Introduction

This evaluation metrics are adopted and modified from the first journal cited here as both types of research are very similar and the journal can be considered as previous art of our research. Hence, the graphs are also shown as examples where it is expected that our research will also result in a much closer result.

The performance distance between the CPU and GPU is going to be evaluated in two perspectives:

1. The rate of computational cost (time) growth with the number of parallel instances.
2. The rate of computational cost growth with the amount of data needs to be handled.

Our research is going to be conducted with the financial analytical data from a stock exchange sector. The amount of data will vary with the problems, the weight of the problem and the context, for the size of calculations in a typical computational problem greatly affects the amount of data handled by the algorithm which processes the problem. Though it affects very less in the computation cost. On the other hand, the number of parallel instances of similar problems does not affect the amount of data handled by the algorithm since it is bounded to the algorithm and the problem but greatly affects the computational cost.

The CPU is good at processing serial instructions involved in a wide range of memory as it has a reduced context switching time but the GPU has worse performance on such instructions. However, the GPU is good at processing several threads in parallel that involves few instructions that use less memory. Therefore, it will be fair to evaluate the performance of the CPU and GPU with regards to the number of parallel instances and the amount of data handled. Section 1 illustrates the evaluation metrics related to the number of parallel instances and section 2 goes through the evaluation based on the amount of data to be handled.

The graphs shown are given for understanding purposes, from experimentally collected data of the genome-related research. Since the evaluation results from test problems are identical for all the CPU and GPU experiments, the graphs of the genome processing results are considered and also expected as our research outcome.

1. The number of parallel instances at once

We need to execute a vast number of parallel instances spread widely in both the CPU and GPU and statically analyze the time taken for them to execute the input of the equal number of parallel instances. The number should be ranging from one instance up to 50,000 instances to have more accurate results.

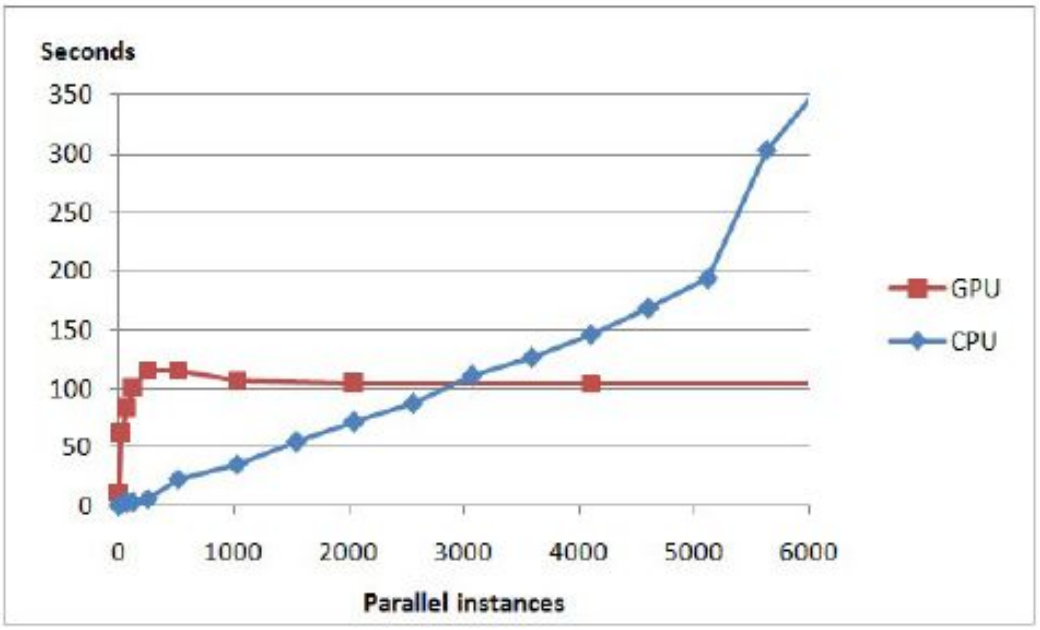

Figure 1: Computational cost against the number of parallel instances [1].

2. The amount of data handled

The CPU and GPU implementations are going to be tested with the problem of sizes ranging from 2 to 30. If we want to do it specifically for stock applications, we need to collect data set having the weight range required and execute them in both the CPU and GPU to measure the computational cost. However, it can also be evaluated by varying the size (parameters) of an object being tested in a testing algorithm. To reduce the effects and benefits gained from parallel instances, the objects are to be run with 256 parallel instances. The graph that resulted in the research conducted by the author of the journal [1] is shown in figure one. His research was with similar conditions but the research was specific to genome kind of problems. As the CPU has less context switching time, the CPU is often more efficient compared to the GPU when it comes to handling data. The research of the author has shown that the computational costs with CPU the same regardless of the amount of data but the GPU had growth with the amount of data.

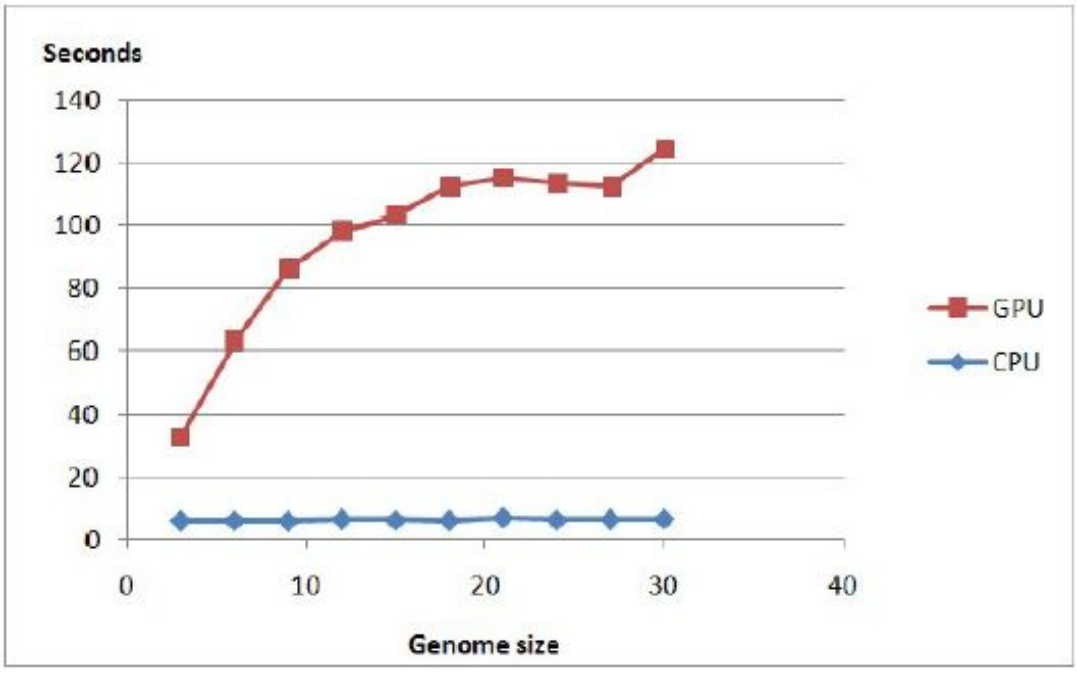

Figure 2: Computational cost against the amount of data handled (genome size)[1].

In light of these results, it is interesting to investigate whether the amount of data handled has any effect on the breakpoint at which the GPU became better than the CPU. To evaluate this, exactly the same experiment performed in the evaluation metrics was run once again but this time with a much smaller amount of data. In this case, too, the CPU starts out better, but the breakpoint now comes earlier, at 1700 instances rather than 3000. This indicates that the breakpoint comes earlier when a smaller amount of data is being handled.

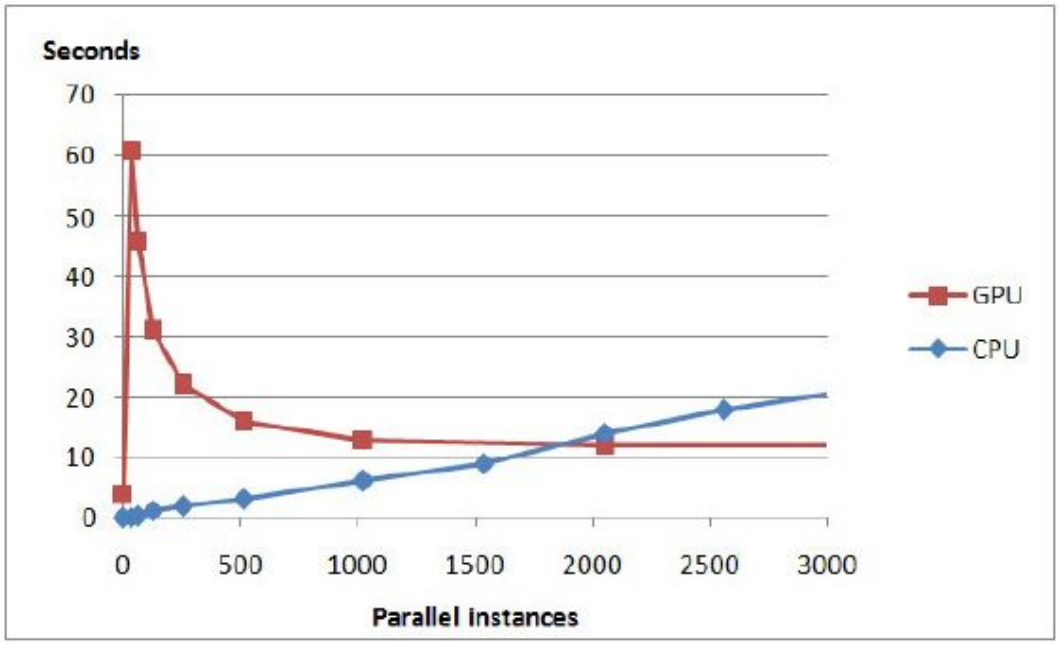

Figure 3.Computational cost in relation to the number of parallel instances when a small amount of data is handled [1].

The hump in the computational cost of the GPU at very small numbers of parallel instances is due to the fact that the GPU does not like to be idle. Below 1024 parallel instances, the GPU cannot fill one thread block, which results in idling and is associated with driver and memory management overheads that are computationally costly. The overhead arises because the program needs to use driver code to communicate with the GPU and spends time copying data back and forth from the GPU.

References

[1] A. Syberfeldt and T. Ekblom, “A comparative evaluation of the GPU vs. The CPU for parallelization of evolutionary algorithms through multiple independent runs”, International Journal of Computer Science & Information Technology (IJCSIT) Vol 9, No 3, June 2017.[Accessed: 31- Dec- 2019] [Online]. http://aircconline.com/ijcsit/V9N3/9317ijcsit01.pdf