**CSC139: Operating System Principles**

**Second Assignment**

**Thread Synchronization**

**Table of Results**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test No. | Array Size | Threads | Zero Index | Sequential | Parent Waits | Parent Continuous Check | Parent Semaphores | Minimum |
| 1. | 100M | 2 | 50M + 2 | 40 ms | 66 ms | 74 ms | 58 ms | 0 |
| 2. | 100M | 4 | 75M + 2 | 47 ms | 57 ms | 66 ms | 52 ms | 0 |
| 3. | 100M | 8 | 88M | 53 ms | 53 ms | 26 ms | 23 ms | 0 |
| 4. | 100M | 2 | -1 | 68 ms | 115 ms | 122 ms | 134 ms | 1 |
| 5. | 100M | 4 | -1 | 56 ms | 80 ms | 73 ms | 144 ms | 1 |
| 6. | 100M | 8 | -1 | 72 ms | 121 ms | 37 ms | 199 ms | 1 |

**System Configuration**

**OS:** Ubuntu 16.04 LTS

**Processor:** Intel® Processor 5Y70 CPU @ 1.10GHz × 4

**Number of Cores:** 2

**Number of Threads:** 4

**Base Frequency:** 1.10 GHz

**Max Turbo Frequency:** 2.60 GHz

**Cache:** 4 MB

**Discussion and Observations**

After running multiple tests on the system it was determined that the sequential search slightly changed as the number of threads changed. For the scheme where the parent has to wait for all the threads to finish, the time improved as the number of threads increased. As for the second scheme where the parent keeps checking on threads continuously in a busy waiting loop and terminates as soon as a thread finds the zero, the time also improved dramatically when the thread number is increased as once a thread finds the zero the parent does not wait for all threads to finish. As for the parent using semaphores also the time was improved dramatically when the number of threads was increased.

One of the observation made in this program is that the minimum value changed from 0 to 1 when the zero index was set to -1, this is because the zero was placed out of bounds of the gData array causing the minimum value to be 0 instead of 1. This could have been caught in the part of the main where the argument is validated, since the main is only checking **if(indexForZero < -1 || indexForZero >= arraySize)** it causes the index of -1 to be accepted as a valid value. In the results overall it shows that parent waiting, parent checking and semaphores have the best performance when the number of threads is increased. According to the time of the semaphore, when using a large number of threads to do a specific task can perform the best.