CS 6240: Assignment 1

Xiao Wang

GitHub: https://github.ccs.neu.edu/xiaowang/CS6240-MapReduce

1. Weather Data Results

Number of worker threads: 3

Running results with no Fibonacci:

(ms)	Sequential	NoLock	CoarseLock	FineLock	NoSharing
1	2849	ERROR	1612	1514	1470
2	3144	ERROR	1545	1593	1486
3	3152	1774	1912	1599	1451
4	2938	1659	1715	1604	1515
5	2916	1527	1536	1473	1449
6	3113	1507	1628	1529	1428
7	2925	1572	1555	1479	1372
8	2923	1611	1626	1529	1530
9	2763	1564	1599	1508	1484
10	2999	1695	1728	1668	1449
Min	2763	1507	1536	1473	1372
Max	3152	1774	1912	1668	1530
Avg	2972.2	1613.625	1645.6	1549.6	1463.4
SpeedUp(avg)		1.841939732	1.806149733	1.918043366	2.031023644

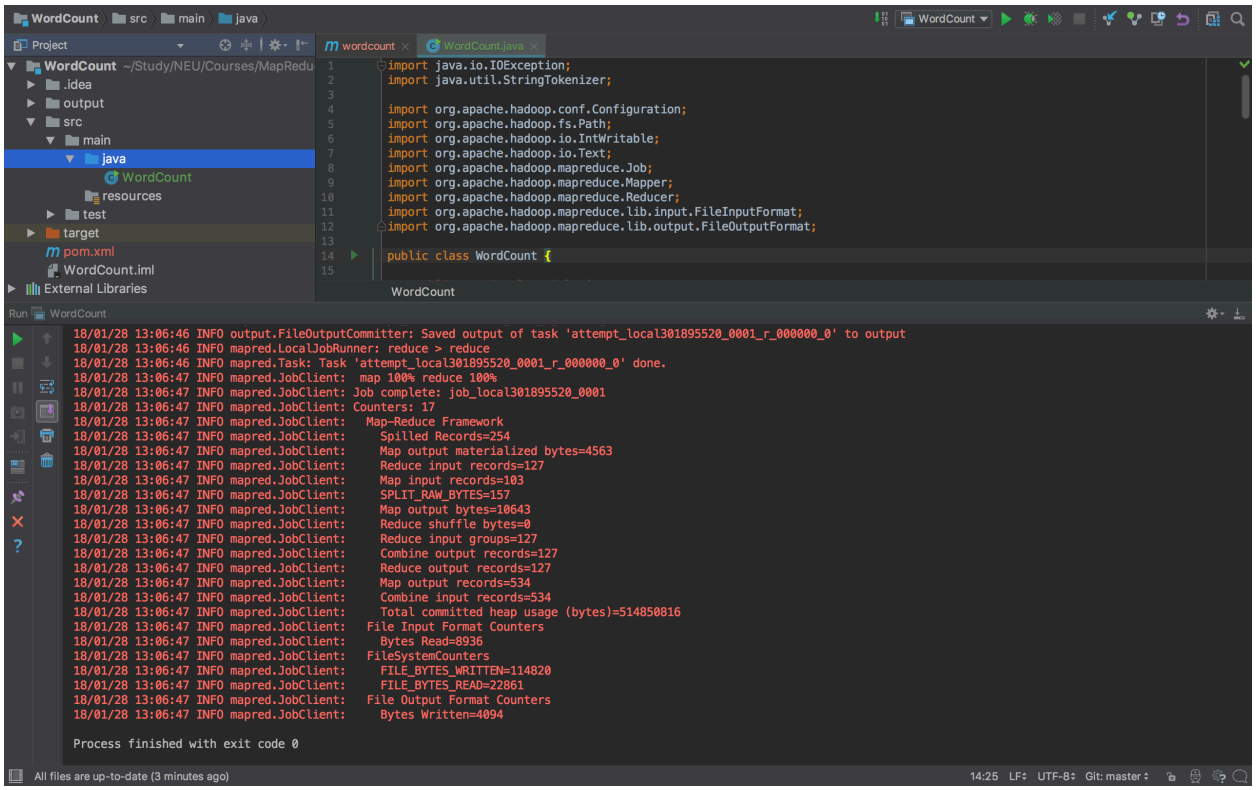
Running results with Fibonacci:

(ms)	Sequential	NoLock	CoarseLock	FineLock	NoSharing
1	12312	5503	10566	5695	5481
2	12082	5446	10378	5556	5376
3	11925	5461	10635	5512	5482
4	12169	5604	10587	5608	5735
5	11974	5824	10575	5574	5676
6	11968	5499	10589	5640	5540
7	11969	ERROR	10944	5902	6833
8	11930	5709	10329	5582	5495
9	12010	5742	11095	5711	5594

10	12191	5708	10848	7272	5655
Min	11925	5446	10329	5512	5376
Max	12312	5824	11095	7272	6833
Avg	12053	5610.66667	10654.6	5805.2	5686.7
SpeedUp(avg)		2.148229563	1.131248475	2.07624199	2.11950692

Q1. I expect NoLock to finish fastest because it doesn't consider consistency, therefore it takes the most advantage of parallelization. The experiments mostly confirm my expectation, although NoSharing performs a little better with no Fibonacci. I think it's because NoLock is using a bigger shared data structure and it requires more overhead.
Q2. I expect Sequential to finish slowest because other versions all take advantage of parallelization in some degree. The experiments confirm my expectation.
Q3. The NoLock version crashes with NullPointerError sometimes because of concurrent accesses.
Q4. Sequential is slower than CoarseLock. I think it's because besides locking time, there's some other computation to do and parallelization reduce the time of this part.
Q5. Higher computation cost reduces the difference between Sequential and CoarseLock. I think the reason is higher computation makes it easier for threads to starve.

2. Word Count Local Execution



3. Word Count AWS Execution

```
[INFO] Deleting /Users/Breezen/Study/NEU/Courses/MapReduce/CS6240-MapReduce/HW1/AWS/target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ wc ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.3:compile (default-compile) @ wc ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 1 source file to /Users/Breezen/Study/NEU/Courses/MapReduce/CS6240-MapReduce/HW1/AWS/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ wc ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /Users/Breezen/Study/NEU/Courses/MapReduce/CS6240-MapReduce/HW1/AWS/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.3:testCompile (default-testCompile) @ wc ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ wc ---
[INFO] No tests to run.
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ wc ---
[INFO] Building jar: /Users/Breezen/Study/NEU/Courses/MapReduce/CS6240-MapReduce/HW1/AWS/target/wc-1.0.jar
[INFO]
[INFO] --- maven-shade-plugin:2.4.3:shade (default) @ wc ---
[INFO] Replacing original artifact with shaded artifact.
[INFO] Replacing /Users/Breezen/Study/NEU/Courses/MapReduce/CS6240-MapReduce/HW1/AWS/target/wc-1.0.jar with /Users/Breezen/Study/NEU/Courses/MapReduce/CS6240-MapReduce/HW1/AWS/targe
t/wc-1.0-shaded.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 4.159 s
[INFO] Finished at: 2018-01-28T19:03:06-05:00
[INFO] Final Memory: 24M/224M
[INFO] ------------------------------------------------------------------------
aws s3 cp target/wc-1.0.jar s3://seanxwang-wordcount
upload: target/wc-1.0.jar to s3://seanxwang-wordcount/wc-1.0.jar
aws s3 rm s3://seanxwang-wordcount/ --recursive --exclude "*" --include "output*"
delete: s3://seanxwang-wordcount/output/_SUCCESS
delete: s3://seanxwang-wordcount/output/part-r-00000
aws emr create-cluster \
               --name "WordCount Cluster" \
               --release-label emr-5.2.1 \
               --instance-groups '[{"InstanceCount":4,"InstanceGroupType":"CORE","InstanceType":"m4.large"},{"InstanceCount":1,"InstanceGroupType":"MASTER","InstanceType":"m4.large
"}]' \
            --applications Name=Hadoop \
            --steps '[{"Args":["WordCount","s3://seanxwang-wordcount/input","s3://seanxwang-wordcount/output"],"Type":"CUSTOM_JAR","Jar":"s3://seanxwang-wordcount/wc-1.0.jar","Actio
nOnFailure":"TERMINATE_CLUSTER","Name":"Custom JAR"}]' \
               --log-uri s3://seanxwang-wordcount/log \
               --service-role EMR_DefaultRole \
               --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole,SubnetId=subnet-197e4152 \
               --region us-east-1 \
               --enable-debugging \
               --auto-terminate
j-1BB9NX8IH8QMA
Seans-MacBook-Pro:AWS Breezen$
```

Webpage Screenshot