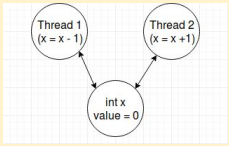


The Effect of Different Hardware and Software on the Runtime Speed of Simple Integer Increment Functions.

Background

Within Computer Science, a great amount of time and energy when programming is optimizing your code in order to decrease the runtime of programs. There are two factors that can affect how fast a program can run. These factors are the way the software of the program was written and the hardware that the software is run on. In regards to software, when a computer is used, there are often several programs and processes running at the same time. This is done by using threads which essentially blocks of code that are able to run at the same time as other threads. So rather than having to go in a specific order, the processes can run at the same time. However, multiple threads can cause your computer to run into an error called a race condition. A race condition in when multiple threads access or modify the same variable at the same time which causes changes in how the program was intended to run. A common example of a race condition is when two threads have the same function of increasing the same integer by one. There are a number of different ways to fix this error but they all essentially do the same function of synchronizing the variable that both functions modify so that only one function is able to access the variable at a time. In regards to hardware, it was important to check whether the runtime of these functions significantly differed between different computers.

Analysis Questions

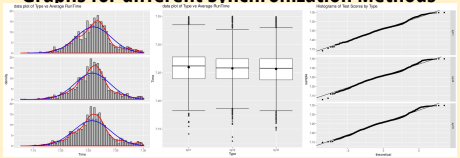
1. Which synchronization methods to fix the error of race conditions have the fastest run time for increasing and decreasing integer values?

2. Is there a significant difference in runtime speed between a Dell Inspiron 13 5000 and an ssh connection between purdue's cs data directory?

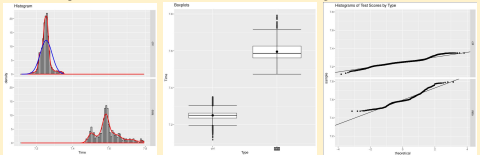
Analysis Methods

A java program was used to collect modify raw data from of runtime speeds from various computers. To answer the first question an ANOVA test was used In order to assess which synchronization method ran the fastest. This test provided information as to whether there is a statistically significant difference between each of the populations of runtimes. To answer the second question a Welch two sample t-test was used to test if there was a statistically significant difference between the two different systems used to collect data. The assumptions made for both tests were met by the data sets.

Graphs for different Synchronization Methods



Graphs for different Hardware Systems



Results

The ANOVA test indicated that there was significant statistical evidence to conclude that the true mean runtime among all three synchronization methods were not equal to each other. This is due to none of the confidence intervals between the synchronization methods containing a zero value for the probal difference between the true means. The Welch Two Sample t-test indicated the p-value of the null hypothesis (the true mean runtime speed on each hardware system) is less than 2.2e-16 which is nearly statistically impossible which allows to conclude that the true mean runtime on both systems are not equal to each other.

ANOVA

```
Fit: aov(formula = Time ~ Type, data = data)

$Type
                 diff          lwr          upr     p adj
syn2-syn1 -0.0020455737 -0.004721670  0.0006305227 0.1723213
syn3-syn1 -0.0029701011 -0.005649932 -0.0002902697 0.0254496
syn3-syn2 -0.0009245274 -0.003600206  0.0017511511 0.6967948
```

Welch Two Sample t-test

```
        Welch Two Sample t-test

data:  comp$Time by comp$Type
t = -125.32, df = 601.07, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.3507441 -0.3399203
sample estimates:
mean in group ctrl mean in group data
          7.250201           7.595533
```

Conclusion

These tests provided two main conclusions for future developers to potentially use to benefit their applications. First off, we concluded that it is faster to run simple processes on a simple computer (Dell Inspiron 13 5000) rather than on Purdue's CS data directory. Which in turn lets allowed guidance as how to speed up the process of collecting data the best. This is probably due to that fact that several people are usually using the Directory at once which is why it makes sense for the system to run slower. We also concluded that of the three synchronization methods, (Method declaration, Atomic Integer and Object Lock) Object Lock ran the fastest of all three methods with Atomic Integer being the second fastest and Method declaration running the slowest. This shows us that if it is applicable to your program, it is most beneficial to use an Object Lock to synchronize variables because it will allow your program to run the fastest.

References

Alexander, G. E., & Crutcher, M. D. (1990). Functional architecture of basal ganglia circuits: neural substrates of parallel processing. *Trends in neurosciences*, *13*(7), 266-271.
Kleiman, S., Shah, D., & Smaalders, B. (1996). *Programming with threads* (p. 48). Sun Soft Press.
Race condition. (2018, April 10). Retrieved from https://en.wikipedia.org/wiki/Race_condition