Report for Lab3 MapReduce

Name: Guanyu Wang Name: Zeyuan Li

# Requirement met

The MapReduce system developed by our team has met almost all the requirements listed in the lab handout. The requirement we met are listed below

* Our system can run on Andrew Linux Machine
* We have defined configure file for system block size, maximum number of task that can be run on one machine, system root directory and jobTracker address
* Our system can run the user code on any Andrew machines
* Our system can execute user code, taskTracker and jobTracker on different machines
* Our system can run multiple jobs (including failed jobs) at the same time without any concurrent related issues
* Our system has designed to have two layers of recovery mechanisms. If task fails on taskTracker, we restart the task for at most three times if it fails. If the taskTracker is down, we can reschedule the failed tasks after a timeout period and re-launch the task on other taskTrackers
* Since we can start multiple taskTrackers (data node), our jobTracker can schedule the jobs to full parallism and can run multiple tasks on multiple machines
* Our system has two layers of recover protection as stated above
* We have written general I/O facilities for our MapReduce library
* We have written management tool in the makefile to facilitate use and test of our library
* We have provided for the system administrator in the makefile comment to tell them how to run the system
* We have also provided javadoc in the source code for developers
* We have included three example programs (WordCount, Anagram and WordCountError). WordCount is the sample test file in Hadoop wiki to count the number of words in the documents. Anagram is to find the all pairs of anagram in the data file. WordCountError is the WordCount program with bug in it. This is to test error handling.

# Requirements didn’t meet

We have met all the requirements, at least to our understanding.

# Capability and limitations

3.1 Capability

Our MapReduce library is fully parallelizable, fault tolerant and behave very similar to the simplified Hadoop functionality. Our system can run different parts of the system (jobTracker, taskTracker and user program) in different machines and can start submitting jobs at any machines.

Our system also can schedule the jobs and tasks in a full-parallelized fashion. That means we can run multiple jobs on multiple machines and run multiple map or reduce tasks within each machine.

We also have implemented a two layers recover mechanism that once the task fails, we can restart the task for at most three times. Besides, if some taskTrackers fail that leads to some tasks fail, our system can reschedule the unfinished tasks to other task trackers.

3.1 Limitation

Since we can assume we have fixed input and output format, we fix the map and reduce input and output. This is a limitation of our system that we didn’t support more flexible input and output like real Hadoop does.

# What to improve

We may improve our I/O part to incorporate more flexible input and output formats. We may also build a more powerful command line shell for our hadoop, which is more painful to build.

# Documentation for application programmers

5.1 Developer guide

The usage of the program is very similar to the original MapReduce program. The user writes the Mapper and Reducer using the old MapReduce API. Since we assume that we have fixed input and output format, we limit our input and output to be both strings.

5.2 Tutorial

5.2.1 run word count

1) run the command to start rmi registry:

**make run-registry**

2) run the command to start jobtracker:

**make run-jt**

3) on every TaskTracker machine:

run the command to start tasktracker:

**make run-tt ADDR=?**

(where ? is the jobtracker machine's hostname)

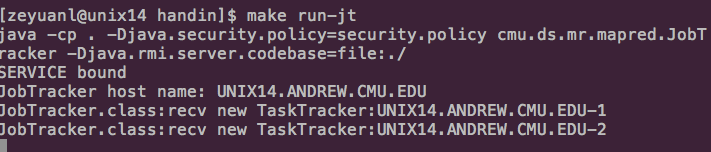
4) run the command to submit the word count job:

**make run-wc IN=? OUT=? NRED=?**

(where the first ? is the input file and second ? is the output directory, the third ? is the number of reduce tasks)

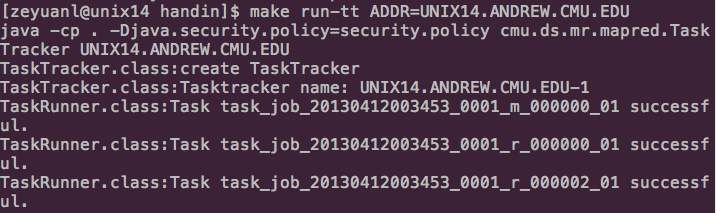
screenshots:

JobTracker

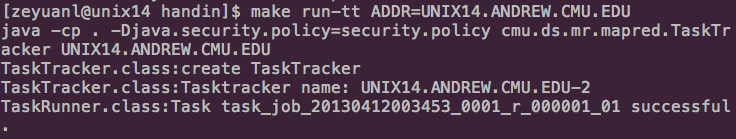


TaskTracker

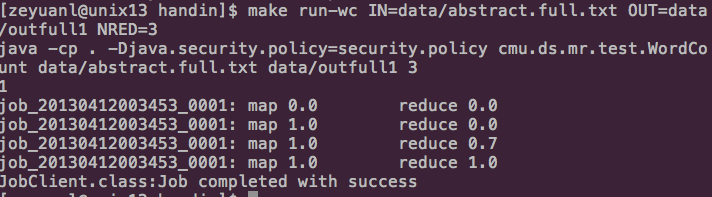
1



2



User program



5.2.2 run anagram

1) run the command to start rmi registry:

**make run-registry**

2) run the command to start jobtracker:

**make run-jt**

3) on every TaskTracker machine:

run the command to start tasktracker:

**make run-tt ADDR=?**

(where ? is the jobtracker machine's hostname)

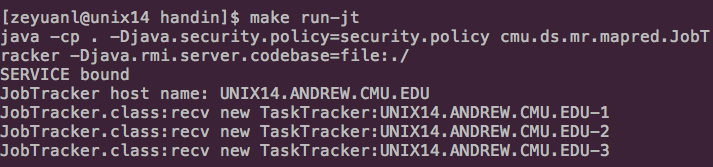
4) run the command to submit the anagram-finding job:

**make run-an IN=? OUT=? NRED=?**

(where the first ? is the input file and second ? is the output directory, the third ? is the number of reduce tasks)

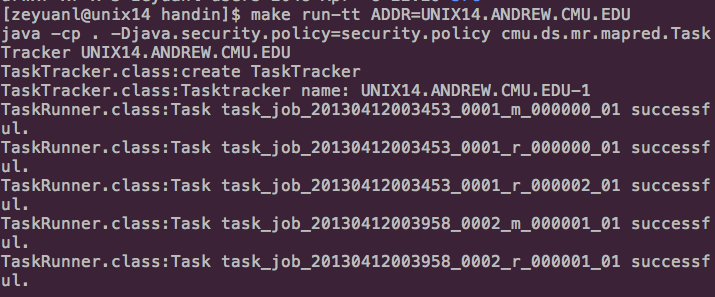
Screenshots:

JobTracker

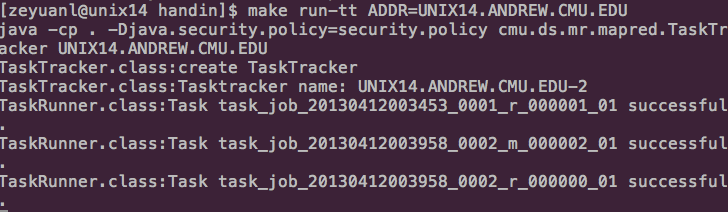


TaskTracker

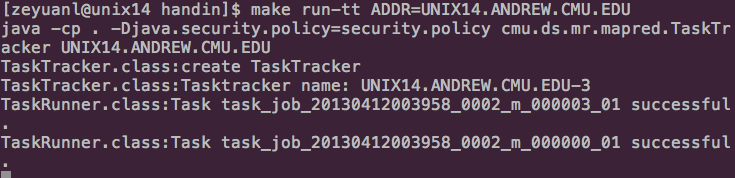
1



2



3



User program

