

## Weather Data Results

### VERSION B

#### Runtimes:

##### 1. Sequential

maximum time 1606

minimum time 1019

average time 1155.8

##### 2. No Lock

No lock maximum time 1288

No lock minimum time 368

No lock average time 486.9

SpeedUp: 2.37

##### 3. Coarse Lock

Coarse lock maximum time 1463

Coarse lock minimum time 510

Coarse lock average time 641.0

SpeedUp: 1.8

##### 4. Fine Lock

Fine lock maximum time 1539

Fine lock minimum time 365

Fine lock average time 526.2

SpeedUp: 2.19

##### 5. No Sharing

No Sharing maximum time 1288

No Sharing minimum time 401

No Sharing average time 553.5

SpeedUp: 2.08

No of threads used in my laptop = 4

### VERSION C

#### Runtimes with Fibonacci

##### 1. Sequential

Sequential maximum time 14327

Sequential minimum time 11135

Sequential average time 11879.0

2. No Lock  
No lock maximum time 10156  
No lock minimum time 4286  
No lock average time 4922.5  
SpeedUp: 2.413
3. Coarse Lock  
Coarse lock maximum time 8026  
Coarse lock minimum time 4398  
Coarse lock average time 4994.6  
SpeedUp: 2.37
4. Fine Lock  
Fine lock maximum time 9775  
Fine lock minimum time 4274  
Fine lock average time 5057.2  
SpeedUp: 2.34
5. No Sharing  
No Sharing maximum time 12538  
No Sharing minimum time 4556  
No Sharing average time 5800.4  
SpeedUp: 2.04

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.

Answer: I would normally expect No Lock version to finish the fastest as all 4 threads work simultaneously and no thread waits for the other record to update the shared data structure. But there might be data inconsistency in case of no lock and multiple threads are trying to update the same record. My experiments also show that No Lock version is the fastest.

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 would expect SEQ to finish slowest as all the 8mil records in the file are getting processed by the same thread one by one. My experiments also prove that SEQ finishes slowest.

3. Compare the temperature averages returned by each program version. Report if any of them is incorrect

Answer: NO lock does not have correct entries for temperature averages as multiple thread try to update the same data structure (maybe same value) at the same time. Also FINE LOCK

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: SEQ takes almost double the time than COARSE-LOCK in both the B and C versions. This is because while SEQ will always perform the execution of a per record basis, COARSE-LOCK might in some cases allow for slightly faster data processing.

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: According to my experiments FINE-LOCK takes approx 50ms more than COARSE-LOCK when Fibonacci computation is added whereas it was taking about 100ms less than COARSE-LOCK.

## Word Count Local Execution

The screenshot shows the Eclipse IDE with the following components:

- Package Explorer:** Shows the project structure with 'simple.hadoop' containing 'src/main/java' and 'com.simple.hadoop' with 'WordCount.java'.
- WordCount.java:**

```

37     extends Reducer<Text,IntWritable,Text,IntWritable> {
38     private IntWritable result = new IntWritable();
39

```
- Console:** Displays the execution output of the WordCount application.
 

```

<terminated> WordCount [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_92.jdk/Contents/Home/bin/java (Sep 24, 2016, 11:14:21 PM)
2016-09-24 23:19:36,325 INFO mapreduce.Job (Job.java:monitorAndPrintJob(1385)) - Job job_local450006795_0001 completed successfully
2016-09-24 23:19:36,354 INFO mapreduce.Job (Job.java:monitorAndPrintJob(1385)) - Counters: 30

File System Counters
  FILE: Number of bytes read=34824437297
  FILE: Number of bytes written=327460733
  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=21907700
  Map output records=248943500
  Map output bytes=2418234700
  Map output materialized bytes=6456795
  Input split bytes=5236
  Combine input records=248943500
  Combine output records=458751
  Reduce input groups=5273
  Reduce shuffle bytes=6456795
  Reduce input records=458751
  Reduce output records=5273
  Spilled Records=1370980
  Shuffled Maps =44
  Failed Shuffles=0
  Merged Map outputs=44
  GC time elapsed (ms)=2764
  Total committed heap usage (bytes)=57347145728

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=1454183628
File Output Format Counters
  Bytes Written=73395

```

## Word Count AWS Execution

The screenshot shows the AWS Management Console with the following details:


- Navigation:** AWS Services, EMR, S3, EC2, Edit.
- Amazon EMR:** Cluster list, Security configurations, VPC subnets, Help.
- Create cluster:** View details, Clone, Terminate.
- Filter:** All clusters (11 clusters (all loaded)).
- Cluster List:**

Name	ID	Status	Creation time (UTC-4)	Elapsed time
My cluster	j-HD0VSN6BO4G3	Terminated All steps completed	2016-09-25 02:58 (UTC-4)	15 minutes
- Summary:**
  - Master: ec2-54-166-0-144.compute-1.amazonaws.com
  - Termination protection: Off
  - Tags: --
- Hardware:**
  - Master: Terminated 1 m3.xlarge
  - Core: Terminated 2 m3.xlarge
  - Task: --
- Steps:**

Name	Status	Start time (UTC-4)	Elapsed time
Custom JAR	Completed	2016-09-25 03:06 (UTC-4)	6 minutes
Setup hadoop debugging	Completed	2016-09-25 03:05 (UTC-4)	2 seconds
- Bootstrap /** No boo
- Links:** View cluster details, View monitoring details.

## Custom JAR

**Status:** Completed**Start time:** 2016-09-25 03:06 (UTC-4)**ID:** s-WZWSD5W29WER**Elapsed time:** 6 minutes**Log files:** [controller](#) | [syslog](#) | stderr\* | stdout\* **JAR location:** s3://naomi-hw1/hadoop.jar**Main class:** None**Arguments:** s3://naomi-hw1/input/hw1.txt s3://naomi-hw1/output**Action on failure:** Terminate cluster**Jobs****Jobs for:** s-WZWSD5W29WER

Filter: <input type="text"/>				
Job	State	Start time (UTC-4)	Actions	
job_1474786896305_0001	COMPLETED	2016-09-25 03:06 (UTC-4)	<a href="#">View tasks</a>	