CS 131 Homework 3: Java shared memory performance races

Elizabeth Flynn
UCLA

Abstract

Java's synchronized feature updates shared memory representations of the state of a simulation in the context of multithreading. This homework assignment seeks to understand if we can increase efficiency by removing the traditional safety implemented through the synchronized feature in Java. Through creating an unsynchronized version that lacks the keyword synchronized and running metric tests, we can test whether multithreading can become faster.

The given task is to see the different tradeoffs in performance for varying types of array sizes, thread types, and synchronization in Java.

1. Performance

The results show that for synchronized operations runtimes increased significantly For instance, Platform 1 Synchronized 5 took 2409.20 ns, compared to Platform 1 Null 5 at 0.567871 ns. However it seems that synchronization was faster for virtual threads at low threads than platform, yet it followed the same trend of increased time as threads increased. For Synchronizes 40 thread 5 array virtual the time was 481.450ns but the Platform version was 672.041 ns.

Graph

The run time for a singular thread that was virtual and synchronized however was 1.73216

2. Code changes and Testing

From our given code we actually did not have much to change. Our first step was to create unsynchronized, which was a firm copy of synchronized, and the only changes were certain name aspects and removing the "synchronized" key word. NullState, State, and Synchronized were left just as they were given. SwapTest was changed to add simple features of CPU time. Unsafe memory had a few changes to add a variable mybean which is an instance of ThreadMXBean used to monitor the thread system. Specifically, it helps us monitor CPU time.

Testing was completed on the SEASnet Linux server lnxsrv01 for each variation of the given thread, and array size, thread type, and synchronization value. One concern is that we do not know how different the testing speeds may be on different SEASnet servers.

3. Problems and Difficulties

During the implementation process, several challenges were encountered. Initially, there were difficulties in downloading and configuring Java for local development. Although Java was installed, the javac command was not recognized, which was fixed through the installation of the Java Development Kit (JDK). Additionally, compiling Java files presented a learning curve due to the installation problems from earlier, requiring further exploration of the correct compilation procedures. Additionally, it was a burden to figure out how to acesss the linux server, transfer files, and then use the files on the Linux servers. Issue was resolved by using Cyberduck. Another challenge involved testing the program on UCLA's Linux servers. Given that multiple students may be concurrently running their own programs, there is a possibility that system load could impact performance and test results. Consequently, for more reliable and consistent testing, local execution may be preferable. Testing on servers with lower usage could mitigate potential inconsistencies introduced by high system demand

3. Conclusion

For small thread counts (such as for 1 thread), virtual threads tend to have slightly higher total real times than platform threads, and platform times were generally faster. With higher thread counts, virtual threads time got worse when given more threads. When using synchronized it significantly slowed down the real swap time and computational time dramatically on both platform and virtual. Thread performance overall was dependent on the type of thread, array size, and whether synchronizing was on or not. Generally as the array size increased, so did processing time.

4. Appendix

Data below was tested on lnx server 1.

time java UnsafeMemory Platform 1 Null 5 10000000

Total real time 0.00113574 s

Average real swap time 0.567871 ns

real 0m0.127s

user 0m0.106s

sys 0m0.053s

time java UnsafeMemory Platform 8 Null 5 10000000

Total real time 0.00126375 s
Average real swap time 0.631874 ns
real 0m0.131s
user 0m0.122s
sys 0m0.046s

time java UnsafeMemory Platform 40 Null 5 10000000
Total real time 0.108990 s
Average real swap time 54.4952 ns
real 0m0.267s
user 0m0.647s
sys 0m0.051s

time java UnsafeMemory Platform 1 Null 100 10000000
Total real time 0.0145265 s
Average real swap time 145.265 ns
real 0m0.142s
user 0m0.124s
sys 0m0.058s

time java UnsafeMemory Platform 8 Null 100 10000000
Total real time 0.219903 s
Average real swap time 2199.03 ns
real 0m0.357s
user 0m1.324s
sys 0m0.070s

time java UnsafeMemory Platform 40 Null 100 10000000
Total real time 0.222730 s
Average real swap time 2227.30 ns
real 0m0.351s
user 0m1.300s
sys 0m0.073s

time java UnsafeMemory Virtual 1 Null 5 10000000
Total real time 0.00328572 s
Average real swap time 1.64286 ns
real 0m0.137s
user 0m0.125s
sys 0m0.051s

time java UnsafeMemory Virtual 8 Null 5 10000000
Total real time 0.0854920 s
Average real swap time 42.7460 ns
real 0m0.232s
user 0m0.533s
sys 0m0.054s

time java UnsafeMemory Virtual 40 Null 5 10000000
Total real time 0.112706 s
Average real swap time 56.3528 ns
real 0m0.254s
user 0m0.688s
sys 0m0.048s

time java UnsafeMemory Virtual 1 Null 100 10000000
Total real time 0.00463047 s
Average real swap time 46.3047 ns
real 0m0.138s
user 0m0.139s
sys 0m0.041s

time java UnsafeMemory Virtual 8 Null 100 10000000
Total real time 0.178577 s
Average real swap time 1785.77 ns
real 0m0.322s
user 0m1.152s
sys 0m0.046s

time java UnsafeMemory Virtual 40 Null 100 10000000
Total real time 0.225480 s
Average real swap time 2254.80 ns

real 0m0.378s

user 0m1.429s

sys 0m0.049s

time java UnsafeMemory Platform 1 Synchronized 5 10000000

Total real time 0.240920 s

Average real swap time 2409.20 ns

real 0m0.379s

user 0m1.542s

sys 0m0.052s

time java UnsafeMemory Platform 8 Synchronized 5 10000000

Total real time 1.06352 s

Average real swap time 531.759 ns

real 0m1.197s

user 0m2.575s

sys 0m0.437s

time java UnsafeMemory Platform 40 Synchronized 5 10000000

Total real time 1.34408 s

Average real swap time 672.041 ns

real 0m1.479s

user 0m3.087s

sys 0m0.506s

time java UnsafeMemory Platform 1 Synchronized 100 10000000

Total real time 0.0150180 s

Average real swap time 150.180 ns

real 0m0.146s

user 0m0.126s

sys 0m0.060s

time java UnsafeMemory Platform 8 Synchronized 100 10000000

Total real time 0.555466 s

Average real swap time 5554.66 ns

real 0m0.683s

user 0m1.014s

sys 0m0.198s

time java UnsafeMemory Platform 40 Synchronized 100 10000000

Total real time 0.746035 s

Average real swap time 7460.35 ns

real 0m0.882s

user 0m1.388s

sys 0m0.220s

time java UnsafeMemory Virtual 1 Synchronized 5 10000000

Total real time 0.00346433 s

Average real swap time 1.73216 ns

real 0m0.148s

user 0m0.135s

sys 0m0.041s

time java UnsafeMemory Virtual 8 Synchronized 5 10000000

Total real time 0.926537 s

Average real swap time 463.268 ns

real 0m1.080s

user 0m2.085s

sys 0m0.391s

time java UnsafeMemory Virtual 40 Synchronized 5 10000000

Total real time 0.962901 s

Average real swap time 481.450 ns

real 0m1.119s

user 0m1.902s

sys 0m0.409s

time java UnsafeMemory Virtual 1 Synchronized 100 10000000

Total real time 0.00435476 s

Average real swap time 43.5476 ns

real 0m0.149s

user 0m0.130s

sys 0m0.055s

time java UnsafeMemory Virtual 8 Synchronized 100 10000000

Total real time 1.02413 s

Average real swap time 10241.3 ns

real 0m1.172s

user 0m2.443s

sys 0m0.378s

time java UnsafeMemory Virtual 40 Synchronized 100 10000000

Total real time 0.885960 s

Average real swap time 8859.60 ns

real 0m1.021s

user 0m2.021s

sys 0m0.349s