

Assignment 1

(Amardeep Singh)

All timing values are in milliseconds.

Q) 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.

Answer: -

[..] – Denotes the 10 run times sorted in ascending order.

Scenario B

Running Times

1. SEQ Running times :-

*Minimum time is =2959
Maximum time is =3815
Average time is =3140*

2. No Lock Running times:-

*Minimum time is =1501
Maximum time is =2225
Average time is =1613*

3. Coarse-Lock Running times:-

*Minimum time is =1611
Maximum time is =2379
Average time is =1750*

4.Fine-Lock

Minimum time is =1514

Maximum time is =2270

Average time is =1618

5.No Share Running times:-

Minimum time is =1597

Maximum time is =2465

Average time is =1701

Scenario C

1. SEQ Running times: -

Minimum time is =3185

Maximum time is =3963

Average time is =3326

2. No Lock Running times:-

Minimum time is =1545

Maximum time is =2254

Average time is =1665

3.Coarse-Lock Running times:-

Minimum time is =1739

Maximum time is =2401

Average time is =1860

4.Fine-Lock

Minimum time is =1617
Maximum time is =2466
Average time is =1696

5.No Share

*Minimum time is =1621
Maximum time is =2620
Average time is =1798*

Question-Report the number of worker threads used and the speedup of themultithreaded versions based on the corresponding average running times. (5 points)

Answer- The number of worker threads used – **4** (depends on the cores of the CPU, generated using

Runtime.getRuntime().availableProcessors())

Speed Up for Multithreaded Versions: (Time for Sequential / Time for Parallel)

Scenario B

2.No Lock Running -

$$\begin{aligned} \text{Speedup} &= 3140 / 1613 \\ &= 1.95 \end{aligned}$$

3.Coarse-Lock -

$$\begin{aligned} \text{Speedup} &= 3140 / 1750 \\ &= 1.79 \end{aligned}$$

4.Fine-Lock -

$$\begin{aligned} \text{Speedup} &= 3140 / 1618 \\ &= 1.94 \end{aligned}$$

5.No Share -

$$\begin{aligned} \text{Speedup} &= 3140 / 1701 \\ &= 1.84 \end{aligned}$$

Scenario C

2.No Lock -

$$\begin{aligned} \text{Speedup} &= 3326 / 1665 \\ &= 1.99 \end{aligned}$$

3.Coarse-Lock -

$$\begin{aligned} \text{Speedup} &= 3326 / 1860 \\ &= 1.78 \end{aligned}$$

4.Fine-Lock -

$$\begin{aligned} \text{Speedup} &= 3326 / 1696 \\ &= 1.96 \end{aligned}$$

5.No Share -

$$\begin{aligned} \text{Speedup} &= 3326 / 1795 \\ &= 1.85 \end{aligned}$$

Question 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 expect the **NO-LOCK** version to be the fastest among all because

- It uses multiple threads to do the job in parallel unlike the SEQ code, hence its faster than SEQ.
- Coarse-lock and Fine-lock use the same multiple threads to get the job done but have an overhead of synchronization and locks because all the threads are modifying a shared data structure. Since NO-LOCK uses no such locks or synchronization there is not overhead making it faster than Coarse-lock and fine-lock.
- In No-sharing all the threads work on their own separate data structures which are then later combined into one to find the average per station. This overhead caused by the combining step (which is not needed in NO-LOCK) causes this to be slower than NO-LOCK.
- ✓ There is no waiting for any thread in NO-LOCK for either any lock or shared resource. Hence making it fastest among all the approaches above.
- ✓ My experiments confirm my expectations.

Question 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 the SEQ version to finish the slowest because in this sequential process entire work is done by a single processor while the other tasks are waiting for the processor to be free and only then one of them can make use of the processor. This leads to high waiting times. Whereas in other versions

(multithreaded) of the program, tasks are performed in parallel by multiple threads, hence making them faster.

- ✓ My experiment confirms my expectation.

Question 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.

Answer: - On comparison of the temperature averages it was found that the temperature averages calculated with NO-LOCK version were wrong and also inconsistent.

Question 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: - Coarse-lock is faster than the SEQ version because in SEQ version all processes have to wait till the processor is done with the task it is executing. Whereas in the Coarse-Lock version the contention is only between the shared resource which is locked and only one thread has access to this at a time. Apart from this threads are free to perform all other tasks (which don't need access to shared resource) in parallel. This is not the case in SEQ version hence leading to more wait times

In Scenario C adding Fibonacci lead to more computation and hence increasing the time taken by both the versions. But still SEQ version remains slower than the Coarse-Lock version.

Question 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 :- On adding extra computation in Scenario C the running times for both the versions increase. The average time in Coarse lock version increases by ~110ms(between scenario B and c) whereas in case of fine lock the increase is ~75ms.

The possible reason will be that in coarse lock we lock the entire resource irrespective of what part of the resource is being accessed. There is a high possibility of parts of resource being unavailable for process despite of the fact that no other processes are working on them. Eg: Locking a HashMap to update any key value in case of coarse lock will lock the Hashmap for all reads and writes happening at other keys. If we add a processing task in this lock it will increase the wait time for other processes, which might need other parts of the same resource.

FINE-LOCK just locks the part of resource, which is being contended for by two processes hence eliminating unnecessary wait times. If we add processing in the block, which locks the parts of the resource, it affects only the processes that are contending for the resource i.e. all reads and writes on the same key. But the other parts of the resources are still available to the processes running, so the processing delays will be reduced.

Week 2

Question1) Word Count Local Execution Show a screenshot of your IDE window. It should contain the following information:

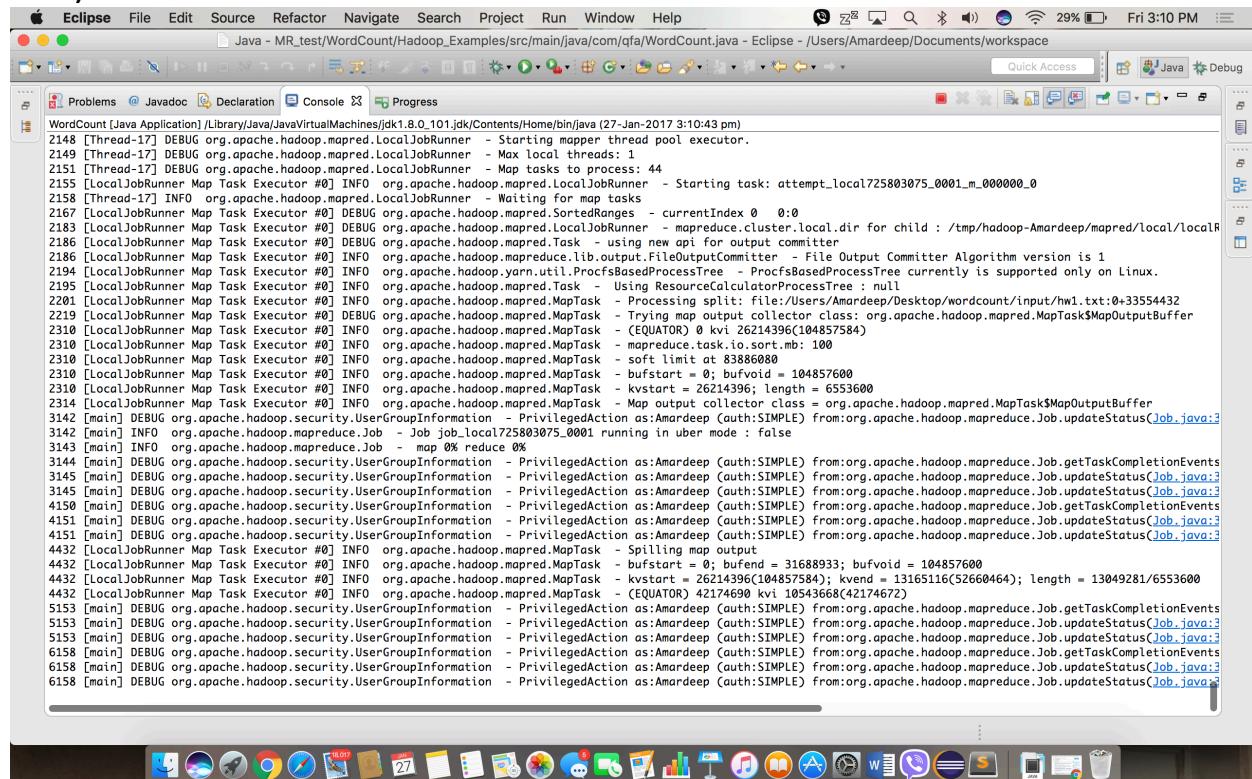
- Project directory structure, showing that the WordCount.java file is somewhere in the src directory.

The screenshot shows the Eclipse IDE interface on a Mac OS X system. The title bar indicates the project is 'WC_AMAR' and the file is 'WordCount.java'. The code editor displays the Java code for a WordCount program, which includes imports for java.io.IOException, java.util.StringTokenizer, and org.apache.hadoop.mapreduce.Mapper. The Mapper class has overridden map and reduce methods. The map method tokenizes input lines and emits word counts. The reduce method sums up the counts for each word. The main method is also shown. The Package Explorer view on the left shows the project structure with 'src/main/java' containing 'WordCount.java' and 'src/main/resources' containing 'log4j.properties'. The Maven Dependencies view shows various Hadoop and Guice dependencies. The bottom dock contains icons for various Mac OS X applications like Finder, Mail, and Safari.

```
2* Licensed to the Apache Software Foundation (ASF) under one
18
19+ import java.io.IOException;
20
21
22 public class WordCount {
23
24     public static class TokenizerMapper
25         extends Mapper<Object, Text, Text, IntWritable>{
26
27         private final static IntWritable one = new IntWritable(1);
28
29         private Text word = new Text();
30
31         @Override
32         public void map(Object key, Text value, Context context
33                         ) throws IOException, InterruptedException {
34             StringTokenizer itr = new StringTokenizer(value.toString());
35             while (itr.hasMoreTokens()) {
36                 word.set(itr.nextToken());
37                 context.write(word, one);
38             }
39         }
40     }
41
42     public static class IntSumReducer
43         extends Reducer<Text,IntWritable,Text,IntWritable> {
44         private IntWritable result = new IntWritable();
45
46         @Override
47         public void reduce(Text key, Iterable<IntWritable> values,
48                           Context context
49                           ) throws IOException, InterruptedException {
50             int sum = 0;
51             for (IntWritable val : values) {
52                 sum += val.get();
53             }
54             result.set(sum);
55             context.write(key, result);
56         }
57     }
58
59     public static void main(String[] args) throws Exception {
60
61
62
63
64
65
66
67
68
69 }
```

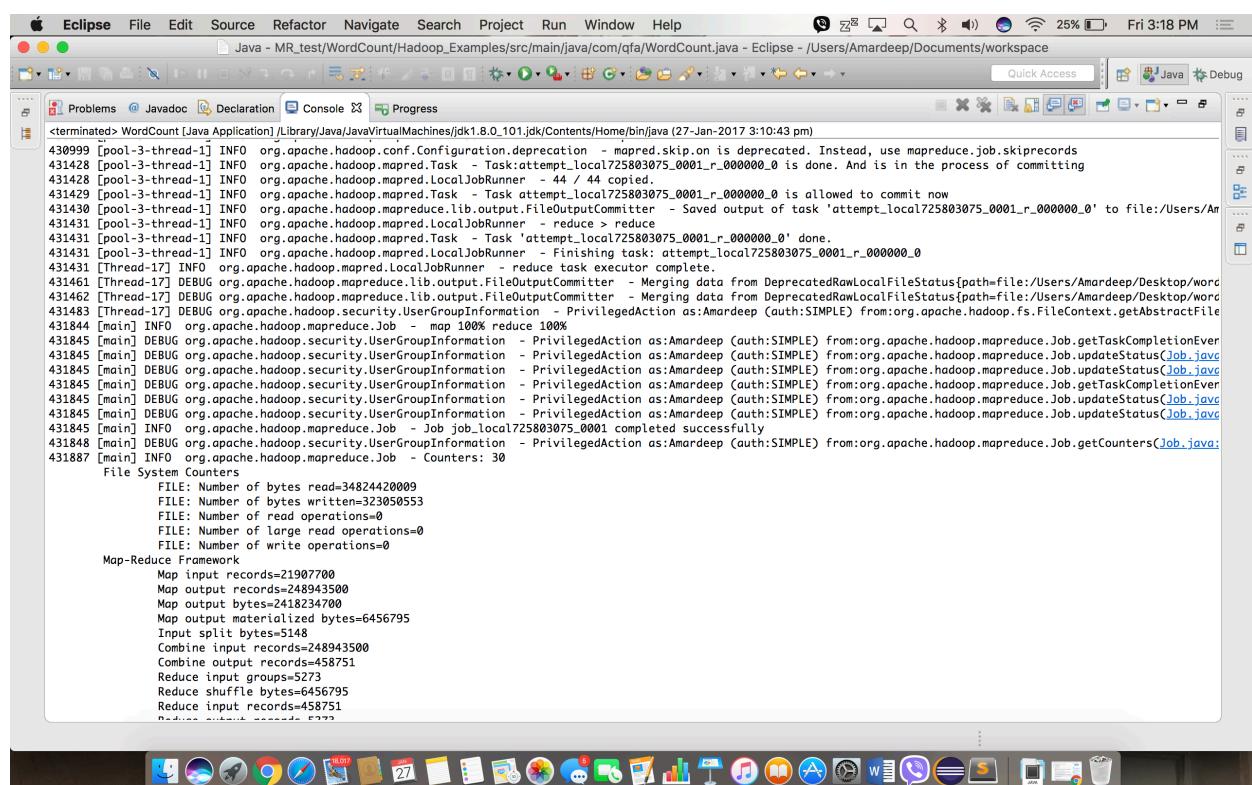
Qustion2) The console output for a successful run of the WordCount program inside the IDE. The console output refers to the job summary information Hadoop produces, not the output your job emits. Show at least the last 20 lines of the console output.

Ans)



```
Java - MR_test/WordCount/Hadoop_Examples/src/main/java/com/qfa/WordCount.java - Eclipse - /Users/Amardeep/Documents/workspace
```

```
2148 [Thread-17] DEBUG org.apache.hadoop.mapred.LocalJobRunner - Starting mapper thread pool executor.
2149 [Thread-17] DEBUG org.apache.hadoop.mapred.LocalJobRunner - Max local threads: 1
2151 [Thread-17] DEBUG org.apache.hadoop.mapred.LocalJobRunner - Map tasks to process: 44
2155 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapred.LocalJobRunner - Starting task: attempt_local725803075_0001_m_000000_0
2156 [Thread-17] INFO org.apache.hadoop.mapred.LocalJobRunner - Waiting for map tasks
2167 [LocalJobRunner Map Task Executor #0] DEBUG org.apache.hadoop.mapred.SortedRanges - currentIndex 0 :0
2183 [LocalJobRunner Map Task Executor #0] DEBUG org.apache.hadoop.mapred.LocalJobRunner - mapreduce.cluster.local.dir for child : /tmp/hadoop-Amardeep/mapred/local/localR
2186 [LocalJobRunner Map Task Executor #0] DEBUG org.apache.hadoop.mapred.Task - using new api for output committer
2186 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter - File Output Committer Algorithm version is 1
2194 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.yarn.util.ProcfsBasedProcessTree - ProcfsBasedProcessTree currently is supported only on Linux.
2195 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapred.Task - Using ResourceCalculatorProcessTree : null
2201 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapred.MapTask - Processing split: file:/Users/Amardeep/Desktop/wordcount/input/hw1.txt:0+33554432
2219 [LocalJobRunner Map Task Executor #0] DEBUG org.apache.hadoop.mapred.MapTask - Trying output collector class: org.apache.hadoop.mapred.MapTask$MapOutputBuffer
2310 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapred.MapTask - (EQUATOR) 0 kv 26214396(10485784)
2310 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapred.MapTask - mapreduce.task.io.sort.mb: 100
2310 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapred.MapTask - soft limit at 83886080
2310 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapred.MapTask - bufstart = 0; bufvoid = 104857600
2310 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapred.MapTask - kvstart = 26214396; length = 6553600
2314 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapred.MapTask - Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
3142 [main] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as: Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.updateStatus(Job.java:3
3142 [main] INFO org.apache.hadoop.mapreduce.Job - Job job_local725803075_0001 running in uber mode : false
3143 [main] INFO org.apache.hadoop.mapreduce.Job - map 0% reduce 0%
3144 [main] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as: Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.getTaskCompletionEvents
3145 [main] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as: Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.updateStatus(Job.java:3
3145 [main] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as: Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.updateStatus(Job.java:3
4150 [main] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as: Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.getTaskCompletionEvents
4151 [main] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as: Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.updateStatus(Job.java:3
4151 [main] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as: Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.updateStatus(Job.java:3
4432 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapred.MapTask - Spilling map output
4432 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapred.MapTask - bufstart = 0; bufend = 31688933; bufvoid = 104857600
4432 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapred.MapTask - kvstart = 26214396(104857584); kvend = 13165116(52660464); length = 13049281/6553600
4432 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapred.MapTask - (EQUATOR) 42174690(1054366842174672)
5151 [main] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as: Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.getTaskCompletionEvents
5153 [main] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as: Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.updateStatus(Job.java:3
5153 [main] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as: Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.updateStatus(Job.java:3
6158 [main] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as: Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.getTaskCompletionEvents
6158 [main] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as: Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.updateStatus(Job.java:3
6158 [main] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as: Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.updateStatus(Job.java:3
```



```
Java - MR_test/WordCount/Hadoop_Examples/src/main/java/com/qfa/WordCount.java - Eclipse - /Users/Amardeep/Documents/workspace
```

```
<terminated> WordCount [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home/bin/java (27-Jan-2017 3:10:43 pm)

430999 [pool-3-thread-1] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.skip.on is deprecated. Instead, use mapreduce.job.skiprecords
431428 [pool-3-thread-1] INFO org.apache.hadoop.mapred.Task - Task attempt_local725803075_0001_r_000000_0 is done. And is in the process of committing
431428 [pool-3-thread-1] INFO org.apache.hadoop.mapred.LocalJobRunner - 44 / 44 copied.
431429 [pool-3-thread-1] INFO org.apache.hadoop.mapred.Task - Task attempt_local725803075_0001_r_000000_0 is allowed to commit now
431430 [pool-3-thread-1] INFO org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter - Saved output of task 'attempt_local725803075_0001_r_000000_0' to file:/Users/Amardeep/Desktop/wordcount/output/part-r-000000
431431 [pool-3-thread-1] INFO org.apache.hadoop.mapred.LocalJobRunner - reduce > reduce
431431 [pool-3-thread-1] INFO org.apache.hadoop.mapred.Task - Task attempt_local725803075_0001_r_000000_0 done.
431431 [pool-3-thread-1] INFO org.apache.hadoop.mapred.LocalJobRunner - Finishing task: attempt_local725803075_0001_r_000000_0
431431 [Thread-17] INFO org.apache.hadoop.mapred.LocalJobRunner - reduce task execute complete.
431461 [Thread-17] DEBUG org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter - Merging data from DeprecatedRawLocalFileStatus[path=file:/Users/Amardeep/Desktop/wordcount/output/part-r-000000] to file:/Users/Amardeep/Desktop/wordcount/output/part-r-000000
431462 [Thread-17] DEBUG org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter - Merging data from DeprecatedRawLocalFileStatus[path=file:/Users/Amardeep/Desktop/wordcount/output/part-r-000000] to file:/Users/Amardeep/Desktop/wordcount/output/part-r-000000
431483 [Thread-17] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as: Amardeep (auth:SIMPLE) from:org.apache.hadoop.fs.FileContext.getAbstractFile
431843 [main] INFO org.apache.hadoop.mapreduce.Job - map 100% reduce 100%
431843 [main] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as: Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.getTaskCompletionEvents
431845 [main] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as: Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.updateStatus(Job.java:3
431845 [main] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as: Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.updateStatus(Job.java:3
431845 [main] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as: Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.getTaskCompletionEvents
431845 [main] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as: Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.updateStatus(Job.java:3
431845 [main] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as: Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.getTaskCompletionEvents
431845 [main] DEBUG org.apache.hadoop.mapreduce.Job - Job job_local725803075_0001 completed successfully
431845 [main] INFO org.apache.hadoop.mapreduce.Job - Job job_local725803075_0001 completed successfully
431845 [main] DEBUG org.apache.hadoop.mapreduce.Job - PrivilegedAction as: Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.getCounters(Job.java:3
431887 [main] INFO org.apache.hadoop.mapreduce.Job - Counters: 30
    File System Counters
        FILE: Number of bytes read=34824420099
        FILE: Number of bytes written=32305053
        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=5148
        Combine input records=248943500
        Combine output records=458751
        Reduce input groups=573
        Reduce shuffle bytes=6456795
        Reduce input records=458751
        Reduce output bytes=6456795
        Reduce output records=458751
```

```
Java - MR_test/WordCount/Hadoop_Examples/src/main/java/com/qfa/WordCount.java - Eclipse - /Users/Amardeep/Documents/workspace
431845 [main] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as:Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.updateStatus(Job.java)
431845 [main] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as:Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.updateStatus(Job.java)
431845 [main] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as:Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.getTaskCompletionEvent(Job.java)
431845 [main] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as:Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.updateStatus(Job.java)
431845 [main] INFO org.apache.hadoop.mapreduce.Job - Job job_local1725803075_0001 completed successfully
431845 [main] DEBUG org.apache.hadoop.security.UserGroupInformation - PrivilegedAction as:Amardeep (auth:SIMPLE) from:org.apache.hadoop.mapreduce.Job.updateStatus(Job.java)
431887 [main] INFO org.apache.hadoop.mapreduce.Job - Counters: 30
File System Counters
FILE: Number of bytes read=34824420009
FILE: Number of bytes written=323050553
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=5148
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)=3284
Total committed heap usage (bytes)=52108460032
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
```

Question2) 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 core nodes (those with HDFS) as workers.

Ans)

Amazon EMR

Cluster list

Security configurations

VPC subnets

Help

Cluster: WordCount_Hw1.txt Terminating Steps completed

Connections: --

Master public DNS: ec2-52-201-238-249.compute-1.amazonaws.com SSH

Tags: -- View All / Edit

Summary

ID: j-62AXNSRO22CB
Creation date: 2017-01-27 16:39 (UTC-5)
Elapsed time: 14 minutes
Auto-terminate: Yes
Termination Off protection:

Network and Hardware

Availability zone: us-east-1a
Subnet ID: subnet-8da296c4
Master: Terminating 1 m4.large
Core: Terminating 2 m4.large
Task: --

Configuration Details

Release label: emr-5.3.0
Hadoop distribution: Amazon 2.7.3
Applications: --
Log URL: s3://amardeep-mr/log/

EMRFS consistent view:
Disabled

Security and Access

Key name: --
EC2 instance profile: EMR_EC2_DefaultRole
EMR role: EMR_DefaultRole
Visible to all users: All Change
Security groups for sg-f7f1588b (ElasticMapReduce-Master): master
Security groups for sg-f0f1588c (ElasticMapReduce-Core & Task: slave)

► Monitoring

► Hardware

Feedback English

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Amazon EMR

Cluster list

Security configurations

VPC subnets

Help

Security groups for sg-f0f1588c (ElasticMapReduce-Core & Task: slave)

► Monitoring

▼ 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
ig-JBD0NGRLET3V	MASTER Master Instance Group	Terminating	m4.large	1
ig-3ASW1YAGBOV1E	CORE Core Instance Group	Terminating	m4.large	2

▼ Steps

Add step Clone step Cancel step

Steps

View all interactive jobs | View all jobs

Feedback English

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Chrome File Edit View History Bookmarks People Window Help Fri 5:00 PM

https://console.aws.amazon.com/elasticmapreduce/home?region=us-east-1#cluster-details;j-62AXNSRO22CB

Services Resource Groups

amardeep0070 @ s-207Z7D7IHZND 1 of 1

Amazon EMR

Cluster list Security configurations VPC subnets Help

Steps

Add step Clone step Cancel step

Steps

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

ID	Name	Status	Start time (UTC-5)	Elapsed time
s-3UHD9BD7BC94I	Custom JAR	Completed	2017-01-27 16:47 (UTC-5)	5 minutes
s-1WMW8JQ7DKHRP	Setup hadoop debugging	Completed	2017-01-27 16:47 (UTC-5)	2 seconds

View all interactive jobs | View all jobs

Configurations

View JSON

Configurations

All classifications Filter configurations ... 0 configurations (all loaded)

Classification	Key	Value
----------------	-----	-------

Events

Bootstrap Actions

Feedback English

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Java Debug

The screenshot shows the Amazon EMR console within a Chrome browser window. The URL is https://console.aws.amazon.com/elasticmapreduce/home?region=us-east-1#cluster-details;j-62AXNSRO22CB. The interface includes a top navigation bar with links for File, Edit, View, History, Bookmarks, People, Window, and Help, along with system status icons like battery level (15%), signal strength, and network. Below the navigation is a tab bar with 'Services' and 'Resource Groups'. The main content area is titled 'Amazon EMR' and contains a sidebar with links for 'Cluster list', 'Security configurations', 'VPC subnets', and 'Help'. The main panel displays 'Steps' and 'Configurations'. Under 'Steps', there are two completed steps: 'Custom JAR' (ID: s-3UHD9BD7BC94I) and 'Setup hadoop debugging' (ID: s-1WMW8JQ7DKHRP). Both steps were completed on 2017-01-27 at 16:47 UTC-5, with an elapsed time of 5 minutes for the first and 2 seconds for the second. Under 'Configurations', there are no configurations listed. At the bottom, there are links for 'Feedback' and 'English', and a footer with copyright information and links to 'Privacy Policy' and 'Terms of Use'.