Runtime Performance Evaluation of GPU and CPU using a Genetic Algorithm Based on Neighborhood Model

Vincent W. A. Tadaiesky¹, Ádamo L. de Santana¹, Lilian de J. C. Dias¹, Ivan de I. de Oliveira¹, Antonio F. L. Jacob Junior², and Fábio M. F. Lobato¹

¹Laboratory of Computational Intelligence and Operational Research, Federal University of Pará, Belém, Pará, Brazil
²Center of Technological Sciences, University of Amazon, Belém, Pará, Brazil

Abstract—Bio-inspired techniques like Genetic Algorithms have a comprehensive applicability to optimization problems. Given the ease of parallelism implementation inherent of these techniques several researches have been developed in such area making use of parallel platforms, especially the CUDA platform. However, the majority of these works are focused on strategies to improve the algorithms convergence without concern for the performance against their runtime. In this context, the present research introduces a runtime comparison of a Genetic Algorithm implementation in CUDA and traditional CPU aiming to investigate the performance difference between them. The results showed that although the Genetic Algorithm has a native parallelism, the CPU overcomes the GPU implementation due to a hardware limitation.

Keywords: Genetic Algorithms; CUDA; Performance Evaluation; GPGPU; CPU;

1. Introduction

Genetic Algorithms (GA) has demonstrated to be a powerful tool to find solutions for optimization problems [1]. Nevertheless, this technique demands a high computational cost as a result of the evaluation of all possible solution in each generation [2]. Aiming to attenuate such problem, parallel processing systems like the General Purpose Graphical Processing Unit (GPGPU) are being utilized [2] [3] [4].

GPGPU are graphic processors used not only to image rendering, but to any problem requiring massive data parallel processing. To facilitate its use, the NVIDIA® corporation provides an API that uses C/C++ languages to program in their graphics cards, called CUDA.

Concerning the efficiency of the CUDA platform, several works have been conducted in order to evaluate its performance, including problems which involve the use of GA. These problems, generally, are aimed to evaluate the comparison with the corresponding CPU implementation.

In this context, the present work proposes a performance comparison of Genetic Algorithms in CPU and GPU, focusing on their runtime and not in their solution convergence in order to verify the best environment for its implementation.

This paper is organized as follows. Related Work are showed in Section 2 where some similar works using GA or CUDA are described; Test configuration are presented in Section 3, describing how the benchmark was performed; The Section 4 presents the results obtained. Finally, the Section 5 presents an analysis of the results and future works.

2. Related Works

Many researches involving genetic algorithms and CUDA have been developed, in general, in the pursue of GA implementation strategies capable of taking advantage of the parallelism inherent to the platform. There are also researches which aim to benchmark the performance of CUDA in various hardware configurations and operating systems.

Regarding the design of GA, [3] presents a island-based GA, which the individuals are allocated in islands. The population in each island evolves independently from each other. This implementation proves to be effective when each island is allocated in a CUDA core, since each CUDA core can work isolatedly, requiring the synchronization only at the end of the GA process.

In [2], the GA strategy is presented using stationary state selection. This implementation defines that the two individuals with best fitness are selected for crossover while the two worst individuals are replaced by their descendants. Given the execution independence between cores, this strategy is proved efficient because it does not need to synchronize the population.

Concerning the performance evaluation of CUDA, [4] develops a benchmark of the CUDA platform used in various hardware configurations and Microsoft Windows® operating system versions. It was verified that, apart from the configuration used, the GPU load is always superior when the problem can be processed in parallel.

It is important to notice that the benchmarking process can change dogmas of some areas, as in [5]. There the authors compared the most used programming languages in bioinformatics. The parameter measured was the runtime of recurrent algorithms in the area, running on two operating systems: Windows® and Linux. The results defeated the
most commonly used programming language, Perl, while Java was, surprisingly, as efficient as C and C++.

In its context, the directions of this area of research, and how it has been approached, motivated this study, denoting the comparison of CUDA and CPU performances running a GA; as shown in the next section.

3. Experiment Configuration

This research reports the performance comparison of a GA based on neighborhood model in the CUDA and CPU environment. For such, the following computer configuration was used: GNU/Linux operating system with Intel® Core™ i5-2310 CPU @ 2.90GHz x 4 with GPU NVIDIA GeForce 8400GS with the CUDA 5.0 enabled.

The objective of the implemented GA is the optimization of the following three benchmark functions obtained from [6]: g01, g04 and g07. They were chosen due to their wide number of variables and restrictions, which collaborate to the CUDA performance, ensuring a massive parallel processing of the data. The GA codification for each benchmark function is identical, except for the fitness calculation that are adapted for each one.

At this point, the GA project for the CUDA platform was coded according to the hardware limitation. In the neighborhood model, each individual is allocated in one core of the GPU. Given the 16 cores available in the graphic card, the model was adapted to allocate each individual in one thread, being the population composed of 500 individuals. The individuals would then compete to be executed in one of the GPU cores.

There were performed 10 iterations of the GA for each function, both in CPU and GPU, with elitism of 25%. The stop criteria used was the number of 100,000 generations so they do not contaminate the test results due to the difference between their convergence solution. It is noteworthy that the object of this study was the runtime comparison, not the GA performance related to its convergence.

4. Results

The results obtained from the experiment’s execution are summarized in Table 1 for the CPU and GPU runtime. Each column represents one of the optimization functions chosen, and each row is the corresponding to architecture for GA implementation.

From the results a significant difference could be noticed between the runtimes, where the GPU are more than 60 times higher than the CPU for every function. These values are the opposite of the results described in [4], where the GPU outperforms the CPU regardless of the environment configuration on both operational system and hardware.

Taking into consideration the project utilized to exploit the CUDA platform parallelism, the gap between the results indicate a hardware limitation found in the graphic card used, caused by the number of precessors core and mostly by the available memory space in them, i.e. the memory was the limiting factor to the GA chromossome encoding.

5. Conclusions and future work

Several researches involving GAs in the CUDA platform are currently being developed, especially on new strategies to take complete advantage of the parallelism offered by the platform. In this scope, this work implemented a GA based on neighborhood model, using benchmarking functions, in order to evaluate the performance of the CUDA platform in relation to CPU.

The experiments indicated that there are a substantial difference between the implementations, possibly caused by a hardware limitation of the CUDA hardware utilized - CUDA platform performance is directly related to the number of cores in the processor and the available amount of memory. Thus, despite the inherent parallelism present in some bio-inspired techniques, its implementation in a multiprocessor environment, such as GPU, do not guarantee an expected improve in performance.

Considering a memory limitation observed during the experiments, it is intended to verify, in future works, algorithms that require the communication CPU/GPU, specially in the cache memory of the graphic processors, e.g. problems that occurs when the encoded GA chromossome exceeds the cache memory available in the GPU processors.

References


<table>
<thead>
<tr>
<th>Table 1: Linux CPU and GPU runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
</tr>
<tr>
<td>CPU</td>
</tr>
<tr>
<td>GPU</td>
</tr>
</tbody>
</table>
