Lock Free Stack

Ross Wagner

March 2019

1 Lock free stack using descriptors

Using the paper "Lock-Free Dynamically Resizable Arrays" as reference I implemented a lock free stack that supports Push, Pop, and Size operations. I used Descriptor object with WriteDescriptor sub objects to announce operations. Before an operation executes it attempts to complete the operation described by the write descriptor. All of this is done so that multiple memory locations can be updated in a linearizable fashion.

1.1 Performance

As seen in figure 1, the time to complete the operations about halved from 1 thread to 3. It makes sense that the lowest time is at 3 threads because. The performance evaluation was done on a virtual machine with 3 cpus allocated to it. As the thread count exceeds 3 the overhead of the threads begins to outweigh their usefulness.

This evaluation can be done on any machine with java 11 or later installed easily. Sumply navigate to the directory with the file HW2ParallelStackSize.jar and run the command:

```
java -jar HW2ParallelStackSize.jar
```

This will create a lock free stack and run a number of push, pop, and size operations with different numbers of threads then graph the result.

2 Restricted Double-Compare-Single-Swap

I'm going to be honest with you. I had no idea what was going on here. I implemented a Restricted Double-Compare-Single-Swap (RDCSS) in the RDCSS.java file. I takes in 5 AtomicRefereces. If the values pointed to by the first old and new pair match and the values of the second new and old pair match, the second new reference is set to point to what is pointed to by the 5th input.

I do not see where this would be useful in allowing my implementation of the stack to update the head value and size value atomically. A Double Compare and Swap could have been used to do this. In this project I do not see where I would need to compare 2 pairs of values then replace one of them with a new value.

3 Figures

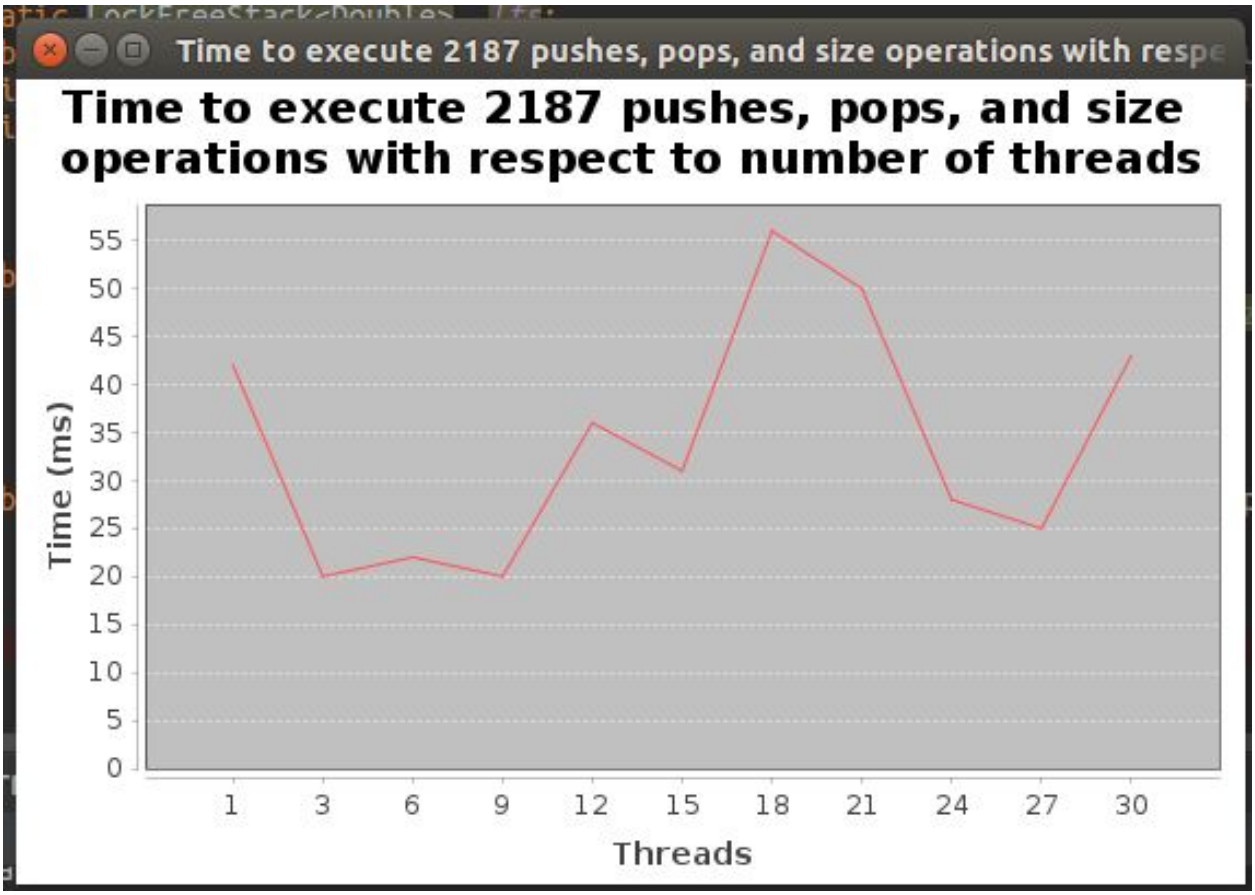

Figure 1 shows the relationship between number of threads and execution time.