Parallel Computing - Lab 1

Liron Mizrahi - 708810

July 25, 2016

1 Accessing Multi-Dimensional Arrays

When accessing multi-dimensional array, it can be accessed column-wise or row-wise. Each row will be stored in contiguous memory blocks and may be seperate from other row's memory. So when the array is accessed column-wise, it will have to load every row into cache seperately to access one element. Whereas if it is accessed row-wise, each row can be loaded into cache and every value can be read from cache before loading the next row.

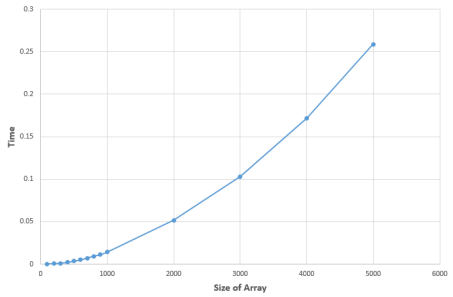

Figure 1.1: Row-wise

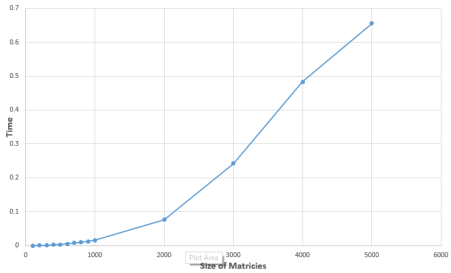

Figure 1.2: Column-wise

It can be seen from the two graphs above that accessing arrays row-wise is significantly faster than accessing it column-wise.

2 Display Output From Threads

If the amount of threads being used is more than the amount of processing units available then some threads will have to be scheduled. The scheduler of the OS being used will determine which threads will run and for how long. This makes the effects of threads unpre-

dictable. The scheduler will keep switching threads which results in the same program giving different outputs.

3 Vector Dot Product

3.1 Effect of Varying the Number of Threads

By increasing the number of threads, some computation can be done at the same time. However if the amount of threads being used is more than the amount of processing units available then there will be scheduling and not all computation will be done in parallel. In the case of calculating the dot product, the actual computations are simple and therefore the startup cost of the threads will outweigh the calculations done per thread.

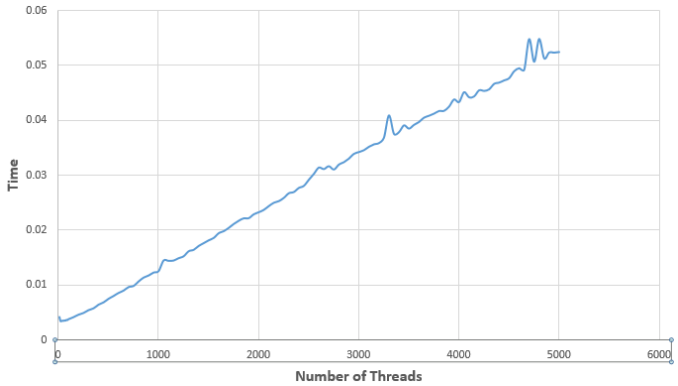

Figure 3.1: Effect of increasing the number of threads for array sizes of 5000

As seen in the graph above, the effect of increasing the threads is not always going to result in better performance. It can also be seen that for very small amount of threads, close to the amount of processing units actually available in the system, the time decreases as all the threads are running in parallel with minimal scheduling.

3.2 Effect of Varying the Size of the Vectors

As the size of the arrays increase, the amount of computations scales proportionally. The work load would be distributed between the threads as to minimize any idling.

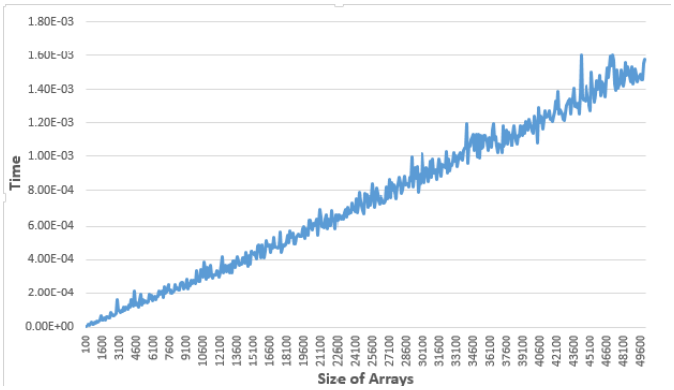

Figure 3.2: Effect of increasing the size of the arrays

As expected the time taken to complete the dot product will increase proportionally at a linear rate.

3.3 Parallelizing Random Number Generation

Theoretically parallelising the random number generator should increase the performance. However, the startup cost of the threads and the fact that these programs were run in a virtual machine will cause more scheduling to occur and actually give worse time practically.

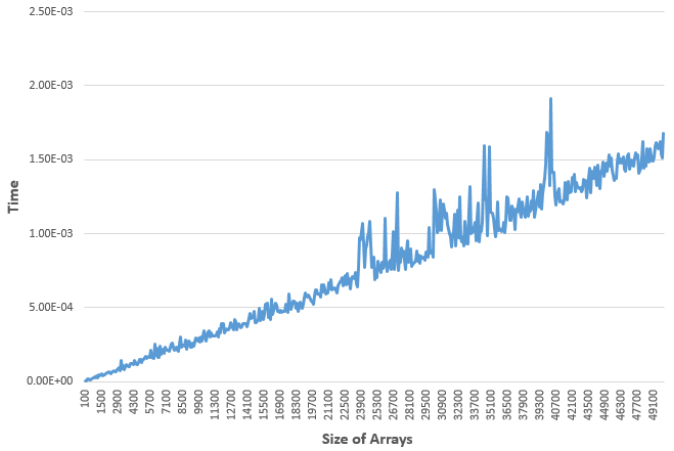

Figure 3.3: Effect of parallelising the RNG

As seen in the graph, the times taken are worse than a sequential RNG in this instance.