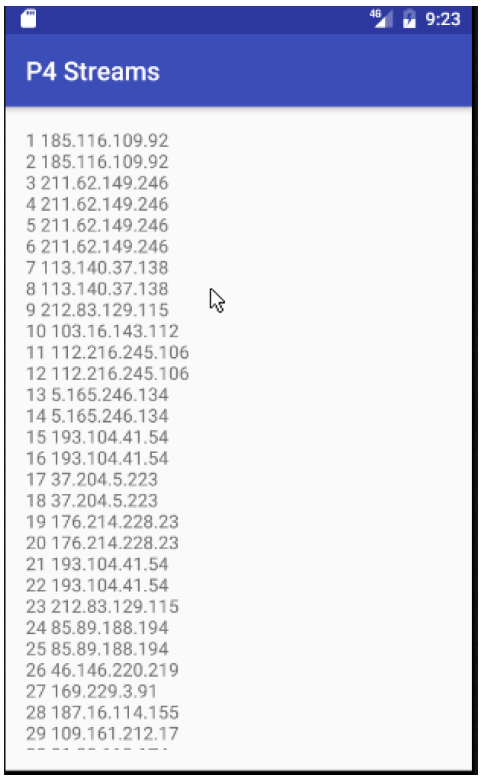
Status Report

The Java version of the program is mostly working in all instances. However, the parsing is very specific and looks for exactly formatted log messages. If any variation is detected, the application will die immediately.

Task 2







Status report

Overall: Mostly working

Known Issues:

- The file is only read when the app first starts, so modifications after that point are not reflected.
- The input file does not wrap content very well, a smaller font may have helped with this.
- Input and output files are stored as resource files in the android app and not configurable.

Experience Report

Androids APKs are still behind the latest java, so reading and writing files was just a little bit less easy/efficient in the android app. I also struggled to get file read/write permissions working properly. The only way I could manage to read in a file was to save it as a resource of the app itself. Adding in permissions for Read and write external storage were necessary to write the file out, but did not seem to fix the issue of reading in files existing that did not belong to the application.

Task 3

- Status report pg. 4
- Experience report pg. 4

Status Report

Overall: Mostly working

Known issues:

- Some non-word HTML keywords are counted from the data downloaded. IE: textalign, center, etc. I could not find a perfect way to remove all HTML markup and keywords from the file, but did the best I could.

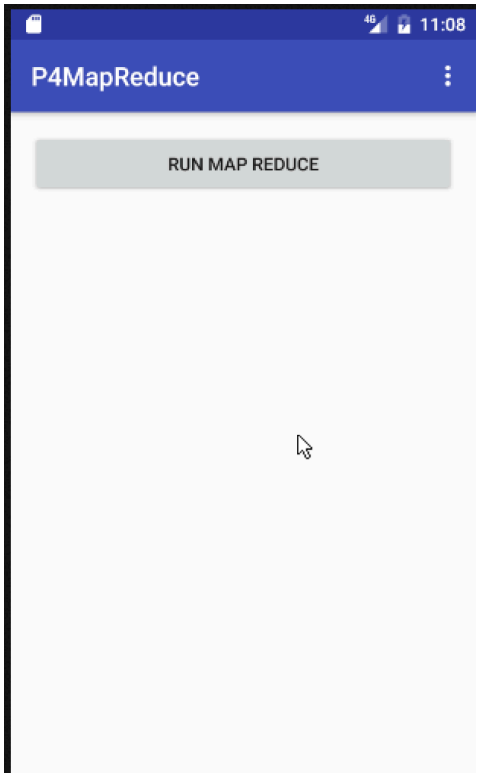
Experience Report

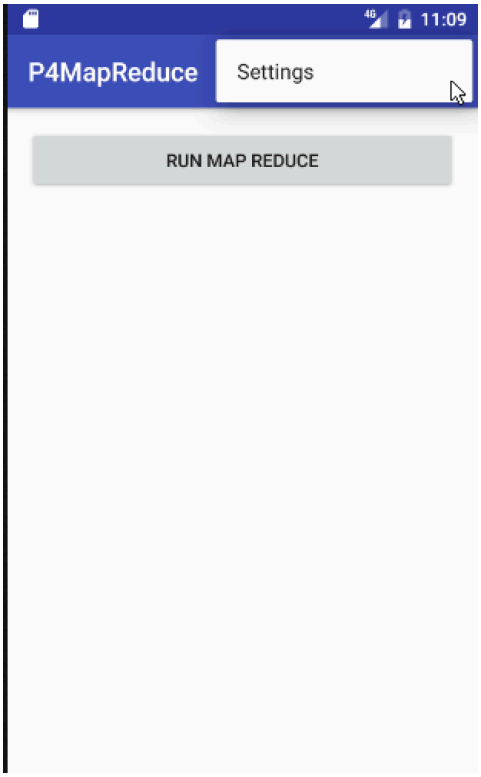
Map and Reduce using AWS cloud was not entirely clear to me. My final implementation ended up using java 8 stream map + filter calls, mapping regexes that remove different HTML tags from the data, as well as splitting lines up by words. I then use a collect expression to aggregate the final filtered stream down to a Map with the words as keys and the number of appearances as values. This is then written to the output file. My implementation did not make use of the Scanner class, instead opting to use regular expression patterns to parse the input down into logical words. WordCount.java showcases an attempted implementation of an Apache Hadoop job that could be run through an Amazon EMR cluster.

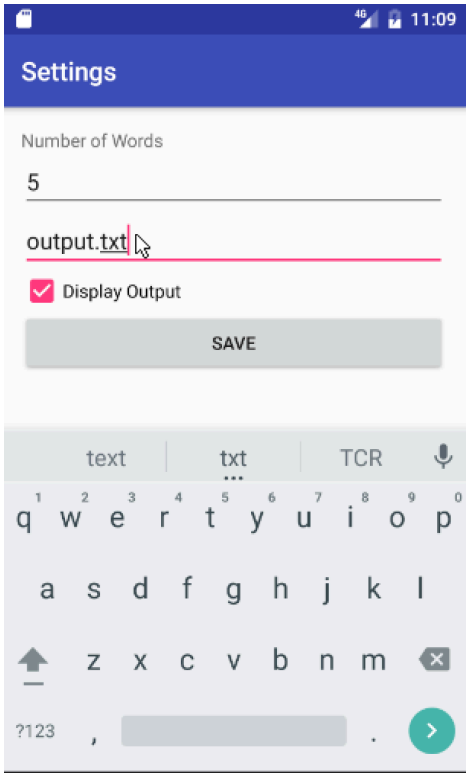
Example inputs and outputs can be found at ./streams/P4StreamsJava/ under github

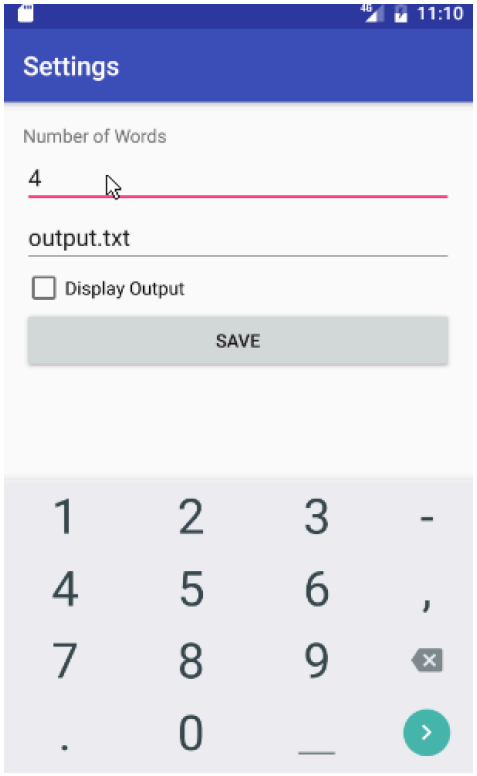
Task 4

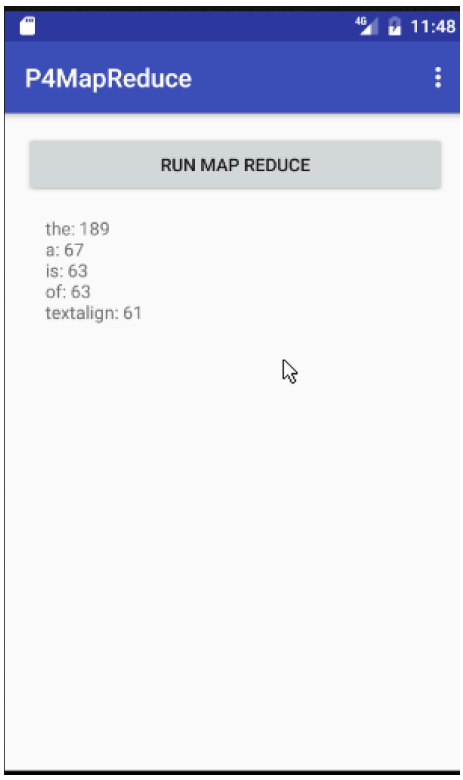
- Screenshots pg. 6-7
- Status report pg. 7
- Experience report pg. 7-8


11:08
P4MapReduce
RUN MAP REDUCE


11:09
P4MapReduce
Settings
RUN MAP REDUCE


11:09
Settings
Number of Words
5
output.txt
Display Output
SAVE
text
txt
TCR


11:10
Settings
Number of Words
4
output.txt
Display Output
SAVE



Status Report

Overall: Mostly Functioning

Issues:

- The settings page does not update controls with their current values when the activity first starts.

Experience Report

Downloading webpage data through android turned out to be a bigger pain than expected. There was some major trouble implementing the download functionality using an AsyncTask (to avoid Network on the main thread exceptions) and aggregating everything into a single Stream object.

After much tinkering I got this to work. I also struggled to get the settings activity to return back to the main activity after pressing save, but adding in a finish() call fixed this issue.

Example inputs and outputs can be found on github at the following paths:

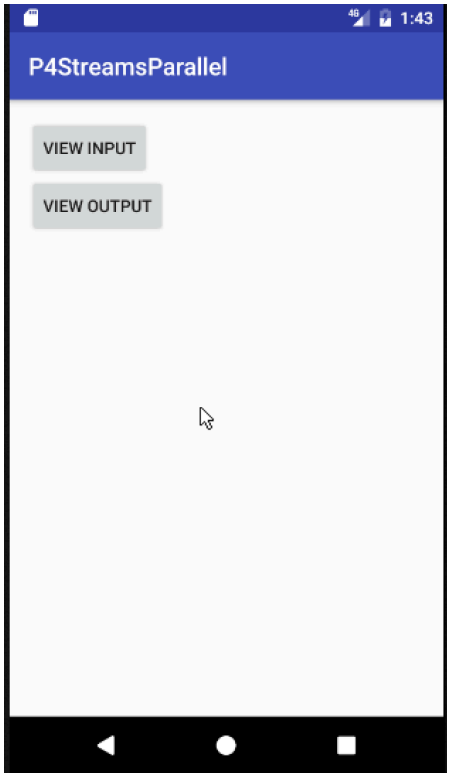
./map-reduce/P4MapReduce/ (Android APK Implementation)

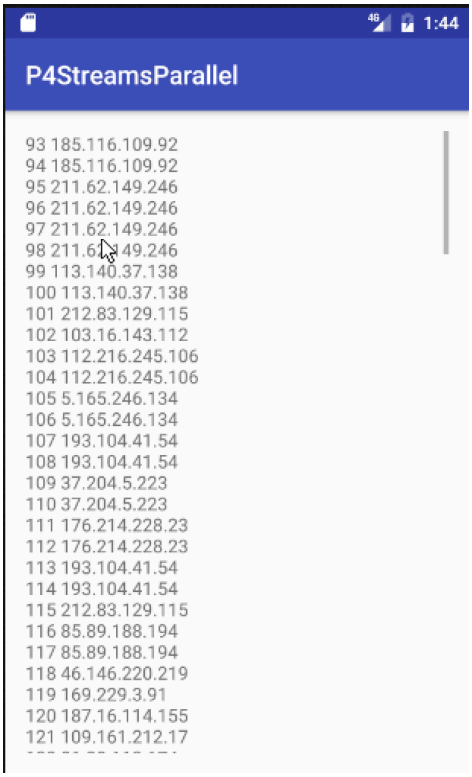
and

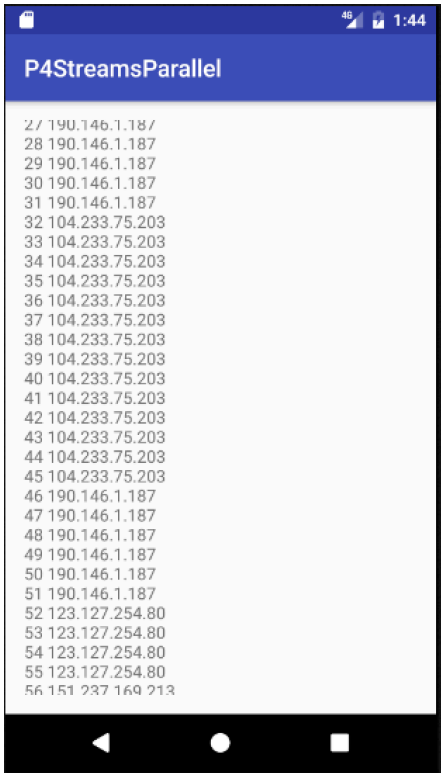
./map-reduce/P4MapReduceJava/ (Java Only Implementation)

Task 5

Parallel Streams APK







Status Report

Overall: Mostly Functioning

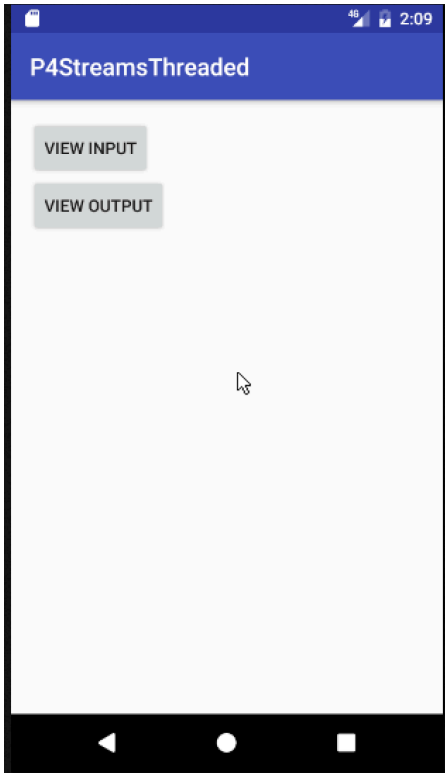
Known Issues:

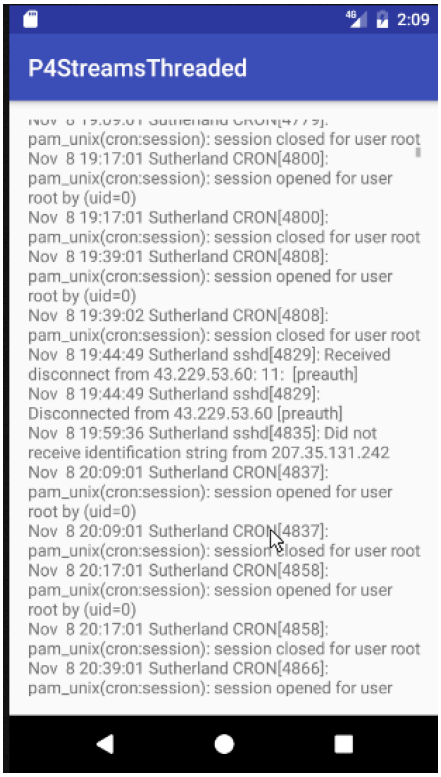
- The output does not come up in proper order, i.e lines 93-124 are shown before lines 1-93. This is because the parallel streams are splitting into (presumably) 2 different chunks on my vm and executing the map/reduce in improper order.

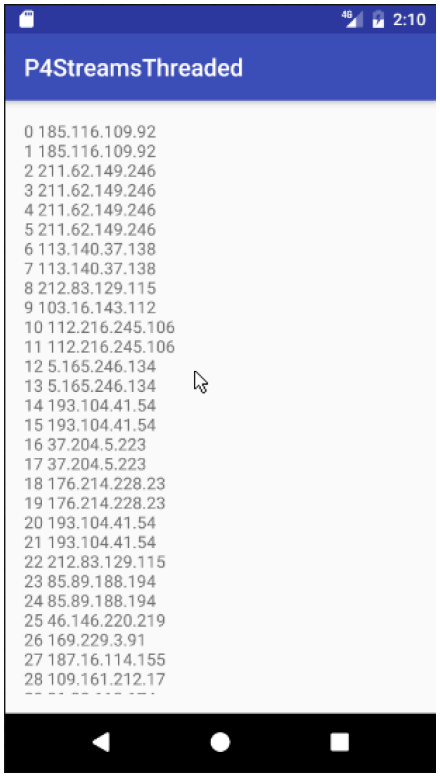
Experience Report

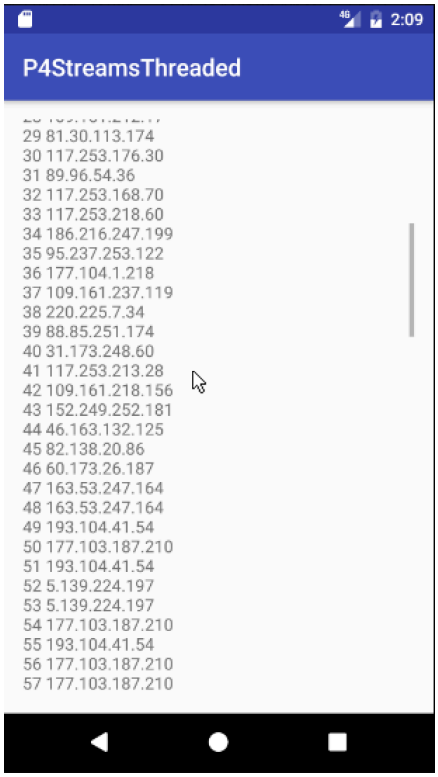
Implementing this using Parallel Streams was not much different than implementing the standard streams method. The only issue I ran into was that the output is now out of order based on the order it's found in the original file. The best thing I could find about keeping parallel streams in order involved using a ConcurrentMap, which didn't seem entirely useful to my specific use case.

Threaded Streams APK









Status Report

Overall: Mostly Functioning

Known Issues: None

Experience Report

Utilizing Threads for the Streams example was even simpler than the Parallel implementation. This was simply a matter of creating a new thread object that called an existing function in my MainActivity.