**Program 3 Report**

***Part 1:***

Specification/algorithm description about SyncQueue and QueueNode (or student-provided classes:

***SyncQueue:***

SyncQueue has a constructor which creates the queue with the default amount of QueueNodes in it (10) or a constructor that accepts a parameter with the max conditions. SyncQueue also has 4 methods that 1) enque and sleep (condition), dequeue and wakeup (condition) and dequeue and sleep (condition with TID). The enqueueAndSleep method creates a QueueNode with the size of the condition and loop through them to put them to sleep. The dequeueAndWakeup methods use a condition (and possibly a TID) to loop through the queue and wake up the TIDs.

***QueueNode:***

The QueueNode is a simple class that holds a list (but used like a queue) to hold integers (TIDs). The two methods, sleep() and wake(), tell the threads to either wait() or notify(). The QueueNode utilizes synchronized to ensure that the threads are synchronized when making updates.

**Part 2:**

Specification/algorithm description about Test3

The use of Test3 to test the old and new kernels has two important algorithms for which to test the threads, 1) computation and 2) I/O streams. For the computation, the Test3 class passes in the argument of the class name and executes TestThread3a. TestThread3a runs heavy computation via a massive complicated math formula with loops. I/O streams, the Test3 again converts the string to an argument and executes that argument which runs TestThread3b. TestThread3b creates a block of data and writes and reads to that block over a loop.

Performance result when running Test3 on Kernel\_1.java



Performance result when running Test3 on Kernel.java



Performance comparisons between Kernel\_1.java and Kernel.java

Results:

|  |  |  |  |
| --- | --- | --- | --- |
| **# of Pairs of Threads** | **Old Kernel** | **New Kernel** | **Diff (new – old)** |
| 1 | 31265 | 31434 | 169 |
| 2 | 61257 | 61227 | -30 |
| 3 | 93349 | 97906 | 4557 |
| 4 | 131661 | 125220 | -6441 |

Results were not completely clear, it seems that there is minimal difference between the pairs of threads at the beginning but the difference between 4 or more threads could be quite significant. I believe that with less threads, the threads have less competition so the difference isn’t huge. As you add more threads, the processes have more competition and thus, having the wake ups for the I/O starts to make an impact. My results could also be different due to my older computer (although not completely sure this would cause any significant errors).