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.

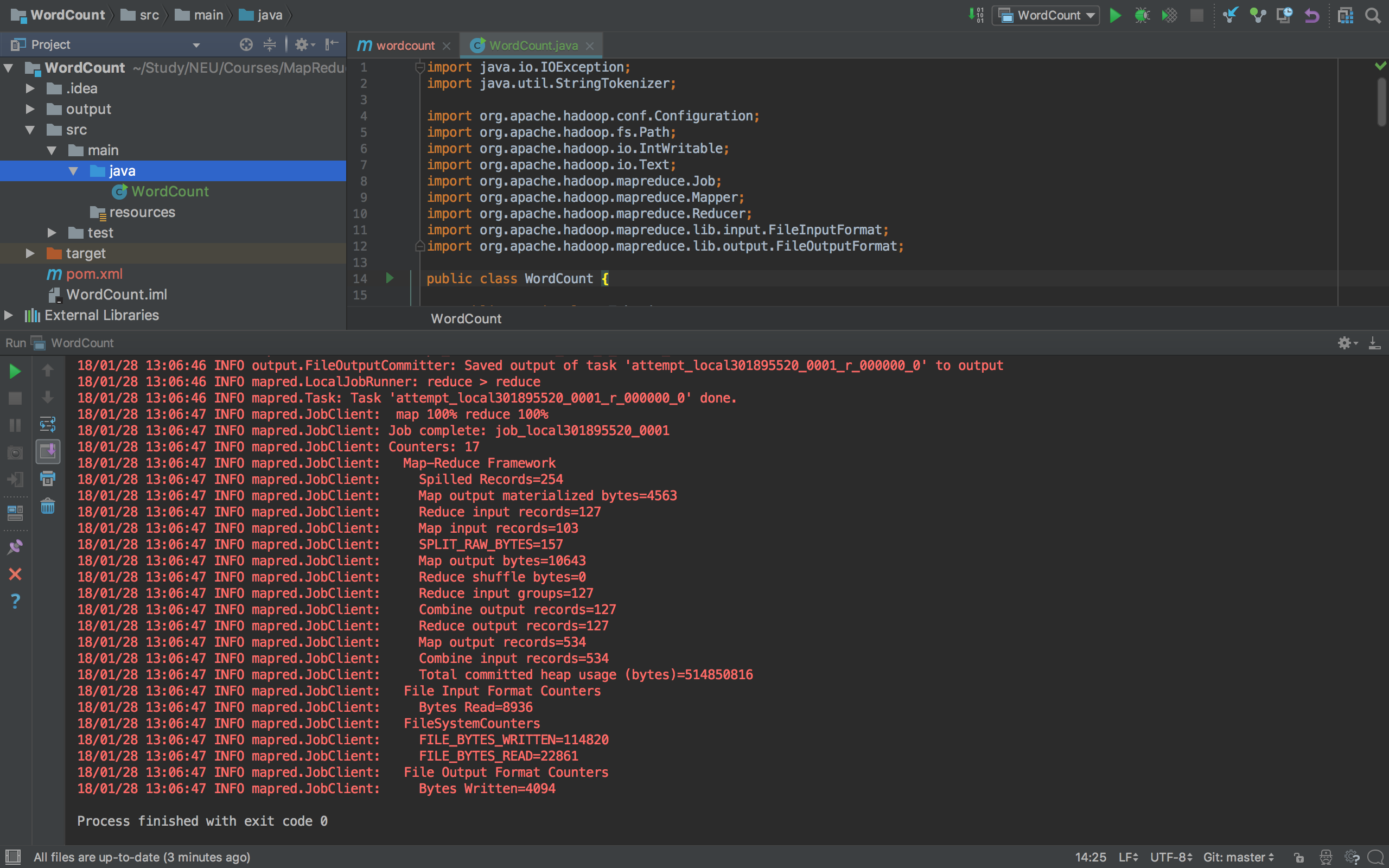
**Q2.** I expect Sequential to finish slowest because other versions all take advantage of parallelization in some degree. The experiments confirm my expectation.

**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.

1. Word Count Local Execution



1. Word Count AWS Execution



