# CS-6240 HW-1 Report

**Ajay Baban Kauthale(ajayk@ccs.neu.edu)**

## Weather Data Source Code

Source code is in “HW01/1. Source Code” directory

## Weather Data Results

### A) Running times without Fibonacci

| Model | Min | Max | Avg |
|---|---|---|---|
| Sequential | 2624 | 6224 | 3088 |
| No Lock | 1674 | 5264 | 2083 |
| Coarse Lock | 2103 | 5776 | 2512 |
| Fine Lock | 1667 | 6437 | 2271 |
| No Sharing | 1819 | 5176 | 2364 |

### B) Running times with Fibonacci

| Model | Min | Max | Avg |
|---|---|---|---|
| Sequential | 12730 | 16553 | 13262 |
| No Lock | 6600 | 11598 | 7394 |
| Coarse Lock | 8135 | 12000 | 8603 |
| Fine Lock | 6585 | 11598 | 7535 |
| No Sharing | 6985 | 11905 | 8349 |

### C) Number of threads: 4 (dual core processor, 4 logical CPU’s)

**Speed Up:**

| Model | Speed up without delay | Speed up with delay |
|---|---|---|
| No Lock | 3.33 | 2.46 |
| Coarse Lock | 3.13 | 2.34 |
| Fine Lock | 3.56 | 2.41 |
| No Sharing | 2.95 | 2.26 |

### D) Questions:

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.

⇨ **No Lock** should be the fastest one.

⇨ The reason I think No Lock should be fastest is because it does not lock data in any way. Because of no data locking each thread can access data simultaneously. Although it does not guarantee correct output, it is fastest one which is confirmed by above results and it meets the expectation.

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.

- ⇨ **Sequential** should be the slowest one.
- ⇨ The reason I think Sequential should be slowest one is because it is not concurrent and single process reading and processing data sequentially. Hence, it should be slowest as compared to any other multithreaded version which is confirmed by my experiment.

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.

- ⇨ During the experiment, no program is crashed. Only discrepancy I noticed is whenever I start other applications (browser etc.) while running my experiment all running times go high. Especially **No Lock** gives different result for every other run.
- ⇨ This seems expected since applications might be competing for the system resources. Also, **No Lock** do not have synchronization in place which is causing different results.

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.)

- ⇨ **Sequential** should be slower that **Coarse Lock** because as we discussed above Sequential process all data one by one in single process which is time consuming. On the other hand, Coarse Lock uses concurrent threads to process that data making it faster than that of the Sequential.
- ⇨ Experiment confirms my understanding and hypothesis is true. The results are as follows

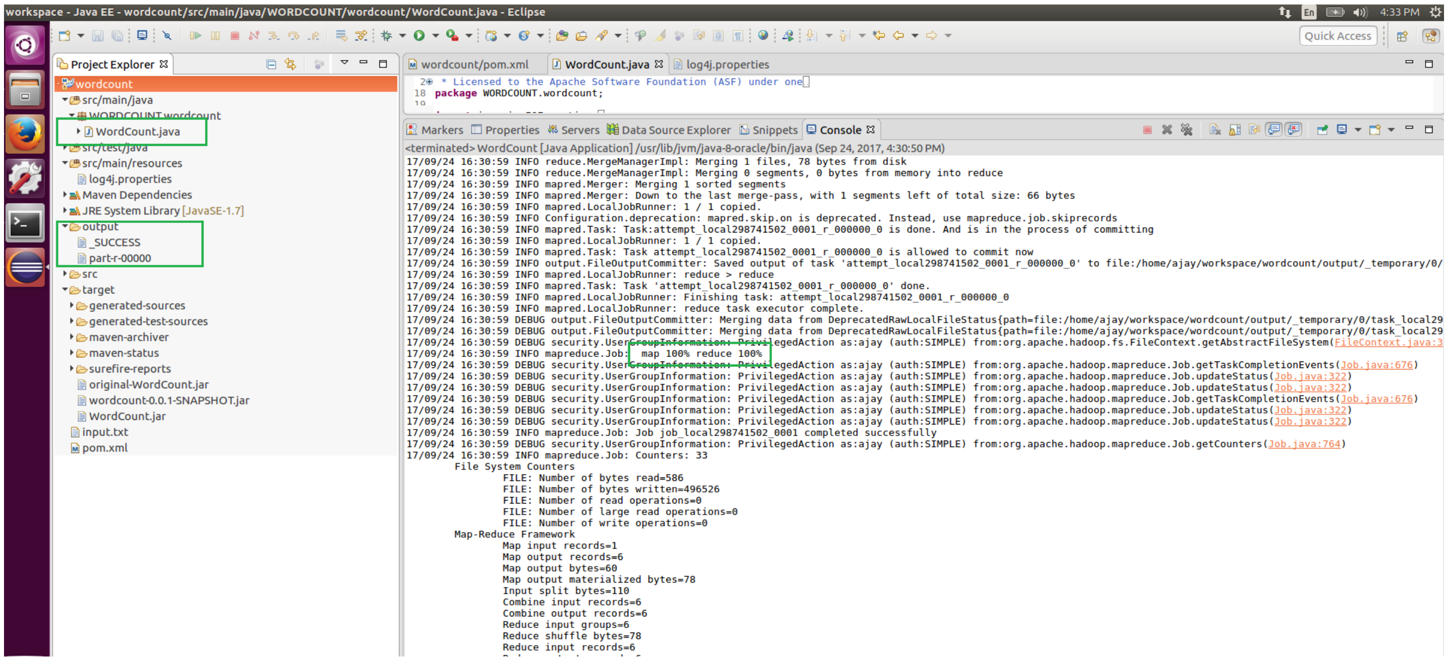
| **Model** | Avg time without Fibonacci | Avg time with Fibonacci |
|---|---|---|
| Sequential | 3088 | 13262 |
| Coarse Lock | 2512 | 8603 |

**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.

- ⇨ Both **Coarse Lock** and **Fine Lock** running times increased with Fibonacci computation.
- ⇨ For the **Coarse Lock**, the difference is slightly greater as compared to **Fine Lock** since all data is locked in **Coarse Lock** instead of station specific data.
- ⇨ For the **Fine Lock**, the station data is locked instead of all data. Therefore, insertion and processing is faster as compared to the **Coarse Lock.**

## Word Count Local Execution (UBUNTU)

Following screenshot contains both directory structure and console output (> 20 lines)

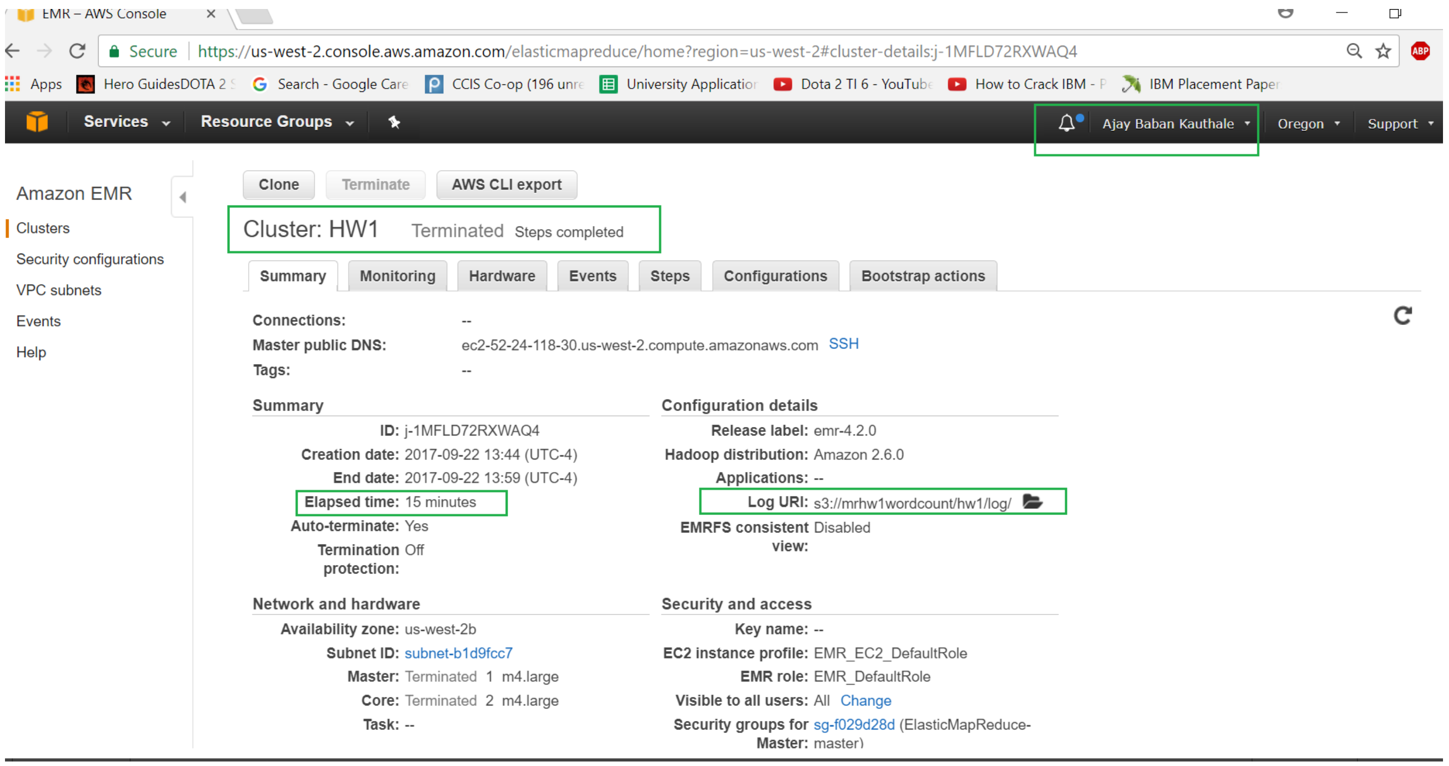


```
 * Licensed to the Apache Software Foundation (ASF) under one
package WORDCOUNT.wordcount;
```

```
<terminated> WordCount [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (Sep 24, 2017, 4:30:50 PM)
17/09/24 16:30:59 INFO reduce.MergeManagerImpl: Merging 1 files, 78 bytes from disk
17/09/24 16:30:59 INFO reduce.MergeManagerImpl: Merging 0 segments, 0 bytes from memory into reduce
17/09/24 16:30:59 INFO mapred.Merger: Merging 1 sorted segments
17/09/24 16:30:59 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left of total size: 66 bytes
17/09/24 16:30:59 INFO mapred.LocalJobRunner: 1 / 1 copied.
17/09/24 16:30:59 INFO Configuration.deprecation: mapred.skip.on is deprecated. Instead, use mapreduce.job.skiprecords
17/09/24 16:30:59 INFO mapred.Task: Task:attempt_local298741502_0001_r_000000_0 is done. And is in the process of committing
17/09/24 16:30:59 INFO mapred.LocalJobRunner: 1 / 1 copied.
17/09/24 16:30:59 INFO mapred.Task: Task attempt_local298741502_0001_r_000000_0 is allowed to commit now
17/09/24 16:30:59 INFO output.FileOutputCommitter: Saved output of task 'attempt_local298741502_0001_r_000000_0' to file:/home/ajay/workspace/wordcount/output/_temporary/0/
17/09/24 16:30:59 INFO mapred.LocalJobRunner: reduce > reduce
17/09/24 16:30:59 INFO mapred.Task: Task 'attempt_local298741502_0001_r_000000_0' done.
17/09/24 16:30:59 INFO mapred.LocalJobRunner: Finishing task: attempt_local298741502_0001_r_000000_0
17/09/24 16:30:59 INFO mapred.LocalJobRunner: reduce task executor complete.
17/09/24 16:30:59 DEBUG output.FileOutputCommitter: Merging data from DeprecatedRawLocalFileStatus{path=file:/home/ajay/workspace/wordcount/output/_temporary/0/task_local29
17/09/24 16:30:59 DEBUG output.FileOutputCommitter: Merging data from DeprecatedRawLocalFileStatus{path=file:/home/ajay/workspace/wordcount/output/_temporary/0/task_local29
17/09/24 16:30:59 DEBUG security.UserGroupInformation: PrivilegedAction as:ajay (auth:SIMPLE) from:org.apache.hadoop.fs.FileContext.getAbstractFileSystem(FileContext.java:3
17/09/24 16:30:59 INFO mapreduce.Job:  map 100% reduce 100%
17/09/24 16:30:59 DEBUG security.UserGroupInformation: PrivilegedAction as:ajay (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.getTaskCompletionEvents(Job.java:676)
17/09/24 16:30:59 DEBUG security.UserGroupInformation: PrivilegedAction as:ajay (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.updateStatus(Job.java:322)
17/09/24 16:30:59 DEBUG security.UserGroupInformation: PrivilegedAction as:ajay (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.updateStatus(Job.java:322)
17/09/24 16:30:59 DEBUG security.UserGroupInformation: PrivilegedAction as:ajay (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.getTaskCompletionEvents(Job.java:676)
17/09/24 16:30:59 DEBUG security.UserGroupInformation: PrivilegedAction as:ajay (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.updateStatus(Job.java:322)
17/09/24 16:30:59 DEBUG security.UserGroupInformation: PrivilegedAction as:ajay (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.updateStatus(Job.java:322)
17/09/24 16:30:59 INFO mapreduce.Job: Job job_local298741502_0001 completed successfully
17/09/24 16:30:59 DEBUG security.UserGroupInformation: PrivilegedAction as:ajay (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.getCounters(Job.java:764)
17/09/24 16:30:59 INFO mapreduce.Job: Counters: 33
        File System Counters
                FILE: Number of bytes read=586
                FILE: Number of bytes written=496526
                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=1
                Map output records=6
                Map output bytes=60
                Map output materialized bytes=78
                Input split bytes=110
                Combine input records=6
                Combine output records=6
                Reduce input groups=6
                Reduce shuffle bytes=78
                Reduce input records=6
```

## Word Count AWS Execution

Following screenshot contains cluster screenshot with three machines (1 master and 2 workers) and amazon S3 storage.



Cluster: HW1 Terminated Steps completed

Connections: --
Master public DNS: ec2-52-24-118-30.us-west-2.compute.amazonaws.com SSH
Tags: --

**Summary**

ID: j-1MFLD72RXWAQ4
Creation date: 2017-09-22 13:44 (UTC-4)
End date: 2017-09-22 13:59 (UTC-4)
Elapsed time: 15 minutes
Auto-terminate: Yes
Termination protection: Off

**Configuration details**

Release label: emr-4.2.0
Hadoop distribution: Amazon 2.6.0
Applications: --
Log URI: s3://mrhw1wordcount/hw1/log/
EMRFS consistent view: Disabled

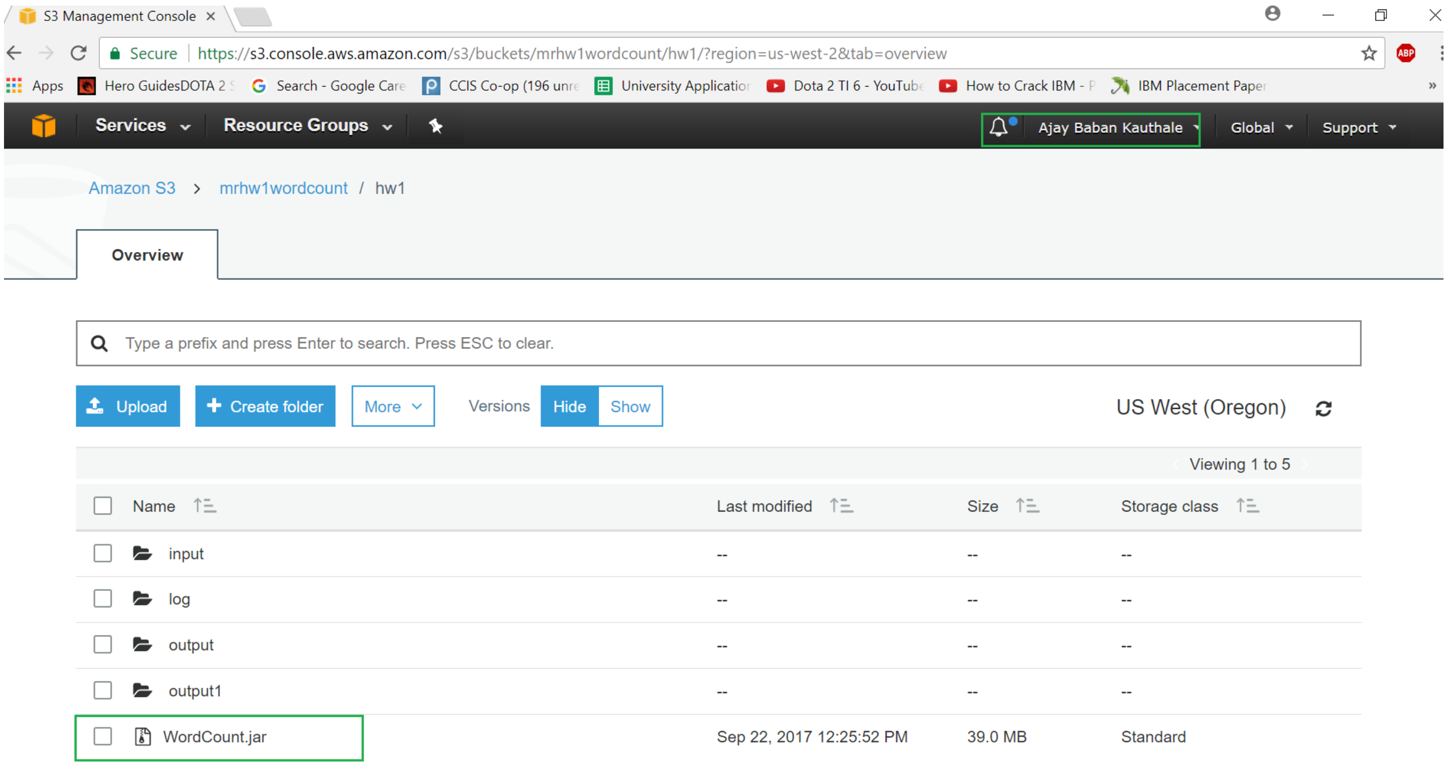
**Network and hardware**

Availability zone: us-west-2b
Subnet ID: subnet-b1d9fcc7
Master: Terminated 1 m4.large
Core: Terminated 2 m4.large
Task: --

**Security and access**

Key name: --
EC2 instance profile: EMR_EC2_DefaultRole
EMR role: EMR_DefaultRole
Visible to all users: All Change
Security groups for Master: sg-f029d28d (ElasticMapReduce-master)

## Amazon S3


S3 Management Console
Secure https://s3.console.aws.amazon.com/s3/buckets/mrhw1wordcount/hw1/?region=us-west-2&tab=overview
Apps Hero GuidesDOTA 2 Search - Google Care CCIS Co-op (196 unre University Applicatior Dota 2 TI 6 - YouTube How to Crack IBM - P IBM Placement Paper
Services Resource Groups
Ajay Baban Kauthale Global Support
Amazon S3 › mrhw1wordcount / hw1
Overview
Type a prefix and press Enter to search. Press ESC to clear.
Upload Create folder More Versions Hide Show
US West (Oregon)
Viewing 1 to 5

| Name | Last modified | Size | Storage class |
|---|---|---|---|
| input | -- | -- | -- |
| log | -- | -- | -- |
| output | -- | -- | -- |
| output1 | -- | -- | -- |
| WordCount.jar | Sep 22, 2017 12:25:52 PM | 39.0 MB | Standard |