

CS 6240: Assignment 1

Multithreading Concepts and AWS Setup

Analyze Data with Sequential and Concurrent Programs

Weather Data Results

All time units are in **ms** (milliseconds)

NAME	AVERAGE	MINIMUM	MAXIMUM	SPEEDUP
SEQ	9535.60	9370.00	10186.00	1.000
NO-LOCK	3645.70	2575.00	5067.00	2.616
COARSE-LOCK	3920.90	2628.00	5287.00	2.432
FINE-LOCK	3468.00	2648.00	5254.00	2.750
NO-SHARING	3092.60	2783.00	5188.00	3.083

After adding Fibonacci(17) computation to all versions we get following results as below

NAME	AVERAGE	MINIMUM	MAXIMUM	SPEEDUP
SEQ	8187.10	7987.00	9387.00	1.000
NO-LOCK	3196.50	2581.00	5015	2.561
COARSE-LOCK	3976.80	2639.00	5044.00	2.059
FINE-LOCK	4025.80	2568.00	5074.00	2.034
NO-SHARING	4072.80	2674.00	5136.00	2.010

A1. As discussed in class and learning modules, theoretical speedup possible for a task divided into n concurrent tasks is n . So in this case even though NO-LOCK has no control over data access by multiple thread leading to data inconsistencies but this will indeed allow it to finish the task in hand in fastest amount of time. The experiments confirm my expectation in my case.

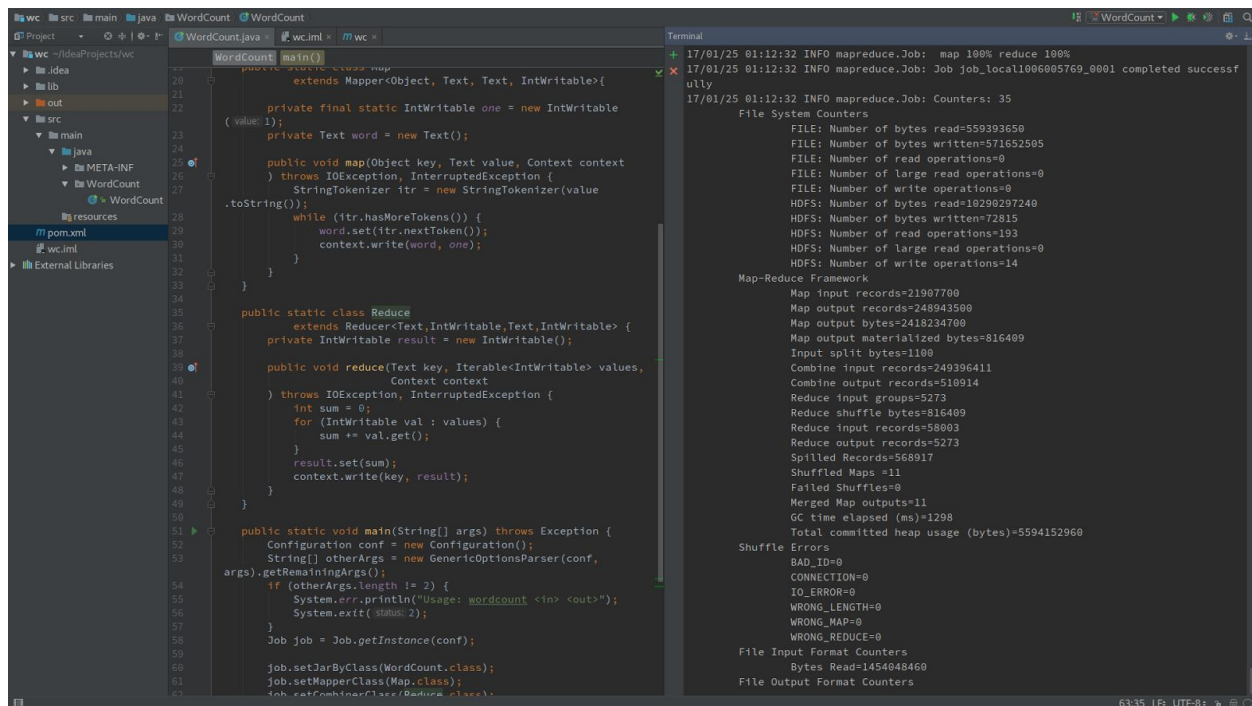
A2. I would expect SEQ to finish slowest since it acquires a lock on the since its occurs in a sequential fashion which is slow with the ones it is put in comparison with who tend to parallelize the task into N smaller units and assign separate units to separate workers for processing. Yes experiments in this case corroborate the same quantitatively performing at least in double the time compared to other algorithms in the play.

A3. Temperature averages are consistent for all programs except NO-LOCK whose averages were inconsistent when compared with others along with occasional NPEs (NullPointerException) which occurs when one thread set flag to the existence of a key before placing anything there and meanwhile another Thread enters and tries to fetch the key which is not available there yet resulting in an NPE.

A4. Running times of SEQ and COARSE-LOCK are 9535.60 and 3920.90 which places COARSE-LOCK apprx. 2.5x faster runtimes. The reason behind this is the amount of parallelization that takes place with COARSE-LOCK by splitting the data in proportion with the number of threads. Ideally the speedup would've been a lot more if perfect parallelism was achieved by instead all parallel tasks (Threads) shared the same accumulation data structure which inturn limits how fast we can write to it since according to COARSE-LOCK strategy only one Thread could write at a time. As mentioned in my response to question 2, I expected COARSE-LOCK to finish second to SEQ since it acquires a lock on the whole data structure upon which most of the CRUD (create, read, update, delete) operations occur which render the multi threading helpless since technically the locks are acquired one after another i.e. in order making it partially sequential. Looking at results from B and C, the results did not vary much that indeed came as a revelation to me since the fibonacci I wrote used Dynamic Programming i.e. no recursion hence no significant performance drags because of memory overflow by deep recursion.

A5. The higher computation cost in part C (additional Fibonacci computation) affect no real difference between COARSE-LOCK and FINE-LOCK because of similar reasons I stated in answer A4 where I mentioned that the fibonacci algorithm I implement had an asymptotic time complexity of $O(n)$ which is alot better than the conventional recursion version and computes really quickly which did not really add much computation overhead to the machine. But I did try with much higher numbers e.g. 45 and fine-locking was able to shave off around 1.5s off the runtime which was what I was expecting since the introduction of granular locking on accumulation objects instead of the whole data structure improved the computation.

Word Count Local Execution



The screenshot displays the IntelliJ IDE interface. On the left, the project structure shows 'src/main/java' containing 'WordCount.java'. The main editor window shows the code for 'WordCount' class, which implements the MapReduce framework. The code includes a 'map' method that tokenizes input text and writes tokens to a 'Text' object, and a 'reduce' method that sums the counts for each token. The 'main' method sets up the configuration and runs the job. On the right, the 'Terminal' window shows the execution logs, including the job's progress (map 100%, reduce 100%), completion status, and various counters for file system and map-reduce operations.

```
public static class Map
    extends Mapper<Object, Text, Text, IntWritable>{
    private final static IntWritable one = new IntWritable(
        1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
        ) throws IOException, InterruptedException {
        StringTokenizer itr = new StringTokenizer(value
            .toString());
        while (itr.hasMoreTokens()) {
            word.set(itr.nextToken());
            context.write(word, one);
        }
    }

    public static class Reduce
        extends Reducer<Text, IntWritable, Text, IntWritable> {
        private IntWritable result = new IntWritable();

        public void reduce(Text key, Iterable<IntWritable> values,
            Context context
            ) throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            result.set(sum);
            context.write(key, result);
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        String[] otherArgs = new GenericOptionsParser(conf,
            args).getRemainingArgs();
        if (otherArgs.length != 2) {
            System.err.println("Usage: wordcount <in> <out>");
            System.exit(-1);
        }
        Job job = Job.getInstance(conf);
        job.setJarByClass(WordCount.class);
        job.setMapperClass(Map.class);
        job.setReducerClass(Reduce.class);
```

```
17/01/25 01:12:32 INFO mapreduce.Job: map 100% reduce 100%
17/01/25 01:12:32 INFO mapreduce.Job: Job job_local1006005769_0001 completed successfully
17/01/25 01:12:32 INFO mapreduce.Job: Counters: 35
File System Counters
  FILE: Number of bytes read=559393650
  FILE: Number of bytes written=571652505
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=10290297240
  HDFS: Number of bytes written=72815
  HDFS: Number of read operations=193
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=14
Map-Reduce Framework
  Map input records=21907700
  Map output records=248943500
  Map output bytes=2418234700
  Map output materialized bytes=816400
  Input split bytes=1100
  Combine input records=249396411
  Combine output records=510914
  Reduce input groups=5273
  Reduce shuffle bytes=816400
  Reduce input records=58003
  Reduce output records=5273
  Spilled Records=568917
  Shuffled Maps=11
  Failed Shuffles=0
  Merged Map outputs=11
  GC time elapsed (ms)=1298
  Total committed heap usage (bytes)=5594152960
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=1454048460
File Output Format Counters
```

IDE: IntelliJ

Word Count AWS Execution

Services

Resource Groups

Ayush Kumar Singh

Oregon

Support

Cluster: My cluster

Terminated

Steps completed

Connections:

--

Master public DNS:

ec2-35-167-197-146.us-west-2.compute.amazonaws.com

SSH

Tags:

--

Summary

Configuration Details

ID: j-1SGLOGNXX4MAP

Release label: emr-5.2.1

Creation date: 2017-01-12 09:38 (UTC-5)

Hadoop distribution: Amazon 2.7.3

End date: 2017-01-12 09:53 (UTC-5)

Applications: --

Elapsed time: 15 minutes

Log URI: s3://singhay-cs6240/log/

Auto-terminate: Yes

EMRFS consistent view: Disabled

Termination protection: Off

view:

Network and Hardware

Security and Access

Availability zone: us-west-2a

Key name: aws

Subnet ID: subnet-00a75249

EC2 instance profile: EMR_EC2_DefaultRole

Master: Terminated 1 m4.large

EMR role: EMR_DefaultRole

Core: Terminated 2 m4.large

Visible to all users: All

Task: --

Change

Security groups for sg-85f42dfd (ElasticMapReduce-Master: master)

Security groups for sg-86f42dfe (ElasticMapReduce-Core & Task: slave)

Monitoring

Feedback

English

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Privacy Policy

Terms of Use

Services

Resource Groups

Ayush Kumar Singh

Oregon

Support

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	Bid Price
ig-3KAU2HC1DJZZI	MASTER Master Instance Group	Terminated (1 Requested)	m4.large	0	1	
ig-1NRWX9I15PKX3	CORE Core Instance Group	Terminated (2 Requested)	m4.large	0	1	

Steps

Add step Clone step Cancel step

Filter: All steps Filter steps ...

1 step (all loaded)

ID	Name	Status	Start time (UTC-5)	Elapsed time	Log files
s-2Q1CMFNE8ZEWX	Custom JAR	Completed	2017-01-12 09:44 (UTC-5)	6 minutes	controller syslog stderr stdout

Feedback

English

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Privacy Policy

Terms of Use

ServicesResource Groups

Ayush Kumar SinghGlobalSupport

UploadCreate FolderActions

Search by prefixNonePropertiesTransfers

All Buckets / singhay-cs6240 / logs

	Name	Storage Class	Size	Last Modified
<input type="checkbox"/>	2017-01-12-12-29-30-AC0812132A07EA16	Standard	302 bytes	Thu Jan 12 07:29:31 GMT-500 2017
<input type="checkbox"/>	2017-01-12-12-31-40-C718078F9D7BFA11	Standard	302 bytes	Thu Jan 12 07:31:41 GMT-500 2017
<input type="checkbox"/>	2017-01-12-12-39-43-F0D1A3F2A00785E5	Standard	603 bytes	Thu Jan 12 07:39:44 GMT-500 2017
<input type="checkbox"/>	2017-01-12-12-40-03-A5677EB6EEE430AB	Standard	351 bytes	Thu Jan 12 07:40:04 GMT-500 2017
<input type="checkbox"/>	2017-01-12-13-19-45-1BDF9D5A60C4832E	Standard	376 bytes	Thu Jan 12 08:19:46 GMT-500 2017
<input type="checkbox"/>	2017-01-12-13-21-54-78E81F3A2C103402	Standard	302 bytes	Thu Jan 12 08:21:55 GMT-500 2017
<input type="checkbox"/>	2017-01-12-13-27-41-65345CC7CF58D7C9	Standard	377 bytes	Thu Jan 12 08:27:42 GMT-500 2017
<input type="checkbox"/>	2017-01-12-13-32-22-1FA93F507808507C	Standard	377 bytes	Thu Jan 12 08:32:23 GMT-500 2017
<input type="checkbox"/>	2017-01-12-13-34-37-11BEE92402DC0665	Standard	376 bytes	Thu Jan 12 08:34:38 GMT-500 2017
<input type="checkbox"/>	2017-01-12-14-25-01-0471089D1D8C5966	Standard	377 bytes	Thu Jan 12 09:25:02 GMT-500 2017
<input type="checkbox"/>	2017-01-12-14-25-44-1E31FFE3FE21BE63	Standard	376 bytes	Thu Jan 12 09:25:45 GMT-500 2017
<input type="checkbox"/>	2017-01-12-14-28-12-30C18D5B846FF618	Standard	377 bytes	Thu Jan 12 09:28:13 GMT-500 2017
<input type="checkbox"/>	2017-01-12-14-32-48-21FA881C5D229E1F	Standard	379 bytes	Thu Jan 12 09:32:49 GMT-500 2017
<input type="checkbox"/>	2017-01-12-14-35-33-5635108F8A2A450A	Standard	376 bytes	Thu Jan 12 09:35:34 GMT-500 2017
<input type="checkbox"/>	2017-01-12-15-17-10-CA32AEE96A527B53	Standard	631 bytes	Thu Jan 12 10:17:11 GMT-500 2017
<input type="checkbox"/>	2017-01-12-15-17-24-92FC4BB860B99135	Standard	5 KB	Thu Jan 12 10:17:25 GMT-500 2017
<input type="checkbox"/>	2017-01-12-15-17-29-7699C36325CF9436	Standard	2.4 KB	Thu Jan 12 10:17:30 GMT-500 2017
<input type="checkbox"/>	2017-01-12-15-17-30-6D2AD40F6EB35B78	Standard	1.2 KB	Thu Jan 12 10:17:32 GMT-500 2017
<input type="checkbox"/>	2017-01-12-15-17-32-60C86412288BE7F4	Standard	1.8 KB	Thu Jan 12 10:17:33 GMT-500 2017
<input type="checkbox"/>	2017-01-12-15-17-39-ABE01771AC36A53F	Standard	457 bytes	Thu Jan 12 10:17:40 GMT-500 2017
<input type="checkbox"/>	2017-01-12-15-17-40-8E78502DA1936504	Standard	4.1 KB	Thu Jan 12 10:17:41 GMT-500 2017

FeedbackEnglish

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

ServicesResource Groups

Ayush Kumar SinghGlobalSupport

UploadCreate FolderActions


Search by prefixNonePropertiesTransfers

All Buckets / singhay-cs6240 / output

	Name	Storage Class	Size	Last Modified
<input type="checkbox"/>	_SUCCESS	Standard	0 bytes	Thu Jan 12 09:50:58 GMT-500 2017
<input type="checkbox"/>	part-r-00000	Standard	23.3 KB	Thu Jan 12 09:50:48 GMT-500 2017
<input type="checkbox"/>	part-r-00001	Standard	23.7 KB	Thu Jan 12 09:50:57 GMT-500 2017
<input type="checkbox"/>	part-r-00002	Standard	24 KB	Thu Jan 12 09:50:58 GMT-500 2017


FeedbackEnglish


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



Services

Resource Groups





Ayush Kumar Singh

Global

Support

Upload

Create Folder

Actions

Search by prefix

None

Properties

Transfers

All Buckets

/ singhay-cs6240

/ input

	Name	Storage Class	Size	Last Modified
<input type="checkbox"/>	hw1.txt	Standard	1.3 GB	Thu Jan 12 09:16:40 GMT-500 2017


Feedback

English

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.


Privacy Policy


Terms of Use



Services

Resource Groups





Ayush Kumar Singh

Global

Support

Upload

Create Folder

Actions

Search by prefix

None

Properties

Transfers

All Buckets

/ singhay-cs6240

/ log

/ j-1SGLOGNXX4MAP

	Name	Storage Class	Size	Last Modified
<input type="checkbox"/>	containers	--	--	--
<input type="checkbox"/>	hadoop-mapreduce	--	--	--
<input type="checkbox"/>	node	--	--	--
<input type="checkbox"/>	steps	--	--	--

Feedback

English

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Privacy Policy

Terms of Use