CS 131 - Homework 3
Report and Test Results

Abstract

In today's world of massive tera-byte databases, multi-core processors, and multi-threaded operating systems, it is critical to be able to provide stable data-race free (DRF) operation where data integrity is paramount. In addition, as commercial developers, we must be aware of performance metrics and choose the highest performance approach to guarantee DRF stability and results. The goal of this assignment was to assess the DRF stability and compare the ways we are able to prevent race conditions; specifically, the speed at which each state can attain results.

Environment

For the testing environment I used two UCLA SEASnet servers as test platforms: lnxsrv06 and lnxsrv10. While they were recommended to use in the project specifications, they also had to meet requirements to run the reliability and speed tests. To provide the correct data readings for the selected tests, the servers had to be able to support Java version 13.0.2, which was detailed in the project's specifications. While there did seem to be some performance differences, I attribute the differences to each server's load during the test phase. The differences are not too extreme, therefore they are not a concern (at least in my limited knowledge of DRFs). My tests were performed later in the evening when the SEASnet servers have higher traffic in comparison to traffic earlier in the day, thus the slightly slower results are to be expected.

Additionally, part of the assignment was to use, and familiarize myself with, the java.util.concurrent.atomic.AtomicLongArray package primarily to test the speeds between the AcmeSafeState class and the Synchronized class (as well as the other classes), specifically the performance between a long array and an atomic long array.

Testing Values

The tests were run with similar test harnesses to provide consistent results. Following the given test harness in the project specifications, the shell command used was:

```
time timeout 3600 java UnsafeMemory [state] [# of theads ] [# of swaps]  [array size]
```

I consistently used 100000000 as the number of swaps, then I used [1, 8, 40, 50] as the number of threads, and [5, 100, 200] for the array sizes. These testing values are clearly stated in the test result tables for ease of analyzing.

Test Results

(See test result tables in the following pages) As shown in the test result tables, some test runs took more time as the thread count increased. For example, AcmeSafeState using an array size of 200, took 1.841 seconds for a single thread while 40 threads in the same test configuration took 6.048 seconds. While this could be interpreted as time spent context switching, 50 threads in the same test configuration actually took almost 2 seconds less. However, for the Null state tests, the results were pretty consistent throughout In another scenario between Synchronized and Unsynchronized operations, the Synchronized option added considerable duration regardless of the thread count. However, the times shown indicate that as thread count increases the difference between Synchronized and Unsynchronized also increases at a faster rate.

Conclusion

Looking at the test results, I would be inclined to avoid multi-threaded operation thus avoiding any potential DRF conflicts while also avoiding any synchronization overhead and providing a far simpler operational model. Perhaps, thread and synchronization performance is dynamic based on the real-time system load and hardware characteristics involved, and a periodic test could determine the optimum operational configuration.

Test Results (continued):

Server: lnxsvr10								
Array size: 5								
State	**1 Thread**		**8 Threads**		**40 Threads**		**50 Threads**	
AcmeSafeState	real user sys	0m1.743s 0m1.323s 0m0.048s	real user sys	0m2.471s 0m8.046s 0m0.051s	real user sys	0m3.624s 0m13.000s 0m0.051s	real user sys	0m3.750s 0m13.596s 0m0.059s
Null	real user sys	0m1.417s 0m1.202s 0m0.042s	real user sys	0m0.789s 0m2.007s 0m0.052s	real user sys	0m1.135s 0m1.434s 0m0.051s	real user sys	0m0.940s 0m1.984s 0m0.052s
Synchronized	real user sys	0m1.811s 0m1.777s 0m0.055s	real user sys	0m5.120s 0m6.094s 0m0.187s	real user sys	0m5.142s 0m6.230s 0m0.232s	real user sys	0m5.091s 0m6.011s 0m0.198s
Unsynchronized	real user sys	0m1.327s 0m1.327s 0m0.043s	real user sys	0m2.775s 0m10.443s 0m0.055s	real user sys	0m2.291s 0m8.706s 0m0.046s	real user sys	0m2.754s 0m10.555s 0m0.055s

Server: lnxsvr10								
Array size: 100								
State	**1 Thread**		**8 Threads**		**40 Threads**		**50 Threads**	
AcmeSafeState	real user sys	0m1.339s 0m1.347s 0m0.040s	real user sys	0m3.716s 0m12.583s 0m0.065s	real user sys	0m3.564s 0m12.789s 0m0.061s	real user sys	0m3.595s 0m12.945s 0m0.050s
Null	real user sys	0m1.182s 0m1.177s 0m0.049s	real user sys	0m0.534s 0m1.699s 0m0.047s	real user sys	0m0.597s 0m1.869s 0m0.056s	real user sys	0m0.525s 0m1.582s 0m0.067s
Synchronized	real user sys	0m1.813s 0m1.818s 0m0.055s	real user sys	0m4.611s 0m5.068s 0m0.122s	real user sys	0m4.748s 0m5.615s 0m0.196s	real user sys	0m4.971s 0m5.838s 0m0.196s
Unsynchronized	real user sys	0m1.349s 0m1.343s 0m0.044s	real user sys	0m3.505s 0m13.567s 0m0.042s	real user sys	0m3.584s 0m13.906s 0m0.045s	real user sys	0m3.679s 0m14.240s 0m0.052s

Server: lnxsvr10								
Array size: 200								
State	**1 Thread**		**8 Threads**		**40 Threads**		**50 Threads**	
AcmeSafeState	real user sys	0m1.341s 0m1.337s 0m0.048s	real user sys	0m3.196s 0m10.822s 0m0.048s	real user sys	0m3.257s 0m11.786s 0m0.050s	real user sys	0m3.185s 0m11.489s 0m0.053s
Null	real user sys	0m1.208s 0m1.195s 0m0.053s	real user sys	0m0.530s 0m1.640s 0m0.059s	real user sys	0m1.555s 0m5.708s 0m0.065s	real user sys	0m0.519s 0m1.554s 0m0.064s
Synchronized	real user sys	0m1.804s 0m1.783s 0m0.061s	real user sys	0m5.174s 0m6.466s 0m0.532s	real user sys	0m4.444s 0m5.048s 0m0.158s	real user sys	0m4.514s 0m5.185s 0m0.187s
Unsynchronized	real user sys	0m1.338s 0m1.345s 0m0.042s	real user sys	0m3.303s 0m10.703s 0m0.048s	real user sys	0m3.698s 0m14.330s 0m0.065s	real user sys	0m3.150s 0m12.123s 0m0.053s

Server: lnxsvr06								
Array size: 5								
State	**1 Thread**		**8 Threads**		**40 Threads**		**50 Threads**	
AcmeSafeState	real user sys	0m1.924s 0m1.942s 0m0.045s	real user sys	0m5.079s 0m39.290s 0m0.080s	real user sys	0m3.672s 0m54.245s 0m0.071s	real user sys	0m2.959s 0m43.754s 0m0.075s
Null	real user sys	0m1.561s 0m1.584s 0m0.038s	real user sys	0m0.498s 0m2.588s 0m0.054s	real user sys	0m0.416s 0m3.856s 0m0.070s	real user sys	0m0.525s 0m5.317s 0m0.081s
Synchronized	real user sys	0m2.615s 0m2.642s 0m0.055s	real user sys	0m13.327s 0m34.075s 0m2.196s	real user sys	0m12.483s 0m28.061s 0m1.939s	real user sys	0m11.232s 0m25.042s 0m1.536s
Unsynchronized	real user sys	0m1.733s 0m1.751s 0m0.050s	real user sys	0m2.856s 0m21.428s 0m0.055s	real user sys	0m2.912s 0m43.540s 0m0.085s	real user sys	0m2.930s 0m39.827s 0m0.075s

Server: lnxsvr06								
Array size: 100								
State	**1 Thread**		**8 Threads**		**40 Threads**		**50 Threads**	
AcmeSafeState	real user sys	0m1.836s 0m1.855s 0m0.046s	real user sys	0m5.216s 0m40.675s 0m0.066s	real user sys	0m3.684s 0m54.102s 0m0.095s	real user sys	0m3.589s 0m54.137s 0m0.086s
Null	real user sys	0m1.511s 0m1.539s 0m0.043s	real user sys	0m0.489s 0m2.522s 0m0.043s	real user sys	0m0.414s 0m3.668s 0m0.084s	real user sys	0m0.548s 0m5.725s 0m0.063s
Synchronized	real user sys	0m3.022s 0m3.040s 0m0.059s	real user sys	0m23.160s 1m5.689s 0m9.551s	real user sys	0m21.718s 1m2.377s 0m6.811s	real user sys	0m12.384s 0m25.298s 0m2.077s
Unsynchronized	real user sys	0m2.538s 0m2.562s 0m0.051s	real user sys	0m4.431s 0m33.479s 0m0.062s	real user sys	0m3.702s 0m49.359s 0m0.087s	real user sys	0m5.953s 0m49.346s 0m0.098s

Server: lnxsvr06								
Array size: 200								
State	**1 Thread**		**8 Threads**		**40 Threads**		**50 Threads**	
AcmeSafeState	real user sys	0m1.841s 0m1.850s 0m0.049s	real user sys	0m5.198s 0m40.169s 0m0.066s	real user sys	0m6.048s 0m45.354s 0m0.127s	real user sys	0m4.271s 0m59.860s 0m0.178s
Null	real user sys	0m1.605s 0m1.632s 0m0.050s	real user sys	0m0.459s 0m2.424s 0m0.058s	real user sys	0m0.741s 0m4.376s 0m0.073s	real user sys	0m1.610s 0m6.884s 0m0.080s
Synchronized	real user sys	0m4.100s 0m3.032s 0m0.058s	real user sys	0m13.413s 0m26.819s 0m2.550s	real user sys	0m14.179s 0m38.950s 0m3.586s	real user sys	0m16.329s 0m45.866s 0m6.124s
Unsynchronized	real user sys	0m2.113s 0m2.145s 0m0.044s	real user sys	0m4.769s 0m16.982s 0m0.060s	real user sys	0m3.977s 0m57.488s 0m0.211s	real user sys	0m4.144s 0m57.109s 0m0.137s