Assigment1
CS6240 Chengze Li

Q1 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)

1.Sequential computation (normal)

input	USC00144932									
output	189	189	189	189	189	189	189	189	189	189
Time	3778	5282	5380	3542	3633	3707	3722	6170	6467	5321

Max: 6467
Min: 3542
Avg: 4700.3

2.No lock computation(normal)

input	USC00144932									
output	165	170	274	235	165	34	189	165	-56	143
Time	1271	3888	2203	3595	1252	1493	4008	1249	22	2199

Max: 4008
Min: 22
Avg: 2118

3.Coarse lock computation(normal)

input	USC00144932									
output	189	189	189	189	189	189	189	189	189	189
Time	3974	3898	3932	3929	3910	3868	3785	3916	3905	3861

Max: 3974
Min: 3785
Avg: 3897.8

4.Fine lock computation(normal)

input	USC00144932									
output	189	189	189	189	189	189	189	189	189	189

Time	3853	3923	3890	4024	3713	3918	3908	3838	3769	3899

Max: 4024
Min: 3713
Avg: 3873.5

5.No share computation

input	USC00144932									
output	189	189	189	189	189	189	189	189	189	189
Time	4054	3925	3916	3825	3838	3781	3975	3895	3779	3948

Max: 4054
Min: 3779
Avg: 3893.6

Using Fibonacci

---

1 sequential computation

input	USC00144932									
output	189	189	189	189	189	189	189	189	189	189
Time	68259	68652	72167	69925	72706	69406	69440	72167	69181	72889

Max: 72889
Min: 68259
Avg: 70479.2

2. no lock computation

input	USC00144932									
output	172	150	150	189	170	239	150	189	197	189
Time	17821	28996	19795	23040	16020	16090	19667	22464	17262	18780

Max: 28996
Min: 16020
Avg: 19993.5

3. coarse lock computation

input	USC00144932									
output	189	189	189	189	189	189	189	189	189	189
Time	19637	23375	20476	18827	18596	19162	18577	18666	19910	22681

Max: 23375
Min: 18577
Avg: 19990.7

4. fine lock computation

input	USC00144932									
output	189	189	189	189	189	189	189	189	189	189
Time	22852	22802	20376	22784	22443	20682	20104	21259	19798	19050

Max: 22852
Min: 19050
Avg: 21215

5. no share computation:

input	USC00144932									
output	189	189	189	189	189	189	189	189	189	189
Time	22796	20381	23121	19487	21177	20016	18566	19260	18498	18607

Max: 23121
Min: 18498
Avg: 20190.9

Q2

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 threads: 4

Speedup: as we can see, the sequential running time is about 70000, and multi-thread running time is about 20000, based on the definition of speed up

Speedup = sequential time / parallel time = 70000 / 20000 = 3.5

Q3

Answer the following questions in a brief and concise manner: (4 points each)

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 thought the fine-lock version should run fastest because it only add lock on the part where the thread will update, so other thread don't have to wait other thread to release the lock if they update different part of shared data structure.

But the result of experiments doesn't meet my expectation, I think the reason may be in this version, it takes some time to split map into different sections, and the effect of splitting map into several sections is not as powerful as I imagine.

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 thought the sequential version is slowest, because it runs the computation in a single thread, other version at least have four threads to work together, there is no doubt that single thread one is slowest

And the experiment's result meet 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.

Yes, the no-lock version will generate incorrect result.

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.)

Based on the result, SEQ is slower than coarse-lock, the reason I think is multi-thread is faster

than single thread.

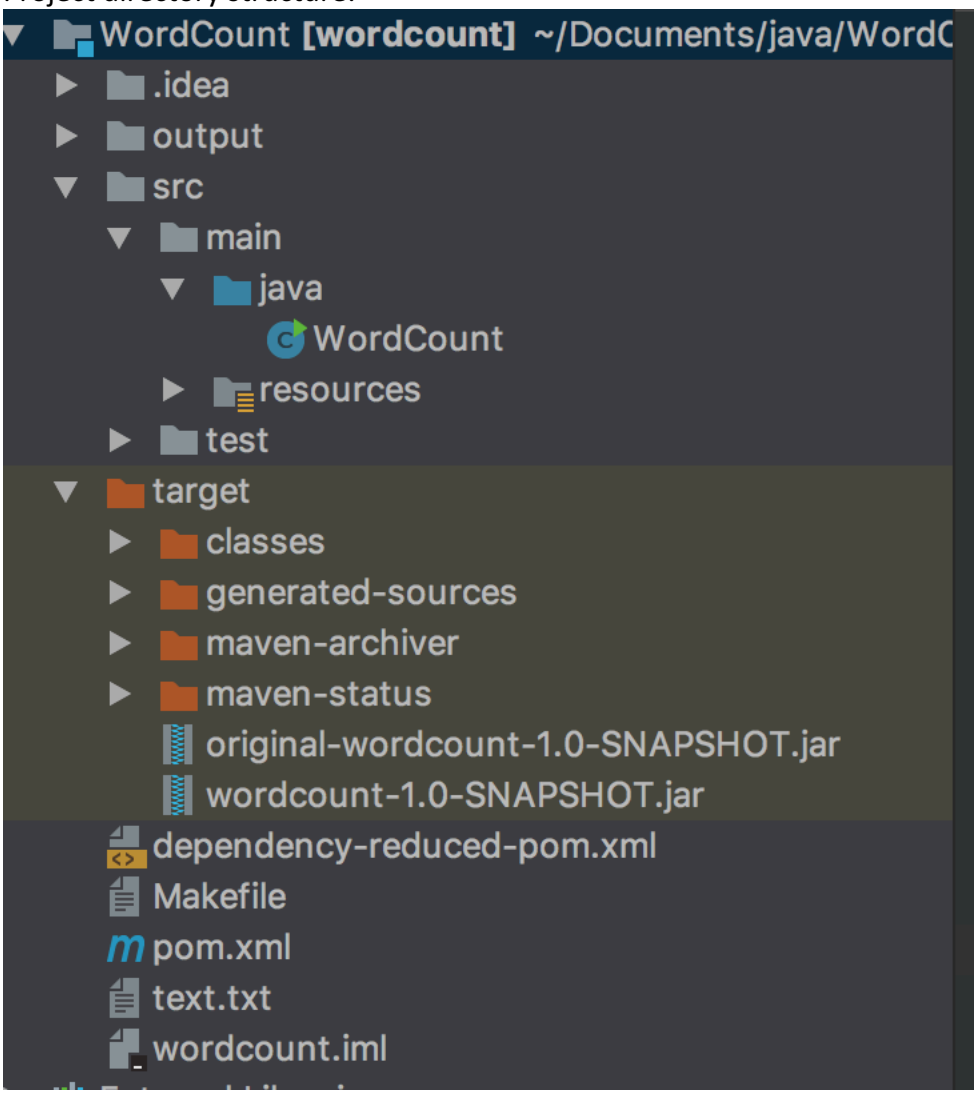
5. **How** does the higher computation cost in part C (additional Fibonacci computation) affect the difference between COARSE-LOCK and FINE-LOCK? Try to **explain** the reason.

It seems the effect on coarse-lock and fine-lock version is almost same, I think the reason is we split shared data structure into relatively large section, so the effect is almost same as split whole map into “one” section. To some extent, fine-lock act as a coarse-lock

---

Word Count

(1) Project directory structure:



(2) console output

```
    File System Counters
        FILE: Number of bytes read=614
        FILE: Number of bytes written=431258
        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=1
        Map output records=7
        Map output bytes=64
        Map output materialized bytes=84
        Input split bytes=118
        Combine input records=7
        Combine output records=7
        Reduce input groups=7
        Reduce shuffle bytes=84
        Reduce input records=7
        Reduce output records=7
        Spilled Records=14
        Shuffled Maps =1
        Failed Shuffles=0
        Merged Map outputs=1
        GC time elapsed (ms)=0
        Total committed heap usage (bytes)=514850816
    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=35
    File Output Format Counters
        Bytes Written=62
1930 [main] DEBUG org.apache.hadoop.security.UserGroupInformation  - PrivilegedAction as:chengze (auth:SIMPLE) from:org.apache.hadc

Process finished with exit code 0
```

（3）screen shots for successful run on emr