CS. 6240-02 HW1 Vikas Janardhanan

Weather Data Results

For each of the versions of your sequential and multithreaded program detailed in B and C, report the minimum, average, and maximum running time observed over the 10 runs. (5 points)

SEQ

PART B:
Average Running Time:2374.9 ms
Minimum Running Time:2196.0 ms
Maximum Running Time:3158.0 ms
PART C:
Average Running Time:13667.0 ms
Minimum Running Time:13067.0 ms
Maximum Running Time:14594.0 ms

NO-LOCK

PART B:
Average Running Time:1316.8 ms
Minimum Running Time:1071.0 ms
Maximum Running Time:2591.0 ms
PART C:
Average Running Time:5244.0 ms
Minimum Running Time:4944.0 ms
Maximum Running Time:5805.0 ms

COARSE-LOCK:

PART B:
Average Running Time:1312.9 ms
Minimum Running Time:1183.0 ms
Maximum Running Time:1778.0 ms
PART C:
Average Running Time:11883.8 ms
Minimum Running Time:10853.0 ms
Maximum Running Time:14218.0 ms

CS. 6240-02 HW1 Vikas Janardhanan

FINE-LOCK
PART B:
Average Running Time:1295.8 ms
Minimum Running Time:1146.0 ms
Maximum Running Time:1878.0 ms
PART C:
Average Running Time:5647.0 ms
Minimum Running Time:5291.0 ms
Maximum Running Time:6206.0 ms

NO-SHARING
PART B:
Average Running Time:1484.0 ms
Minimum Running Time:1079.4 ms
Maximum Running Time:2528.0 ms
PART C:
Average Running Time: 5653.0 ms
Minimum Running Time: 3823.8 ms
Maximum Running Time:7475.0 ms

Report the number of worker threads used and the speedup of the multithreaded versions based on the corresponding average running times. (5 points)
Number of Worker Threads = 4

SPEED UP:
NO-LOCK
PART B: 1.8
PART C: 2.6

COARSE-LOCK
PART B: 1.8
PART C: 1.15

FINE-LOCK
PART B: 1.83
PART C: 2.4

CS. 6240-02 HW1 Vikas Janardhanan

NO-SHARING
PART B: 1.6
PART C: 2.41

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 is normally expected to finish fastest.
   - When more than one thread works on the data, the resulting parallelism helps to speed up the computation
   - Since there is no lock in this case, there is no bottleneck of waiting for a resource
   - Since there is no synchronization mechanism to take care of the concurrent access of the data structure, the results produced is expected to contain errors.
   - The experiment results are in line with expectation, we have observed the least running time of 1071ms in NO-LOCK

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.
   SEQ is expected to finish slower as compared to others.
   Since data is read in a sequential manner its expected to finish slower. The experiment does confirm our expectation. However, for small inputs SEQ might be faster because of the extra time spent by OS in context switching where threads are involved. The advantage of threads normally is significant when the input size is large such as the present case.

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 output returned by NO-LOCK version is erroneous because of the below reasons:

Since there is no synchronization mechanism used here the updated results in the data structure is erroneous.ie, suppose updating happens from one or more threads at the same time for the same key, then update from one of the thread is lost in the process.
NO-LOCK version was also crashing with nullpointerException because of the inconsistent nature of the data structure due to lack of synchronization.

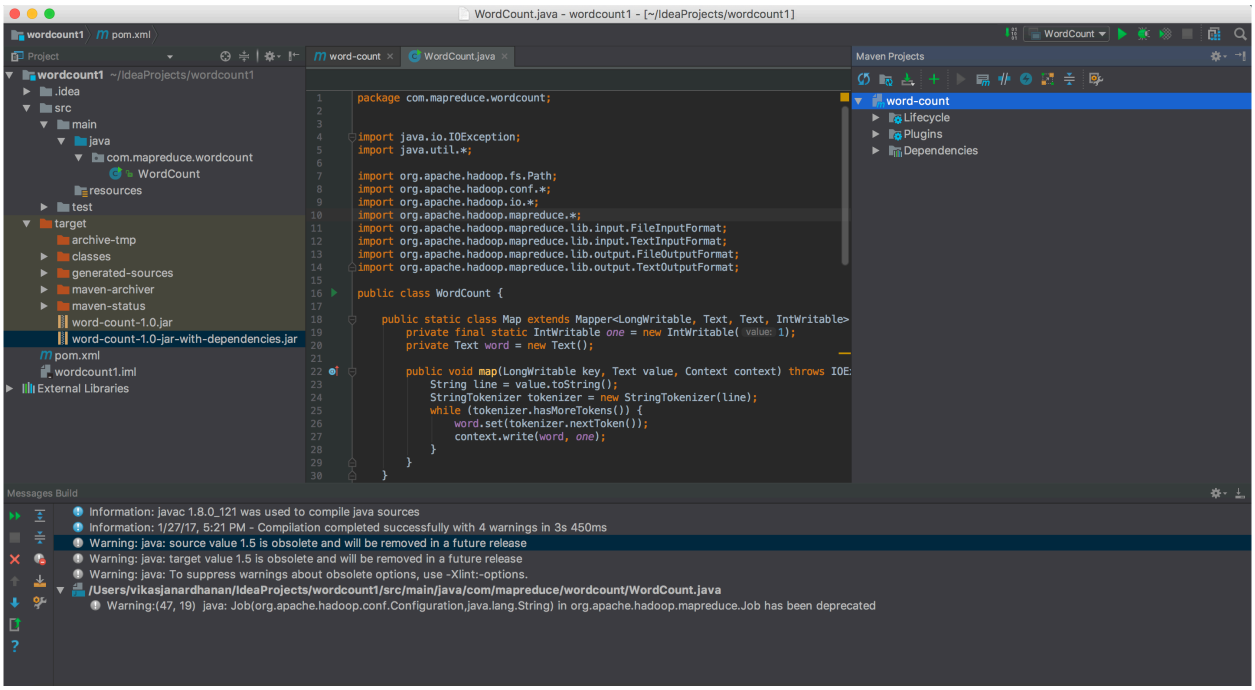
4. Compare the running times of SEQ and COARSE-LOCK. Try to explain why one is slower than the other. (Make sure to consider the results of both B and C—this might support or refute a possible hypothesis.)
   SEQ is slower than COARSE-LOCK:
   - In SEQ the processing of the input happens sequentially whereas in COARSE-LOCK multiple threads works in parallel on the data, this speeds up the processing
   - Only bottleneck in COARSE-LOCK is the synchronized access to the data structure. However, due to parallel nature its faster than SEQ
   - However, when delay is introduced using Fibonacci, the time taken is almost same, because at a time only one thread can have access to data structure and until the delay time is over other threads wait on acquiring the lock. This makes the thread execution more like sequential.

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.
   - In COARSE-LOCK, Fibonacci function call is in the synchronized block and it cannot run parallel on two threads as the lock is acquired on the whole data structure.
   - However, in FINE-LOCK, the same function can run in parallel as there is a chance that not all threads would be updating for the same key in the hashmap.
   - Because of the above reasons, FINE-LOCK performance is better.

CS. 6240-02 HW1 Vikas Janardhanan

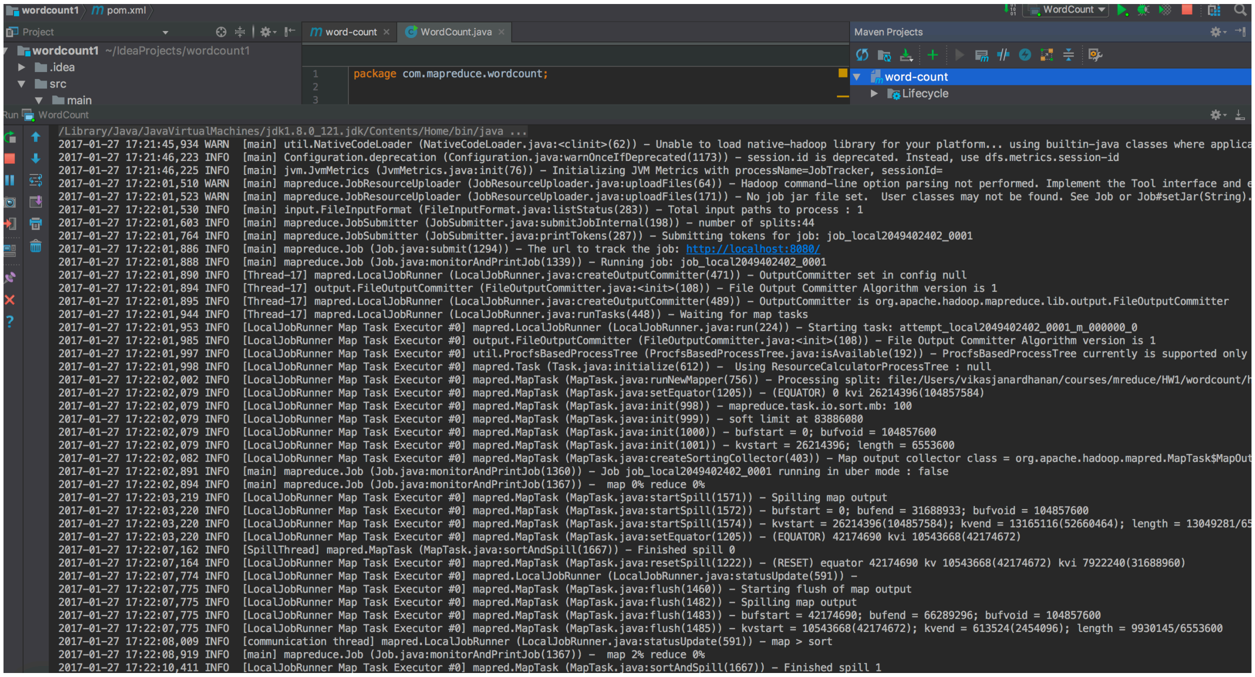
Word Count Local Execution

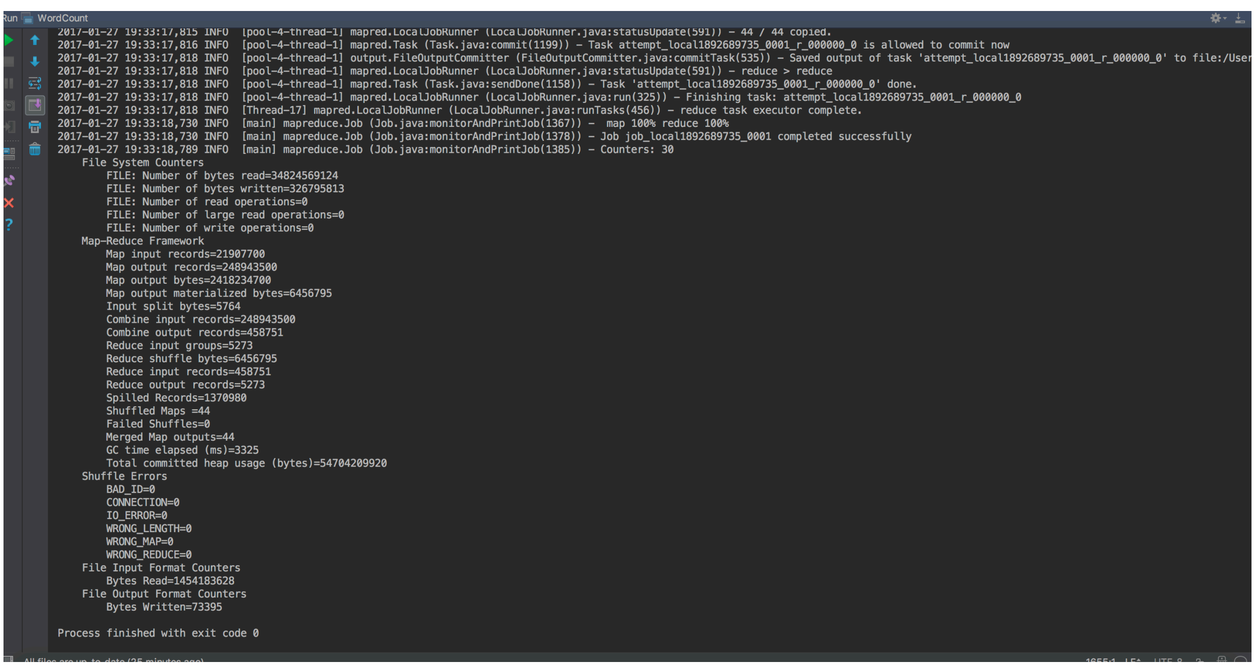
Project directory structure, showing that the WordCount.java file is somewhere in the src directory. (10 points)



CS. 6240-02 HW1 Vikas Janardhanan

The console output for a successful run of the WordCount program inside the IDE. The console output refers to the job summary information Hadoop produces, not the output your job emits. Show at least the last 20 lines of the console output. (10 points)





CS. 6240-02 HW1 Vikas Janardhanan

Word Count AWS Execution

Show a similar screenshot that provides convincing evidence of a successful run of the Word Count program on AWS. Make sure you run the program using at least three machines, i.e., one master node and two workers. (10 points)

	Name	ID	Status	Creation time	Elapsed time	Normalized instance hours
	My cluster	j-3N9OEKNF9D83K	Terminated User request	2017-01-27 20:09 (UTC-5)	19 minutes	12

Summary

Master public DNS: ec2-54-162-172-14.compute-1.amazonaws.com
Termination protection: Off
Tags: --

Hardware

Master: Terminated 1 m4.large
Core: Terminated 2 m4.large
Task: --

View cluster details View monitoring details

Steps View all interactive jobs

Name	Status	Start time (UTC-5)	Elapsed time
Custom JAR	Completed	2017-01-27 20:16 (UTC-5)	6 minutes
Setup hadoop debugging	Completed	2017-01-27 20:16 (UTC-5)	2 seconds

Bootstrap Actions

Name

No bootstrap actions availabl

Steps

Add step Clone step Cancel step

Steps View all interactive jobs | View all jobs

Filter: All steps Filter steps ... 2 steps (all loaded)

ID	Name	Status	Start time (UTC-5)	Elapsed time	Log files
s-1F27UY4FJWPCQ	Custom JAR	Completed	2017-01-27 20:16 (UTC-5)	6 minutes	View logs
s-13PQGKMZAQ39T	Setup hadoop debugging	Completed	2017-01-27 20:16 (UTC-5)	2 seconds	View logs

Hardware

Add task instance group

Instance Groups

Filter: Filter instance groups ... 2 instance groups (all loaded)

ID	Node type & name	Status	Instance Type	Instance Count	EBS Volumes per Instance
ig-3GVSE115NYDXW	CORE Core - 2	Terminated (2 Requested)	m4.large	0	1
ig-2WLZBG6F1DG47	MASTER Master - 1	Terminated (1 Requested)	m4.large	0	1

CS. 6240-02 HW1 Vikas Janardhanan

Controller log:

```
  AWS_DEFAULT_REGION=us-east-1
  AWS_ELB_HOME=/opt/aws/apitools/elb
  LESS_TERMCAP_us=[04;38;5;111m
  EC2_HOME=/opt/aws/apitools/ec2
  TERM=linux
  XFILESEARCHPATH=/usr/dt/app-defaults/%L/Dt
  runlevel=3
  LANG=en_US.UTF-8
  AWS_CLOUDWATCH_HOME=/opt/aws/apitools/mon
  MAIL=/var/spool/mail/hadoop
  LESS_TERMCAP_ue=[0m
  LOGNAME=hadoop
  PWD=/
  LANGSH_SOURCED=1
  HADOOP_CLIENT_OPTS=-Djava.io.tmpdir=/mnt/var/lib/hadoop/steps/s-1F27UY4FJWPCQ/tmp
  _=/etc/alternatives/jre/bin/java
  CONSOLETYPE=serial
  RUNLEVEL=3
  LESSOPEN=||/usr/bin/lesspipe.sh %s
  previous=N
  UPSTART_EVENTS=runlevel
  AWS_PATH=/opt/aws
  USER=hadoop
  UPSTART_INSTANCE=
  PREVLEVEL=N
  HADOOP_LOGFILE=syslog
  HOSTNAME=ip-172-31-0-196
  NLSPATH=/usr/dt/lib/nls/msg/%L/%N.cat
  HADOOP_LOG_DIR=/mnt/var/log/hadoop/steps/s-1F27UY4FJWPCQ
  EC2_AMITOOL_HOME=/opt/aws/amitools/ec2
  SHLVL=5
  HOME=/home/hadoop
  HADOOP_IDENT_STRING=hadoop
INFO redirectOutput to /mnt/var/log/hadoop/steps/s-1F27UY4FJWPCQ/stdout
INFO redirectError to /mnt/var/log/hadoop/steps/s-1F27UY4FJWPCQ/stderr
INFO Working dir /mnt/var/lib/hadoop/steps/s-1F27UY4FJWPCQ
INFO ProcessRunner started child process 8142 :
hadoop    8142  3311  0 01:16 ?        00:00:00 bash /usr/lib/hadoop/bin/hadoop jar /mnt/var/lib/hadoop/steps/s-1F27UY4FJWPCQ/wordcount-1.0.jar s3://neu.mr.hw1/input
s3://neu.mr.hw1/output
2017-01-28T01:16:57.138Z INFO HadoopJarStepRunner.Runner: startRun() called for s-1F27UY4FJWPCQ Child Pid: 8142
INFO Synchronously wait child process to complete : hadoop jar /mnt/var/lib/hadoop/steps/s-1F27UY4F...
INFO waitProcessCompletion ended with exit code 0 : hadoop jar /mnt/var/lib/hadoop/steps/s-1F27UY4F...
INFO total process run time: 390 seconds
2017-01-28T01:23:25.270Z INFO Step created jobs: job_1485566076966_0001
2017-01-28T01:23:25.271Z INFO Step succeeded with exitCode 0 and took 390 seconds
```

Syslog

Secure https://s3-us-west-2.amazonaws.com/neu.mr.hw1/logs/j-3N9OEKNF9D83K/steps/s-1F27UY4FJWPCQ/syslog.gz?X-Amz-Date=20170128T021823Z&X-Amz-Expires=30...

```
2017-01-28 01:17:05,276 INFO org.apache.hadoop.yarn.client.RMProxy (main): Connecting to ResourceManager at ip-172-31-0-196.ec2.internal/172.31.0.196:8032
2017-01-28 01:17:08,047 WARN org.apache.hadoop.mapreduce.JobResourceUploader (main): Hadoop command-line option parsing not performed. Implement the Tool interface and execut
your application with ToolRunner to remedy this.
2017-01-28 01:17:09,033 INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat (main): Total input paths to process : 1
2017-01-28 01:17:09,045 INFO com.hadoop.compression.lzo.GPLNativeCodeLoader (main): Loaded native gpl library
2017-01-28 01:17:09,048 INFO com.hadoop.compression.lzo.LzoCodec (main): Successfully loaded & initialized native-lzo library [hadoop-lzo rev
f644c4f7eb0b342b189783939524370d26bf554f]
2017-01-28 01:17:09,241 INFO org.apache.hadoop.mapreduce.JobSubmitter (main): number of splits:22
2017-01-28 01:17:09,392 INFO org.apache.hadoop.mapreduce.JobSubmitter (main): Submitting tokens for job: job_1485566076966_0001
2017-01-28 01:17:09,963 INFO org.apache.hadoop.yarn.client.api.impl.YarnClientImpl (main): Submitted application application_1485566076966_0001
2017-01-28 01:17:10,044 INFO org.apache.hadoop.mapreduce.Job (main): The url to track the job: http://ip-172-31-0-196.ec2.internal:20888/proxy/application_1485566076966_0001/
2017-01-28 01:17:10,045 INFO org.apache.hadoop.mapreduce.Job (main): Running job: job_1485566076966_0001
2017-01-28 01:17:22,294 INFO org.apache.hadoop.mapreduce.Job (main): Job job_1485566076966_0001 running in uber mode : false
2017-01-28 01:17:22,295 INFO org.apache.hadoop.mapreduce.Job (main):  map 0% reduce 0%
2017-01-28 01:17:42,431 INFO org.apache.hadoop.mapreduce.Job (main):  map 1% reduce 0%
2017-01-28 01:17:43,438 INFO org.apache.hadoop.mapreduce.Job (main):  map 3% reduce 0%
2017-01-28 01:17:46,457 INFO org.apache.hadoop.mapreduce.Job (main):  map 4% reduce 0%
2017-01-28 01:17:52,490 INFO org.apache.hadoop.mapreduce.Job (main):  map 5% reduce 0%
2017-01-28 01:17:55,506 INFO org.apache.hadoop.mapreduce.Job (main):  map 6% reduce 0%
2017-01-28 01:17:56,510 INFO org.apache.hadoop.mapreduce.Job (main):  map 7% reduce 0%
2017-01-28 01:17:58,519 INFO org.apache.hadoop.mapreduce.Job (main):  map 8% reduce 0%
2017-01-28 01:17:59,525 INFO org.apache.hadoop.mapreduce.Job (main):  map 9% reduce 0%
2017-01-28 01:18:00,531 INFO org.apache.hadoop.mapreduce.Job (main):  map 10% reduce 0%
2017-01-28 01:18:01,535 INFO org.apache.hadoop.mapreduce.Job (main):  map 11% reduce 0%
2017-01-28 01:18:02,539 INFO org.apache.hadoop.mapreduce.Job (main):  map 12% reduce 0%
2017-01-28 01:18:04,549 INFO org.apache.hadoop.mapreduce.Job (main):  map 13% reduce 0%
2017-01-28 01:18:05,553 INFO org.apache.hadoop.mapreduce.Job (main):  map 14% reduce 0%
2017-01-28 01:18:08,575 INFO org.apache.hadoop.mapreduce.Job (main):  map 16% reduce 0%
2017-01-28 01:18:10,585 INFO org.apache.hadoop.mapreduce.Job (main):  map 17% reduce 0%
2017-01-28 01:18:26,660 INFO org.apache.hadoop.mapreduce.Job (main):  map 19% reduce 0%
2017-01-28 01:18:29,671 INFO org.apache.hadoop.mapreduce.Job (main):  map 21% reduce 0%
2017-01-28 01:18:34,698 INFO org.apache.hadoop.mapreduce.Job (main):  map 23% reduce 0%
2017-01-28 01:18:35,703 INFO org.apache.hadoop.mapreduce.Job (main):  map 25% reduce 0%
2017-01-28 01:18:41,735 INFO org.apache.hadoop.mapreduce.Job (main):  map 26% reduce 0%
2017-01-28 01:18:44,746 INFO org.apache.hadoop.mapreduce.Job (main):  map 27% reduce 0%
2017-01-28 01:18:50,778 INFO org.apache.hadoop.mapreduce.Job (main):  map 29% reduce 0%
2017-01-28 01:18:51,782 INFO org.apache.hadoop.mapreduce.Job (main):  map 32% reduce 0%
2017-01-28 01:18:52,788 INFO org.apache.hadoop.mapreduce.Job (main):  map 33% reduce 0%
2017-01-28 01:18:54,796 INFO org.apache.hadoop.mapreduce.Job (main):  map 35% reduce 0%
2017-01-28 01:18:56,810 INFO org.apache.hadoop.mapreduce.Job (main):  map 36% reduce 0%
2017-01-28 01:19:09,864 INFO org.apache.hadoop.mapreduce.Job (main):  map 38% reduce 0%
2017-01-28 01:19:10,867 INFO org.apache.hadoop.mapreduce.Job (main):  map 39% reduce 0%
2017-01-28 01:19:13,888 INFO org.apache.hadoop.mapreduce.Job (main):  map 40% reduce 0%
2017-01-28 01:19:18,906 INFO org.apache.hadoop.mapreduce.Job (main):  map 42% reduce 0%
2017-01-28 01:19:22,920 INFO org.apache.hadoop.mapreduce.Job (main):  map 44% reduce 0%
2017-01-28 01:19:23,924 INFO org.apache.hadoop.mapreduce.Job (main):  map 45% reduce 0%
```