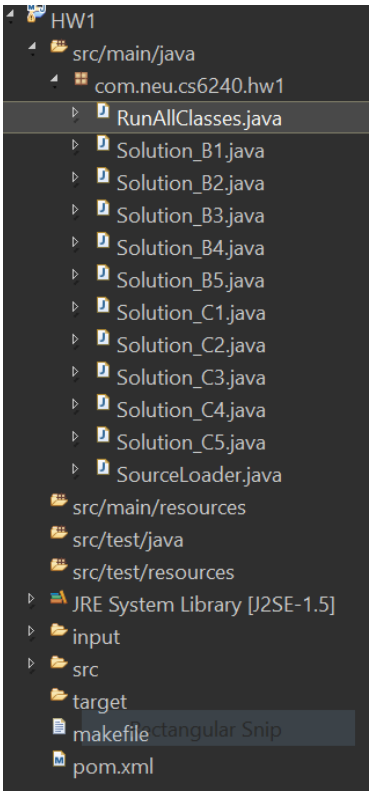
Ankit Sharma
Homework 1
CS6240 Section-01

Deliverables:

1. Source Code: As explained here
2. Report: This file
3. Log files: Please find the folder log and logs in the current folder.
4. Final result AWS: Please find the folder output with the final results of the AWS run. This folder also contains the output produced by weather data source code.

Weather Data Source Code

Kindly find the source code in HW1 folder. It is a maven project with the following structure:



To compile the entire project just type in “make” once you are inside the HW1 folder.

To run the project kindly type inside the HW1 folder

- **make local filename=[fileName of the file which will be used as input]**

This will run the RunAllClasses file which will run all the rest of the classes in the project.

Ankit Sharma
Homework 1
CS6240 Section-01

To run the emr cluster kindly type inside the HW1 folder

- **make emr jarFile=[Jar which needs to be run] inputFile=[Input file on which the jar needs to do the processing]**

This will run the aws emr for the given jarfile and the given inputfile.

This is all that is needed to be run the project and the emr.

PS: for emr I used --profile setting to set my aws cli profile.

Also if you want you can run individual files but kindly provide the input file name as argument.

Weather Data Results

Minimum, Maximum & Average Running Time

All running times are in milliseconds

B.1. Sequential Run

Minimum Running Time	Maximum Running Time	Average Running Time
2601.0	5953.0	3024.0

B.2. No Lock

Minimum Running Time	Maximum Running Time	Average Running Time
1314.0	2111.0	1475.0

B.3. Coarse Lock

Minimum Running Time	Maximum Running Time	Average Running Time
2936.0	6075.0	3296.0

B.4. Fine Lock

Minimum Running Time	Maximum Running Time	Average Running Time
1445.0	2569.0	1617.0

Ankit Sharma
Homework 1
CS6240 Section-01

B.5. No sharing

Minimum Running Time	Maximum Running Time	Average Running Time
1327.0	4626.0	1709.0

C.1. Sequential Run

Minimum Running Time	Maximum Running Time	Average Running Time
2687.0	5767.0	3035.0

C.2. No Lock

Minimum Running Time	Maximum Running Time	Average Running Time
1307.0	4561.0	1684.0

C.3. Coarse Lock

Minimum Running Time	Maximum Running Time	Average Running Time
2990.0	6274.0	3409.0

C.4. Fine Lock

Minimum Running Time	Maximum Running Time	Average Running Time
1375.0	4714.0	1761.0

C.5. No sharing

Minimum Running Time	Maximum Running Time	Average Running Time
1361.0	4626.0	1764.0

Ankit Sharma
Homework 1
CS6240 Section-01

Worker Threads used and Speedup

B.1. Sequential Run

Worker Threads	Speedup
1	N/A

B.2. No Lock

Worker Threads	Speedup
4	2.05

B.3. Coarse Lock

Worker Threads	Speedup
4	0.917

B.4. Fine Lock

Worker Threads	Speedup
4	1.87

B.5. No share

Worker Threads	Speedup
4	1.769

C.1. Sequential Run

Worker Threads	Speedup
1	N/A

C.2. No Lock

Worker Threads	Speedup
4	1.80

C.3. Coarse Lock

Worker Threads	Speedup
4	0.89

Ankit Sharma
Homework 1
CS6240 Section-01

C.4. Fine Lock

Worker Threads	Speedup
4	1.723

C.5. No share

Worker Threads	Speedup
4	1.720

1. The No-Lock version should end the fastest as all the threads can access the data structure at the same time. This ensures that the No-Lock will be fastest but the result obtained will not be correct. The experiments conform to this.

2. The Coarse-Lock version should end the slowest because in this case the data structure is getting locked whenever one of the threads get access to it. Once this thread gets done with the data structure it notifies other threads that they can now work on it, causing delays. The experiments conform to this.

3. The average temperatures returned by No-Lock returns incorrect values. The reason behind this is because all the threads can simultaneously update the data structure hence causing data errors.

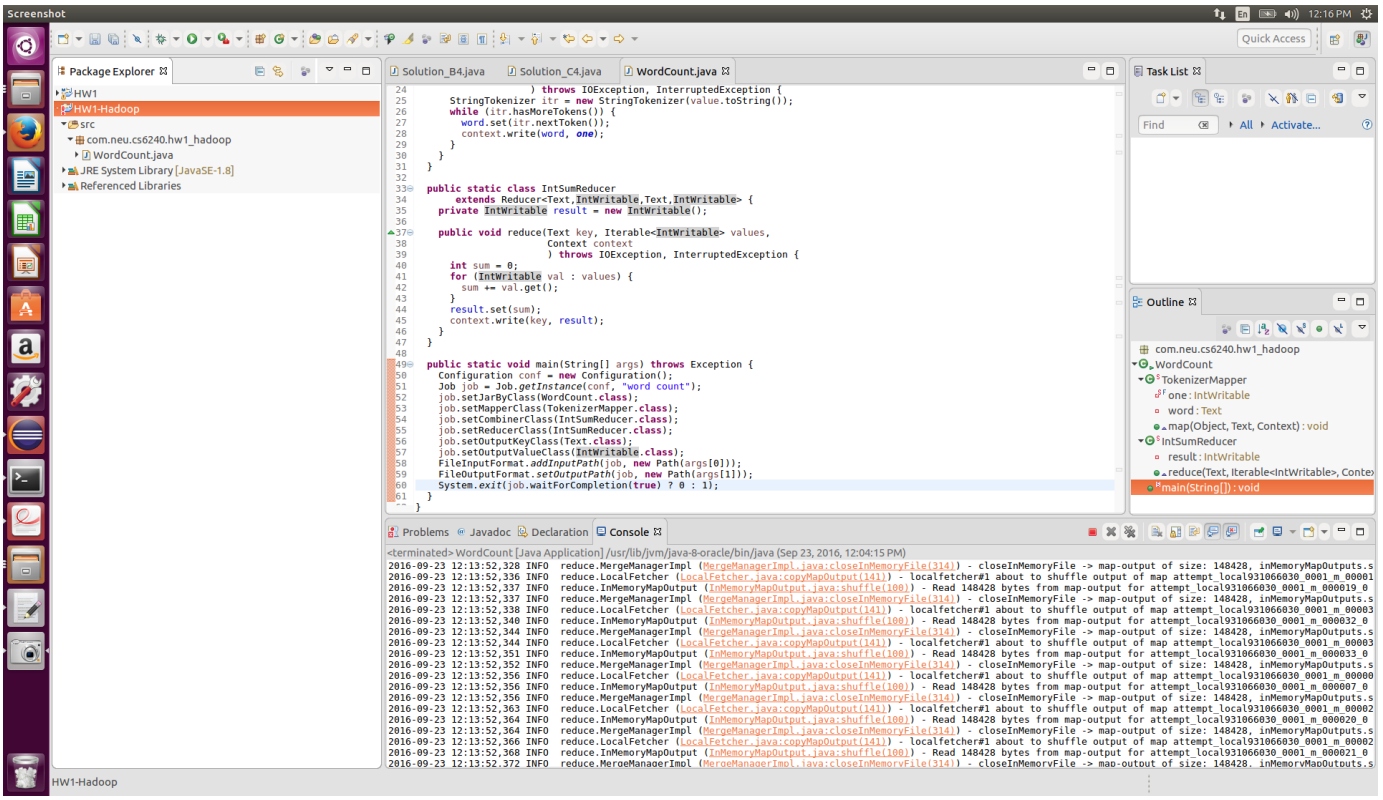
4. Coarse Lock is slower since once a thread gets access to the data structure it blocks access to the data structure for other threads. One it gets done with its work, it notifies other threads that they can now use the data structure. This waking up of threads causes delays. Where as in Sequential run, only one thread works on the data structure and hence no delays are causes due to blocking.

5. The extra computation in C slows down the Fine-locking a bit as compared to Coarse-Locking. This is because for Fine-locking the Fibonacci computation happens inside the lock thereby limiting the access to the data structure whereas in Coarse-Locking in B or C the lock is on the entire data structure thereby not affecting its performance.

Ankit Sharma
Homework 1
CS6240 Section-01

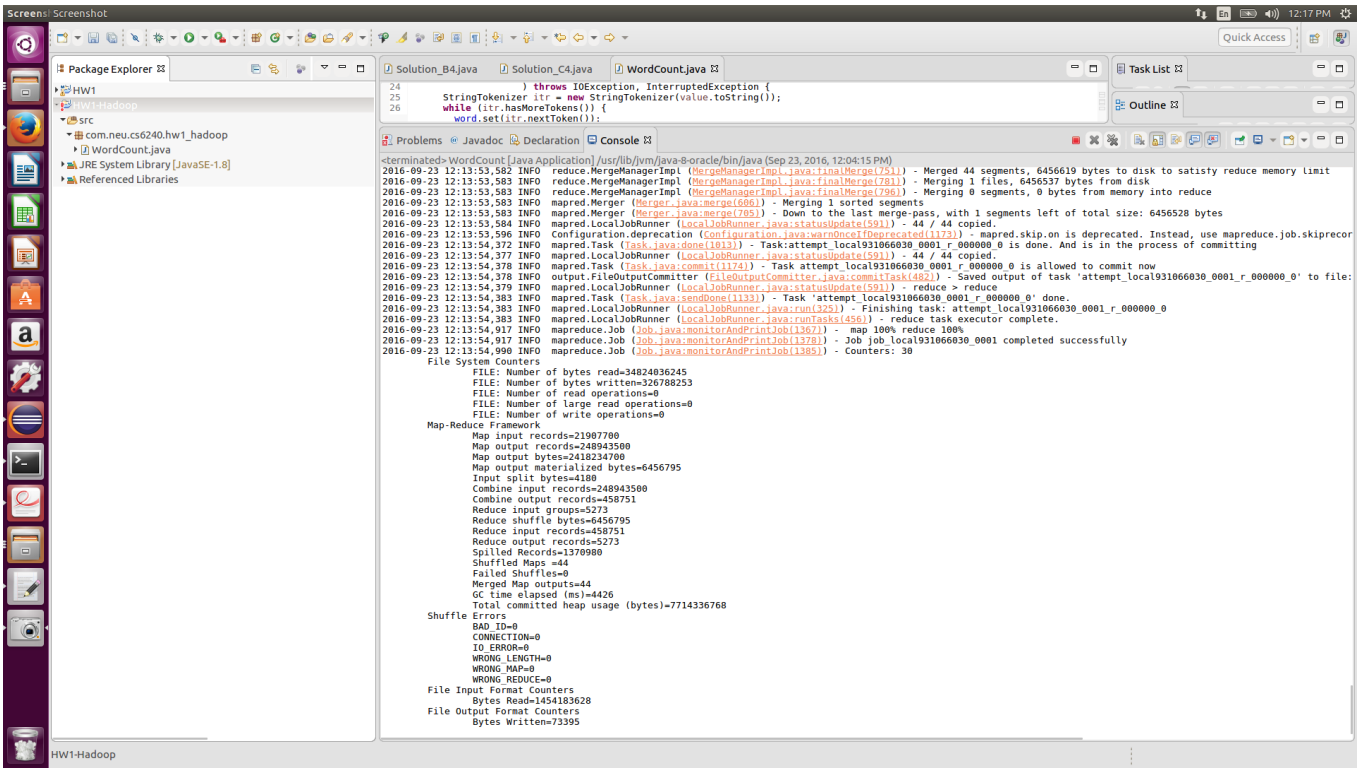
Word Count Local Execution

1. Project Structure:



Ankit Sharma
Homework 1
CS6240 Section-01

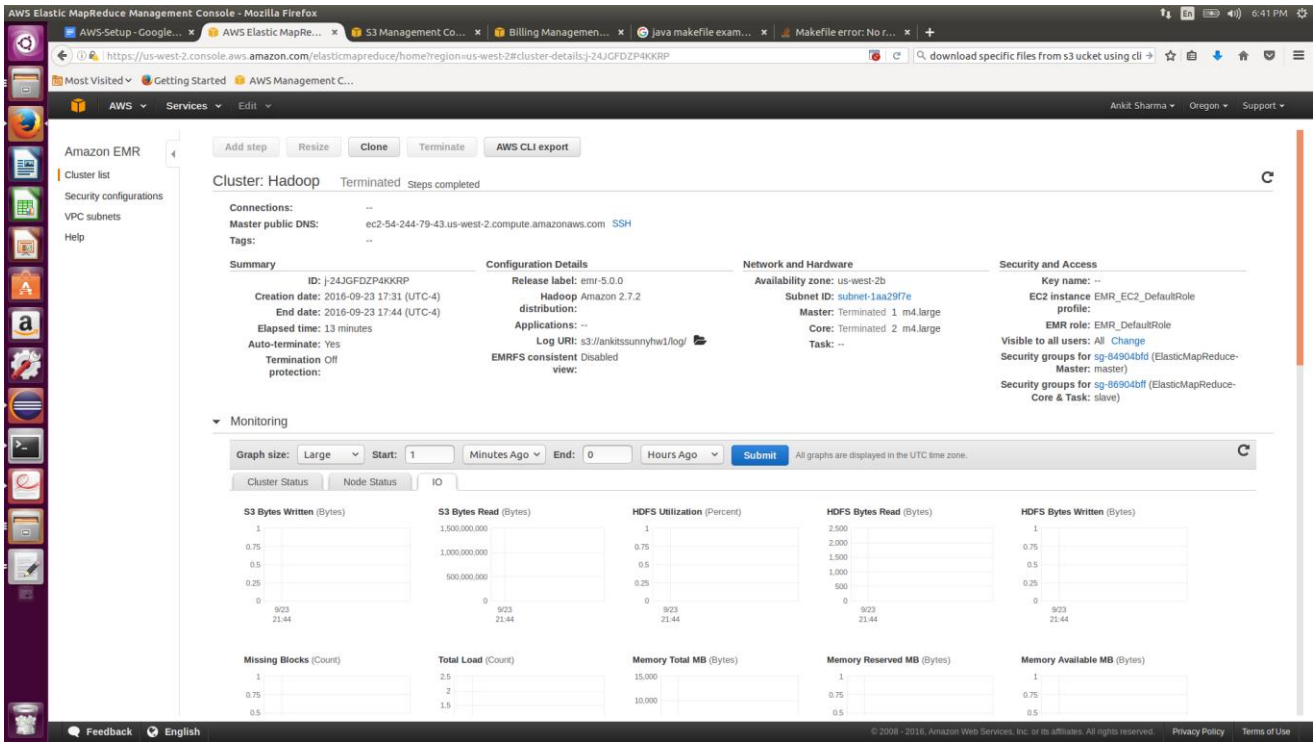
2. Console Output:



Ankit Sharma
Homework 1
CS6240 Section-01

Word Count AWS Execution

1. Screenshot of successful run



2. Find the log files and the output in the Assignment Folder.