Diana Monreal

Csc139

Second Assignment

Fall 2018

**Table of Results Dedicated Machine in LAB 2003**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test No. | Array Size | Threads | Index for Zero | Search complete | Parent Waits | Parent Continuous Check | Parent Semaphores | Minimum |
| 1. Run1 | 100M | 2 | 50M + 1 | 26 ms | 30 ms | 0 ms | 0 ms | 0 |
| Run2 | 100M | 2 | 50M + 1 | 25 ms | 31 ms | 0 ms | 0 ms | 0 |
| 1. Run1 | 100M | 4 | 75M + 1 | 39 ms | 17 ms | 0 ms | 0 ms | 0 |
| Run 2 | 100M | 4 | 75M + 1 | 1. ms | 17 ms | 0 ms | 0 ms | 0 |
| 1. Run1 | 100M | 8 | 88M | 45 ms | 15 ms | 15 ms | 3 ms | 0 |
| Run2 | 100M | 8 | 88M | 45 ms | 15 ms | 15 ms | 1 ms | 0 |
| 1. Run1 | 100M | 2 | -1 | 51 ms | 31 ms | 31 ms | 30 ms | 1 |
| Run2 | 100M | 2 | -1 | 52 ms | 30 ms | 30 ms | 29 ms | 1 |
| 1. Run1 | 100M | 4 | -1 | 52 ms | 17 ms | 23 ms | 17 ms | 1 |
| Run2 | 100M | 4 | -1 | 51 ms | 17 ms | 22 ms | 17 ms | 1 |
| 1. Run1 | 100M | 8 | -1 | 52 ms | 15 ms | 24 ms | 14 ms | 1 |
| Run2 | 100M | 8 | -1 | 52 ms | 15 ms | 23 ms | 15 ms | 1 |

**System Configuration:**

[diana@sturgeon:38]> lscpu

Architecture: x86\_64

CPU op-mode(s): 32-bit, 64-bit

Byte Order: Little Endian

CPU(s): 8

On-line CPU(s) list: 0-7

Thread(s) per core: 2

Core(s) per socket: 4

Socket(s): 1

NUMA node(s): 1

Vendor ID: GenuineIntel

CPU family: 6

Model: 60

Model name: Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz

Stepping: 3

CPU MHz: 800.000

BogoMIPS: 7183.46

Virtualization: VT-x

L1d cache: 32K

L1i cache: 32K

L2 cache: 256K

L3 cache: 8192K

NUMA node0 CPU(s): 0-7

[diana@sturgeon:39]>

**Discussion and Conclusions:**

After running two run of tests on the system, sequential search is certainly the slowest in all cases since it is a single threaded linear search. After setting the index to 0, the following slowest should to be when the parent that waits for every one of it's children, since it goes into a busy loop until the point that it is finished. On the off chance that the zero is set to - 1, Semaphores lose some speed since it needs to lock, then wait for the process to finish for each thread that searches, just to discover no 0. The third longest method when there is no zero, is the point at which the parent persistently keeps an eye on the children to finish. The speediest method is always semaphores, regardless of whether by a little when there is no zero.

In sequential search, we need to emphasize through every component until the point when a zero is found. At that point it closes. The second one which waits for children stops when every one of the threads complete, regardless of whether the zero has just been found. The third where the parent checks the children implies that the parent slows down the processing by going back verifying whether 0 was found, however it is still faster than waiting for the children to return since it's proactively checking for 0 instead of being in a busy loop. The last search with semaphores has the parent set in the waiting queue by utilizing semaphore wait commands, so it authorizes assets that can be utilized on different strings once it discovers 0 and canceled. On the off chance that there is no 0, the threads should wait since the critical section locked by means of the semaphores, which keeps us the program from checking when the thread count has achieved the desired value.