1. Weather Data Results

1.1) Weather Data Results for each of the versions of sequential and multithreaded program **without** Fibonacci Delay:

No of Threads = 4

```
SEQ
Average Running Time : 2947.9
Maximum Running Time : 3502
Minimum Running Time : 2853

NO-LOCK
Average Running Time : 1530.8
Maximum Running Time : 1601
Minimum Running Time : 1502
Speed Up : 1.92

COARSE-LOCK
Average Running Time : 1536.0
Maximum Running Time : 1568
Minimum Running Time : 1520
Speed Up : 1.91

FINE-LOCK
Average Running Time : 1557.8
Maximum Running Time : 1613
Minimum Running Time : 1534
Speed Up : 1.89

NO-SHARING
Average Running Time : 1538.2
Maximum Running Time : 1562
Minimum Running Time : 1525
Speed Up : 1.91
```

1.2) Weather Data Results for each of the versions of sequential and multithreaded program **with** Fibonacci Delay:

No of Threads = 4

```
SEQ
Average Running Time : 2931.3
Maximum Running Time : 3854
Minimum Running Time : 2770

NO-LOCK
Average Running Time : 1523.5
Maximum Running Time : 1577
Minimum Running Time : 1501
Speed Up : 1.92

COARSE-LOCK
Average Running Time : 1540.8
Maximum Running Time : 1607
Minimum Running Time : 1523
Speed Up : 1.90

FINE-LOCK
Average Running Time : 1562.4
Maximum Running Time : 1688
Minimum Running Time : 1538
Speed Up : 1.87

NO-SHARING
Average Running Time : 1540.6
Maximum Running Time : 1630
Minimum Running Time : 1514
Speed Up : 1.90
```

1.3) Question and Answers

1. Which program version (SEQ, NO-LOCK, COARSE-LOCK, FINE-LOCK, NO-SHARING) would you normally expect to finish fastest and why? Do the experiments confirm your expectation? If not, try to explain the reasons.

Answer: I expected the NO-LOCK program version to finish the fastest. It is faster than SEQ because we are processing the data using multiple threads (parallel processing). And it is faster than other multi-threaded versions because we are not blocking the processing while working on shared data or no overhead of merging the results as we have seen in NO-SHARING approach.

Yes, The results confirm my expectation. We got the best Speed up in NO-LOCK program.

2. Which program version (SEQ, NO-LOCK, COARSE-LOCK, FINE-LOCK, NO-SHARING) would you normally expect to finish slowest and why? Do the experiments confirm your expectation? If not, try to explain the reasons.

Answer: I expected the SEQ program to finish slowest as we are not doing any parallel processing here. We have a data available in memory and have a processors to work for but due to lack of parallelism we are not using them optimally. We are processing each line one by one here and which is making it slower.

Yes, The results confirm my expectation.

3. Compare the temperature averages returned by each program version. Report if any of them is incorrect or if any of the programs crashed because of concurrent accesses.

Answer: I have added a code which compares the averages of each program version and found that NO-LOCK program is giving the incorrect averages and all other approaches are giving the same results. This is because we have not used Locking in this version, due to this multiple threads makes shared data structure inconsistent.

4. Compare the running times of SEQ and COARSE-LOCK. Try to explain why one is slower than the other. (Make sure to consider the results of both B and C—this might support or refute a possible hypothesis.)

Answer: COARSE-LOCK program is much faster that SEQ (Speed Up around 1.9) for both and C. This is because in SEQ approach we are processing each record one by one but in COARSE-LOCK approach we have divided our same task to 4 threads and each thread work on there piece of data.

Though we have a good performance in COARSE-LOCK, if we do very intense work (high time) inside synchronized block (locked common data structure) than at a time only one thread can work on that part of code which makes it serial and reduces our performance.

5. How does the higher computation cost in part C (additional Fibonacci computation) affect the difference between COARSE-LOCK and FINE-LOCK? Try to explain the reason.

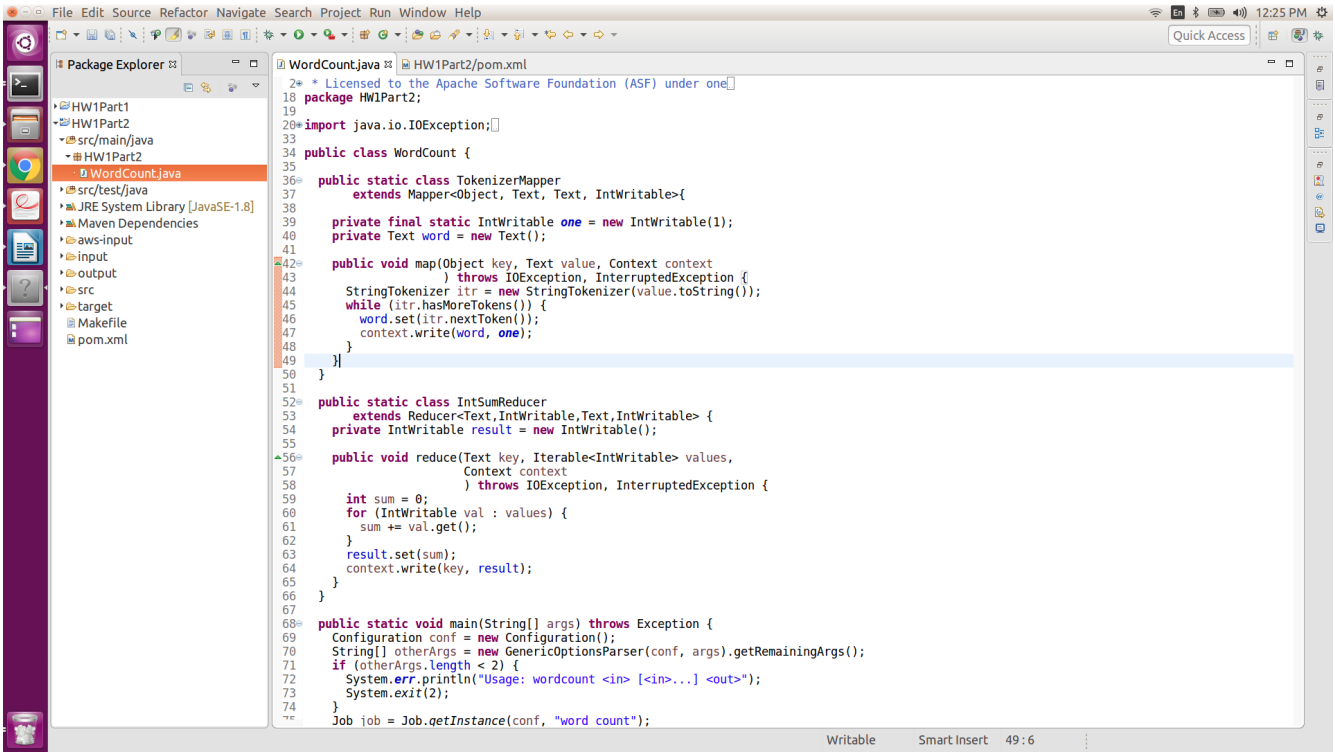
Answer: I cannot see much difference in my results but by speed-up measure, I can say adding the higher computation cost in Part C reduced Speed-up of COARSE-LOCK approach from 1.91 to 1.9 while of FINE-LOCK approach from 1.89 to 1.87 (higher impact).

In COARSE-LOCK, we are locking entire data structure while using shared data structure. On the other side in FINE-LOCK, we are locking only specific item of the data structure while working on that piece. So as we increase the computation (Part C) at the time when we have locked the entire shared data structure, all other threads who want to work on the other part of the data structure will also wait. While in case of FINE-LOCK, two threads who want to work on different part of data structure can work simultaneously.

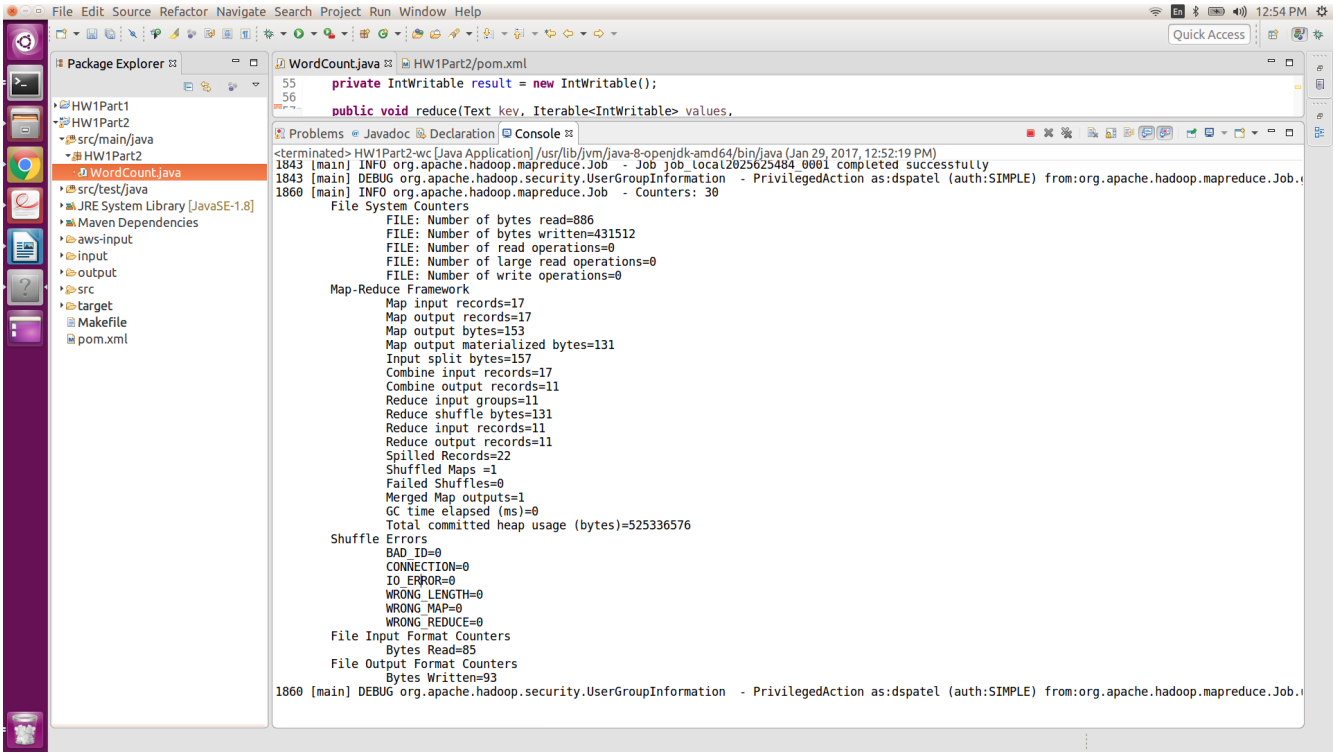
Ideally, as we increase the task to be done while locking common data structure (inside synchronized block) the performance of COARSE-LOCK approach should reduce at higher rate compared to the reduction in performance of FINE-LOCK approach.

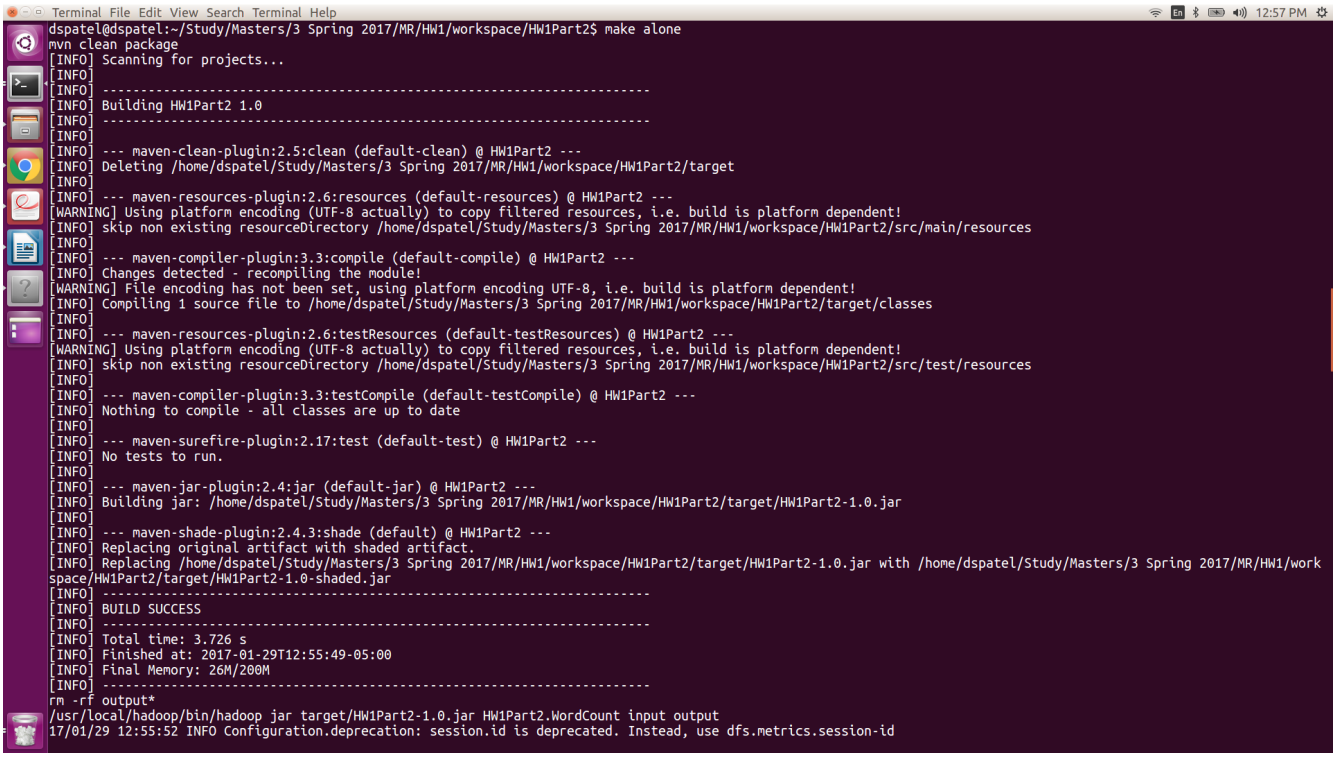
2. Word Count Local Execution

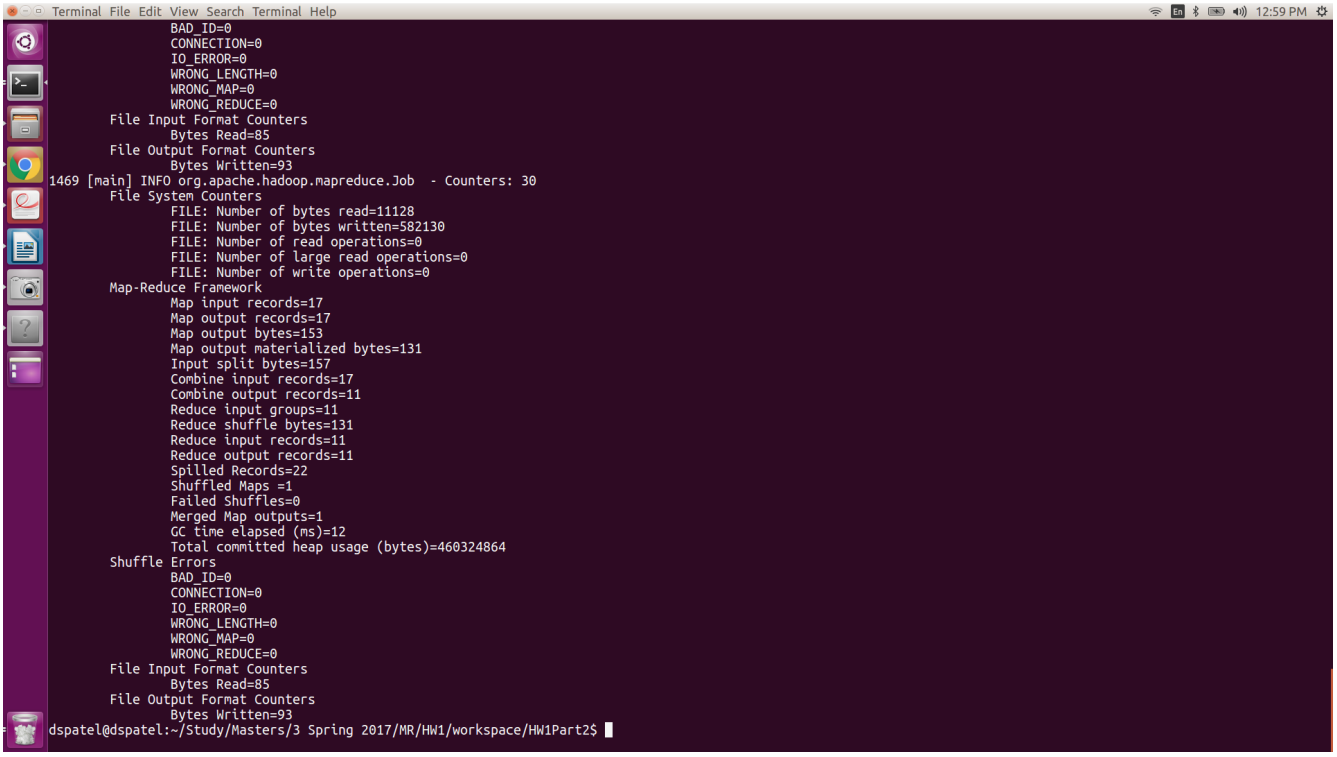
2.1) Project directory structure



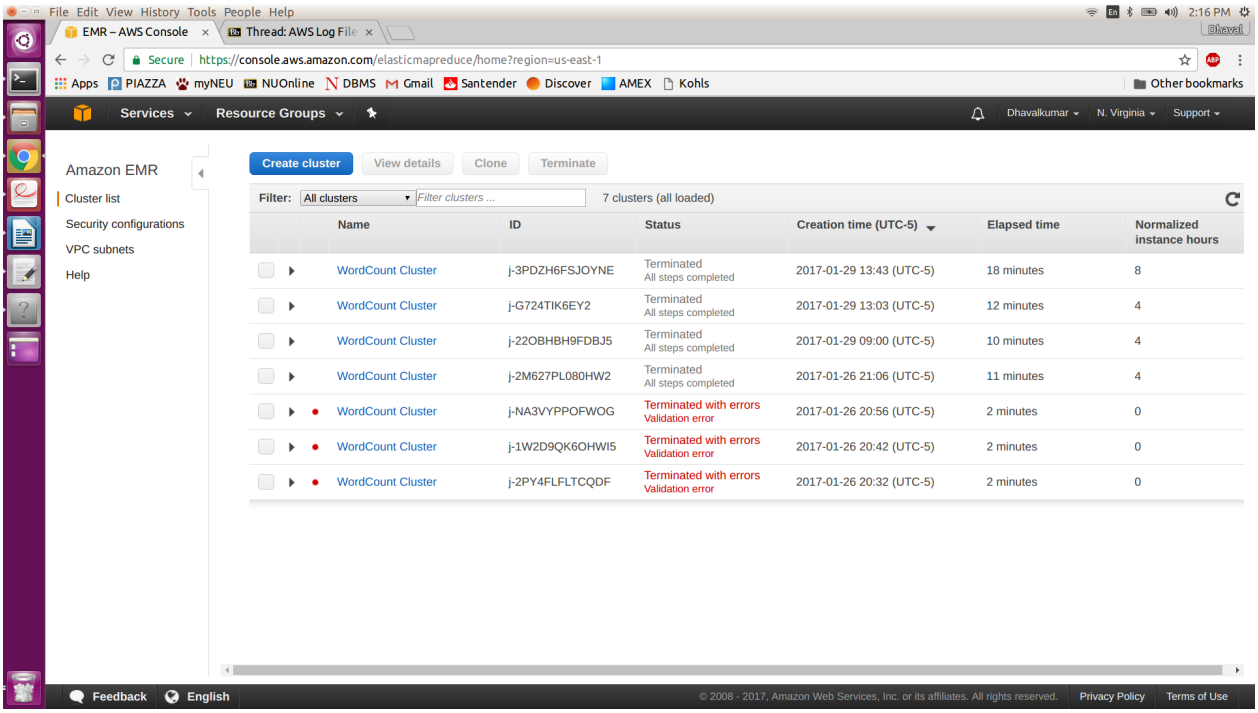
2.2) Console Output for a successful run of the WordCount Problem from IDE and Command Prompt



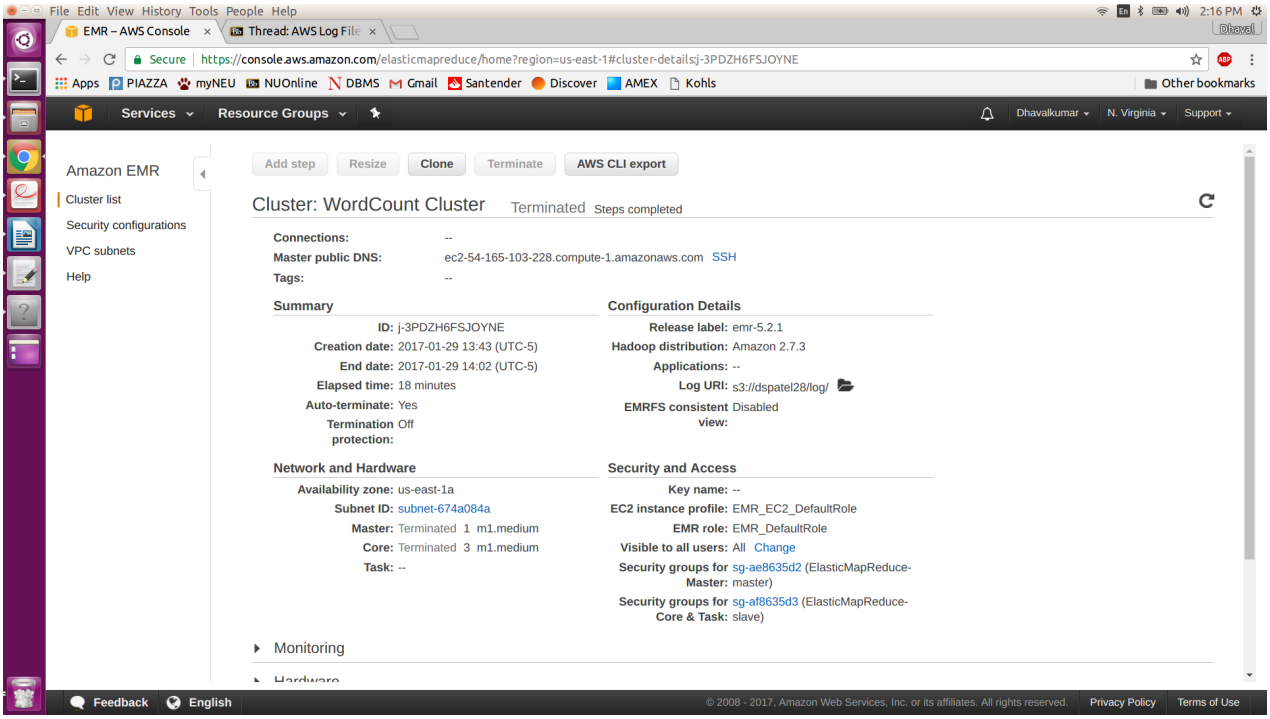

Terminal File Edit View Search Terminal Help
12:57 PM
dspatel@dspatel:~/Study/Masters/3 Spring 2017/MR/HW1/workspace/HW1Part2$ make alone
mvn clean package
[INFO] Scanning for projects...
[INFO]
[INFO] --
[INFO] Building HW1Part2 1.0
[INFO] --
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ HW1Part2 ---
[INFO] Deleting /home/dspatel/Study/Masters/3 Spring 2017/MR/HW1/workspace/HW1Part2/target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ HW1Part2 ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /home/dspatel/Study/Masters/3 Spring 2017/MR/HW1/workspace/HW1Part2/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.3:compile (default-compile) @ HW1Part2 ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 1 source file to /home/dspatel/Study/Masters/3 Spring 2017/MR/HW1/workspace/HW1Part2/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ HW1Part2 ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /home/dspatel/Study/Masters/3 Spring 2017/MR/HW1/workspace/HW1Part2/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.3:testCompile (default-testCompile) @ HW1Part2 ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.17:test (default-test) @ HW1Part2 ---
[INFO] No tests to run.
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ HW1Part2 ---
[INFO] Building jar: /home/dspatel/Study/Masters/3 Spring 2017/MR/HW1/workspace/HW1Part2/target/HW1Part2-1.0.jar
[INFO]
[INFO] --- maven-shade-plugin:2.4.3:shade (default) @ HW1Part2 ---
[INFO] Replacing original artifact with shaded artifact.
[INFO] Replacing /home/dspatel/Study/Masters/3 Spring 2017/MR/HW1/workspace/HW1Part2/target/HW1Part2-1.0.jar with /home/dspatel/Study/Masters/3 Spring 2017/MR/HW1/workspace/HW1Part2/target/HW1Part2-1.0-shaded.jar
[INFO] --
[INFO] BUILD SUCCESS
[INFO] --
[INFO] Total time: 3.726 s
[INFO] Finished at: 2017-01-29T12:55:49-05:00
[INFO] Final Memory: 26M/200M
[INFO] --
rm -rf output*
/usr/local/hadoop/bin/hadoop jar target/HW1Part2-1.0.jar HW1Part2.WordCount input output
17/01/29 12:55:52 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id


Terminal File Edit View Search Terminal Help
12:59 PM
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=85
File Output Format Counters
Bytes Written=93
1469 [main] INFO org.apache.hadoop.mapreduce.Job - Counters: 30
File System Counters
FILE: Number of bytes read=11128
FILE: Number of bytes written=582130
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
Map-Reduce Framework
Map input records=17
Map output records=17
Map output bytes=153
Map output materialized bytes=131
Input split bytes=157
Combine input records=17
Combine output records=11
Reduce input groups=11
Reduce shuffle bytes=131
Reduce input records=11
Reduce output records=11
Spilled Records=22
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=12
Total committed heap usage (bytes)=460324864
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=85
File Output Format Counters
Bytes Written=93
dspatel@dspatel:~/Study/Masters/3 Spring 2017/MR/HW1/workspace/HW1Part2$

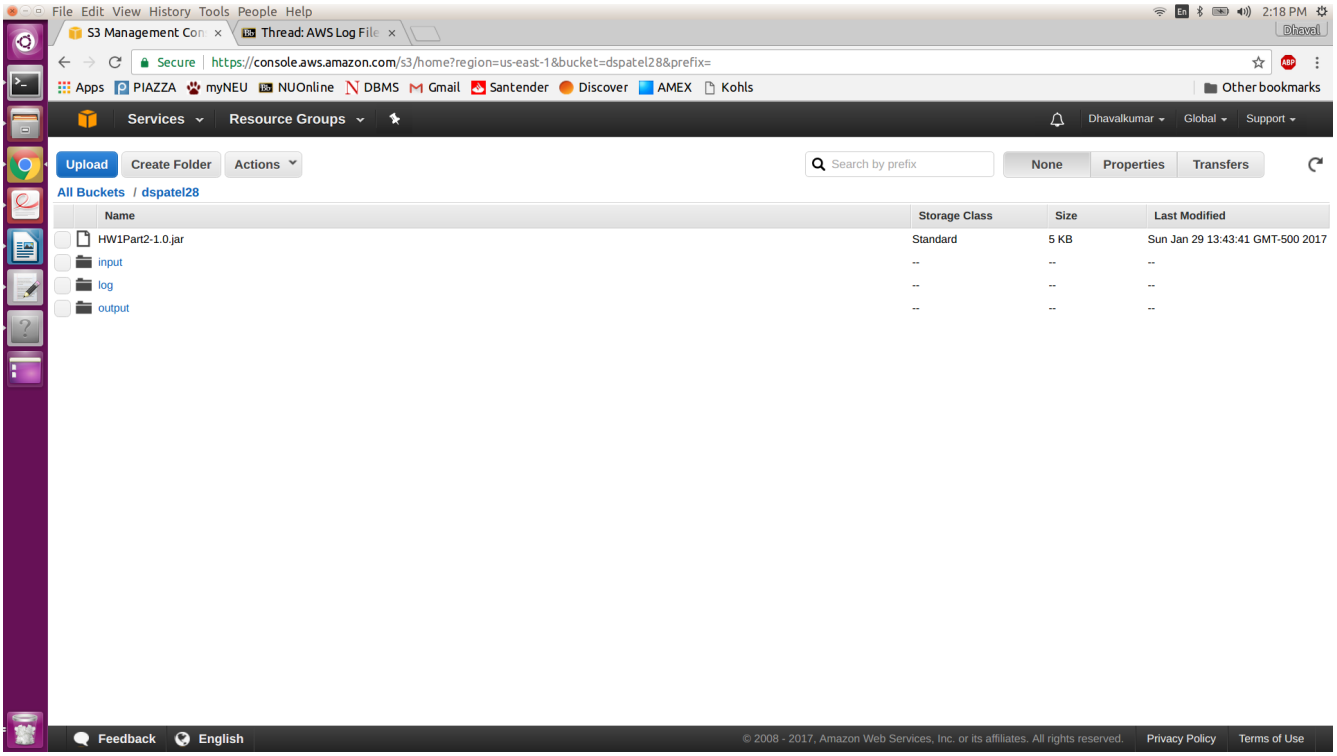
3 Word Count AWS Execution



You can see 1 Master and 3 worker nodes in below image:



Bucket after successful run:



Result files after successful run: