

Week 1**Weather Data Result**

The following output results are found out based on the input file – 1912.csv

Multi-threaded Versions – 2 numbers of worker thread.

Code	Time(in ms)	MinTime(in ms)	MaxTime(in ms)	AverageTime(in ms)	SpeedUp
Seq	6999	5606	6999	5822.4	1
	5771				
	5700				
	5691				
	5734				
	5606				
	5781				
	5637				
	5651				
	5654				
NoLock	4824	2234	5245	4325.6	1.35
	4495				
	2457				
	4844				
	2234				
	4651				
	5012				
	5245				
	4698				
	4796				
CoarseLock	5447	5121	5528	5312	1.1
	5246				
	5528				
	5408				
	5353				
	5121				
	5262				
	5198				
	5386				
	5171				
FineLock	2559	2559	5101	4641.7	1.25
	5101				
	4998				
	4948				

HOMEWORK 1

CS6240 Parallel Data Processing – Map Reduce

Rohit Patnaik

	5007				
	4918				
	4898				
	4620				
	4626				
	4742				
NoSharing	4987	4674	5148	4842.8	1.21
	4942				
	4800				
	4735				
	4799				
	4742				
	4674				
	5148				
	4789				
	4812				
SeqV2(Fib)	6499	5751	6499	5986.6	1
	5884				
	5913				
	5862				
	6100				
	5892				
	5751				
	5934				
	6096				
	5935				
NoLockV2(Fib)	2318	2318	5167	4675.4	1.28
	4864				
	4880				
	4812				
	4754				
	4963				
	4849				
	5116				
	5167				
	5031				
CoarseLockV2(Fib)	6308	5123	6695	5603.2	1.07
	5663				
	5123				
	5995				
	5286				
	5263				

HOMEWORK 1

CS6240 Parallel Data Processing – Map Reduce

Rohit Patnaik

	6695				
	5253				
	5241				
	5205				
FineLockV2(Fib)	4869	4322	5249	4796.3	1.25
	4752				
	5122				
	4654				
	5249				
	4698				
	4322				
	4712				
	4597				
	4988				
NoSharingV2(Fib)	5277	4543	5510	5004.8	1.2
	5141				
	5062				
	5064				
	4993				
	4875				
	5510				
	4707				
	4543				
	4876				

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.

I would expect NO-LOCK version to run the fastest since it's a multithreaded version and it doesn't use any kind of lock which wouldn't cause any delay time.

Yes, the experiment did confirm my expectations.

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.

I would expect SEQ version to be the slowest. However, there might be a case where COARSE-LOCK might be the slowest. It would depend on the input data. More the data with same key, would result in locking of the HashMap would cause delay.

Yes, the experiment did 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.**

The temperature averages are same in SEQ, COARSE-LOCK, FINE-LOCK and NO-SHARING versions. I found incorrect results in the NO-LOCK version because of two threads accessing data with no-lock, hence causing DIRTY-READ and incorrect update of value in the HashMap.

- 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.)**

The running time of SEQ and COARSE-LOCK are pretty much similar in my experiment with COARSE-LOCK just edging above SEQ with SPEED UP of 1.10(Case B) and 1.07(Case C).

The reason SEQ might be slower than COARSE-LOCK are –

SEQ version runs sequentially while COARSE-LOCK version tries to run parallelly. Since, the lock is on HashMap, there is a delay in time execution. Also, the time delay would be more if there is a lot of data per key since the lock is in place during the update.

- 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.**

The additional Fibonacci computation affect both COARSE-LOCK and FINE-LOCK by increasing the running time due to an extra functionality being executed.

In my experiment, the difference between COARSE-LOCK and FINE-LOCK increases as compared to CASE B (Can be verified seeing the difference in SPEED UP in the above table) which tells us, since COARSE LOCK is held for longer time, the time of execution also increases.

WEEK 2

Word Count Local Execution

Project Directory Structure:

```
rohit@rohit-VirtualBox: ~/eclipse-workspace/wc/src
rohit@rohit-VirtualBox:~$ ls
a.out  Desktop  Downloads  eclipse  examples.desktop  hadoop-2.7.4.tar.gz.nds  input_file.txt  Pictures  sun.c  tree-structure.txt  wordcount_example  wordcount_res
rohit@rohit-VirtualBox:~$ cd eclipse-workspace
rohit@rohit-VirtualBox:~/eclipse-workspace$ ls
wc
rohit@rohit-VirtualBox:~/eclipse-workspace$ cd wc
rohit@rohit-VirtualBox:~/eclipse-workspace/wc$ ls
input  output_res  pon.xml  src  target
rohit@rohit-VirtualBox:~/eclipse-workspace/wc$ cd src
rohit@rohit-VirtualBox:~/eclipse-workspace/wc/src$ tree
.
├── main
│   ├── java
│   │   ├── cs6240
│   │   │   └── wc
│   │   │       ├── App.java
│   │   │       └── WordCount.java
│   └── test
│       ├── java
│       │   ├── cs6240
│       │   │   └── wc
│       │   │       └── AppTest.java
└── 8 directories, 3 files
rohit@rohit-VirtualBox:~/eclipse-workspace/wc/src$
```

HOMEWORK 1

CS6240 Parallel Data Processing – Map Reduce

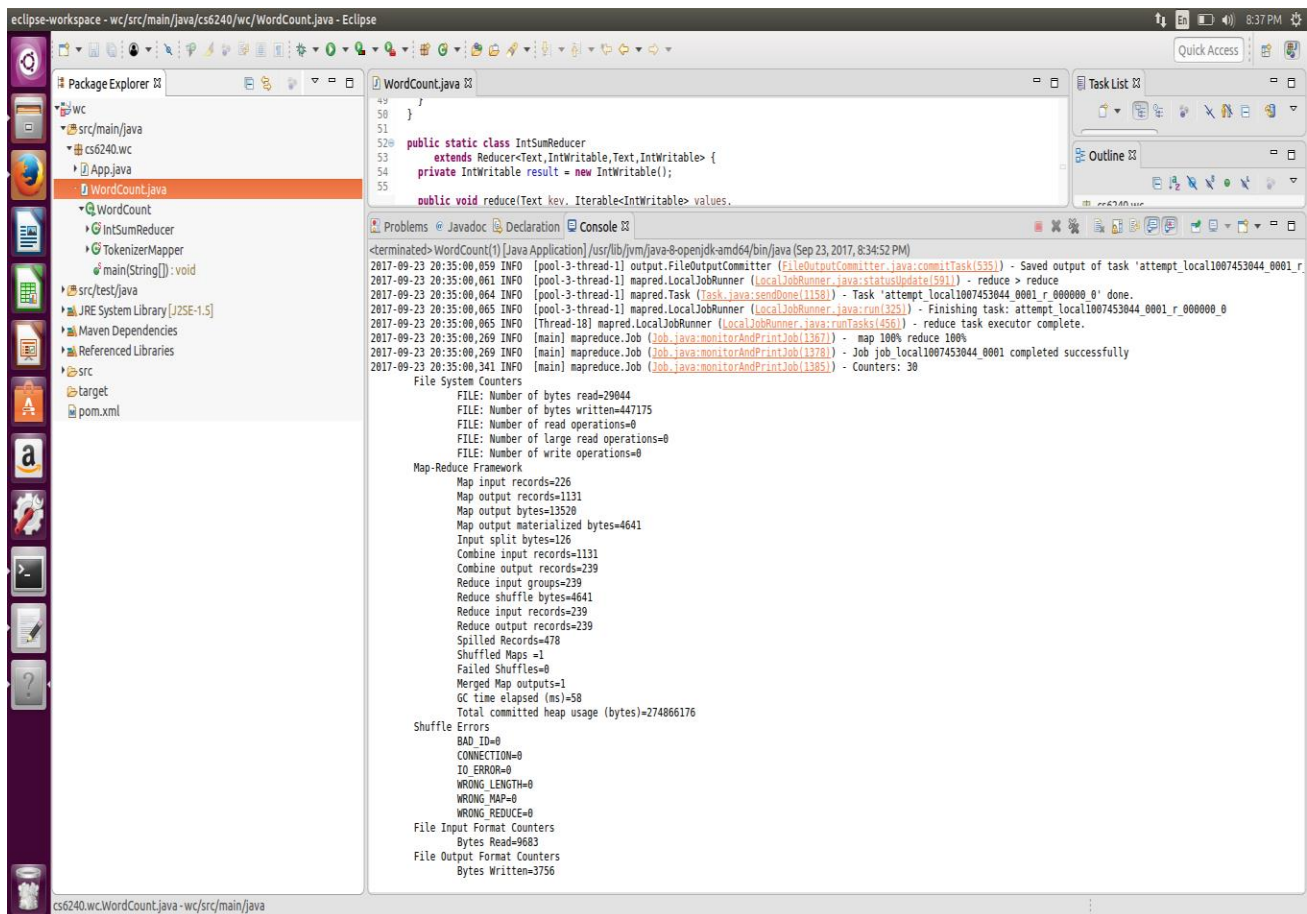
Rohit Patnaik

Output:

Please note, I couldn't run locally on the given data file in HW1, i.e., hw1.txt due to disk space issue on my Linux VM.

I executed it locally on a randomly large file.

From Eclipse Editor:



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure: `src/main/java` containing `cs6240.wc` and `App.java`. The `WordCount.java` file is selected. The main editor shows the `IntSumReducer` class with the following code:

```
49 }
50
51 public static class IntSumReducer
52     extends Reducer<Text, IntWritable, Text, IntWritable> {
53     private IntWritable result = new IntWritable();
54
55     public void reduce(Text key, Iterable<IntWritable> values,
```

The Console window at the bottom shows the execution output for the MapReduce job. The output includes the following information:

```
<terminated> WordCount(1) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Sep 23, 2017, 8:34:52 PM)
2017-09-23 20:35:00,059 INFO [pool-3-thread-1] output.FileOutputCommitter (FileOutputCommitter.java:commitTask(535)) - Saved output of task 'attempt_local1007453044_0001_r
2017-09-23 20:35:00,061 INFO [pool-3-thread-1] mapred.LocalJobRunner (LocalJobRunner.java:statusUpdate(591)) - reduce > reduce
2017-09-23 20:35:00,064 INFO [pool-3-thread-1] mapred.Task (Task.java:sendDone(1158)) - Task 'attempt_local1007453044_0001_r_000000_0' done.
2017-09-23 20:35:00,065 INFO [pool-3-thread-1] mapred.LocalJobRunner (LocalJobRunner.java:run(325)) - Finishing task: attempt_local1007453044_0001_r_000000_0
2017-09-23 20:35:00,065 INFO [Thread-18] mapred.LocalJobRunner (LocalJobRunner.java:runTasks(456)) - reduce task executor complete.
2017-09-23 20:35:00,269 INFO [main] mapreduce.Job (Job.java:monitorAndPrintJob(1367)) - map 100% reduce 100%
2017-09-23 20:35:00,269 INFO [main] mapreduce.Job (Job.java:monitorAndPrintJob(1378)) - Job job_local1007453044_0001 completed successfully
2017-09-23 20:35:00,341 INFO [main] mapreduce.Job (Job.java:monitorAndPrintJob(1385)) - Counters: 30

File System Counters
  FILE: Number of bytes read=29044
  FILE: Number of bytes written=447175
  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=226
  Map output records=1131
  Map output bytes=13520
  Map output materialized bytes=4641
  Input split bytes=126
  Combine input records=1131
  Combine output records=239
  Reduce input groups=239
  Reduce shuffle bytes=4641
  Reduce input records=239
  Reduce output records=239
  Spilled Records=478
  Shuffled Maps=1
  Failed Shuffles=0
  Merged Map outputs=1
  GC time elapsed (ms)=58
  Total committed heap usage (bytes)=274866176

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=9683
File Output Format Counters
  Bytes Written=3756
```

From Terminal:

```

rohit@rohit-VirtualBox: ~
17/09/23 15:20:41 INFO reduce.MergeManagerImpl: Merged 8 segments, 20012 bytes to disk to satisfy reduce memory limit
17/09/23 15:20:41 INFO reduce.MergeManagerImpl: Merging 1 files, 20002 bytes from disk
17/09/23 15:20:41 INFO reduce.MergeManagerImpl: Merging 0 segments, 0 bytes from memory into reduce
17/09/23 15:20:41 INFO mapred.Merger: Merging 1 sorted segments
17/09/23 15:20:41 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left of total size: 19992 bytes
17/09/23 15:20:41 INFO mapred.LocalJobRunner: 8 / 8 copied.
17/09/23 15:20:41 INFO Configuration.deprecation: mapred.skip.on is deprecated. Instead, use mapreduce.job.skiprecords
17/09/23 15:20:42 INFO mapred.Task: Task:attempt_local982432570_0001_r_000000_0 is done. And is in the process of committing
17/09/23 15:20:42 INFO mapred.LocalJobRunner: 8 / 8 copied.
17/09/23 15:20:42 INFO mapred.Task: Task attempt_local982432570_0001_r_000000_0 is allowed to commit now
17/09/23 15:20:42 INFO output.FileOutputCommitter: Saved output of task 'attempt_local982432570_0001_r_000000_0' to file:/home/rohit/word_count_res/temporary/0/task_local982432570_0001_r_000000
17/09/23 15:20:42 INFO mapred.LocalJobRunner: reduce > reduce
17/09/23 15:20:42 INFO mapred.Task: Task 'attempt_local982432570_0001_r_000000_0' done.
17/09/23 15:20:42 INFO mapred.LocalJobRunner: Finishing task: attempt_local982432570_0001_r_000000_0
17/09/23 15:20:42 INFO mapred.LocalJobRunner: reduce task executor complete.
17/09/23 15:20:42 INFO mapreduce.Job: map 100% reduce 100%
17/09/23 15:20:42 INFO mapreduce.Job: Job job_local982432570_0001 completed successfully
17/09/23 15:20:42 INFO mapreduce.Job: Counters: 30

File System Counters
  FILE: Number of bytes read=2946821
  FILE: Number of bytes written=5454025
  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=745
  Map output records=2754
  Map output bytes=34804
  Map output materialized bytes=20044
  Input split bytes=821
  Combine input records=2754
  Combine output records=1160
  Reduce input groups=588
  Reduce shuffle bytes=20044
  Reduce input records=1160
  Reduce output records=588
  Spilled Records=2320
  Shuffled Maps =8
  Failed Shuffles=0
  Merged Map outputs=8
  GC time elapsed (ms)=412
  Total committed heap usage (bytes)=1398206464

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=26036

File Output Format Counters
  Bytes Written=10097

rohit@rohit-VirtualBox:~$

```

Word Count AWS Execution

The screenshot displays the AWS EMR console interface in a Mozilla Firefox browser. The browser's address bar shows the URL: `https://console.aws.amazon.com/elasticmapreduce/home?region=us-east-1#cluster-details:j-S2AYQ3IL127N`. The console header includes the AWS logo, navigation tabs for 'Services' and 'Resource Groups', and a user profile for 'rohitpatnaik' in the 'N. Virginia' region.

The left-hand navigation pane lists various AWS EMR resources: Amazon EMR, Clusters, Security configurations, VPC subnets, Events, and Help. The 'Clusters' section is currently selected.

The main content area shows the details for a cluster named 'My cluster', which is in a 'Terminated' state with 'Steps completed'. At the top of this section are buttons for 'Clone', 'Terminate', and 'AWS CLI export'. Below these are tabs for 'Summary', 'Monitoring', 'Hardware', 'Events', 'Steps', 'Configurations', and 'Bootstrap actions'. The 'Summary' tab is active.

The 'Summary' tab displays the following information:

- Connections:** --
- Master public DNS:** `ec2-54-88-250-160.compute-1.amazonaws.com` with an [SSH](#) link.
- Tags:** --

The details are organized into three columns:

Summary	Configuration details	Network and hardware
ID: j-S2AYQ3IL127N	Release label: emr-5.8.0	Availability zone: us-east-1c
Creation date: 2017-09-22 15:28 (UTC-4)	Hadoop distribution: Amazon 2.7.3	Subnet ID: subnet-85e0d1df
End date: 2017-09-22 15:39 (UTC-4)	Applications: --	Master: Terminated 1 m3.xlarge
Elapsed time: 11 minutes	Log URI: <code>s3://rohitbucket/log/</code>	Core: Terminated 2 m3.xlarge
Auto-terminate: Yes	EMRFS consistent view: Disabled	Task: --
Termination protection: Off	Custom AMI ID: --	

Below the configuration details, the 'Security and access' section provides the following information:

- Key name:** --
- EC2 instance profile:** EMR_EC2_DefaultRole
- EMR role:** EMR_DefaultRole
- Visible to all users:** [All](#) [Change](#)
- Security groups for Master:** sg-a8a430db (ElasticMapReduce-Master: master)
- Security groups for Core & Task:** sg-99aa36ea (ElasticMapReduce-Core & Task: slave)

The footer of the console includes a 'Feedback' link, the language 'English (US)', and a copyright notice: '© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use'.

HOMEWORK 1

CS6240 Parallel Data Processing – Map Reduce

Rohit Patnaik

The screenshot shows the AWS S3 console interface. The breadcrumb navigation indicates the path: Amazon S3 > rohitpatnaik / output. The 'Overview' tab is active. A search bar is present with the placeholder text 'Type a prefix and press Enter to search. Press ESC to clear.' Below the search bar are buttons for 'Upload', 'Create folder', and 'More'. The region is set to 'US East (N. Virginia)'. A table lists the contents of the bucket, showing columns for Name, Last modified, Size, and Storage class. The file 'part-r-00004' is highlighted. The taskbar at the bottom shows various application icons and the system clock indicating 5:21 PM on 9/23/2017.

Name	Last modified	Size	Storage class
_SUCCESS	Sep 22, 2017 3:36:48 PM	0 B	Standard
part-r-00000	Sep 22, 2017 3:36:41 PM	10.0 KB	Standard
part-r-00001	Sep 22, 2017 3:36:41 PM	10.3 KB	Standard
part-r-00002	Sep 22, 2017 3:36:42 PM	10.5 KB	Standard
part-r-00003	Sep 22, 2017 3:36:42 PM	10.4 KB	Standard
part-r-00004	Sep 22, 2017 3:36:42 PM	10.0 KB	Standard
part-r-00005	Sep 22, 2017 3:36:45 PM	9.9 KB	Standard
part-r-00006	Sep 22, 2017 3:36:48 PM	9.9 KB	Standard