A Study in Cuda Usage

Pedro Carrega[1] and Vasco Ferreira[1]

Departamento de Informtica da Faculdade de Cincias da Universidade de Lisboa
{fc49080,fc49070}@alunos.fc.ul.pt

Abstract. This paper was developed with the purpose to explain GPU programing and the improvements that can be made. To demonstrate this we used the Floyd-Warshall algorithm since it is very demanding for the CPU due to the high number of computations required. During our research we discovered that the available materials for learning GPGPU are quite difficult to understand for those who are looking to learn the basics of GPU programing. For that we developed this paper explaining from scratch GPGPU and some of the most significant improvements that can be made. We will start with a simple sequential solution of the algorithm in CUDA and from there go step by step till we use all the tools and processing power that a GPU has to offer. This paper is helpful for someone who intends to learn the basics of GPGPU or could be used as a teaching guide in an IT course to introduce GPGPU to students.

Keywords: GPGPU · CUDA · Floyd-Warshall.

1 Introduction

The introduction should set the context for your project. Why is this topic relevant?

You should also define the scope of your project. You could design a software artifact that would end poverty and famine, but that is not realistic.

For example, this document describes the structure your paper should have. Despite using the LNCS LaTeX template [1], the formatting template is not relevant, only the content structure is relevant.

Finally, you should define the goals of your project. For instance,

- To propose a method for the parallelization of Genetic Algorithms
- An implementation of such algorithm
- The experimental evaluation of such method, with comparison with a sequential alternative.

2 Background

This is an optional section, as it depends on your project. In projects where a given specific knowledge is required to understand the article, you should give a brief introduction of the required concepts. In the case of genetic algorithms, you should present the basic algorithm in this section.

[1] LNCS is the official template for Europar 2019, in case you are interested.

3 Approach

In this section, you should present your approach. Notice that an approach may be different than an implementation. An approach should be generic and ideally applied for different machines, virtual machines or languages. You should present algorithms or generic diagrams explaining the approach.

4 Implementation Details

In this section you can talk about details of how you implemented your approach. Depending on the detail of your approach, this might be an optional section.

As an example, I would detail how I implemented the phaser in the genetic algorithm, or how I implemented parallel mergesort on ForkJoin. Another aspect could be how to organize your arrays to minimize overhead in recursive calls.

5 Evaluation

5.1 Experimental Setup

In this section you should describe the machine(s) in which you are going to evaluate your system. Select the information that is relevant.

5.2 Results

In this section you should present the results. Do not forget to explain where the data came from.

You should include (ideally vectorial) plots, with a descriptive caption. Make sure all the plots (Like Figure 1 are well identified and axis and metrics are defined.

5.3 Discussion

Here you should discuss the results on a high level. For instance, based on our results, the parallelization of the merge-sort is relevant as no other parallel work occurs at the same time, and the complexity $O(Nlog(N))$ can have a large impact when the number of individuals is high.

6 Related Work

This section can be either the second one, or the second-to-last. In the case where knowledge of other competing works is required, it could come before. But if you are confident on what you did, it should appear at the end, where you can compare existing works against yours. An example is below:

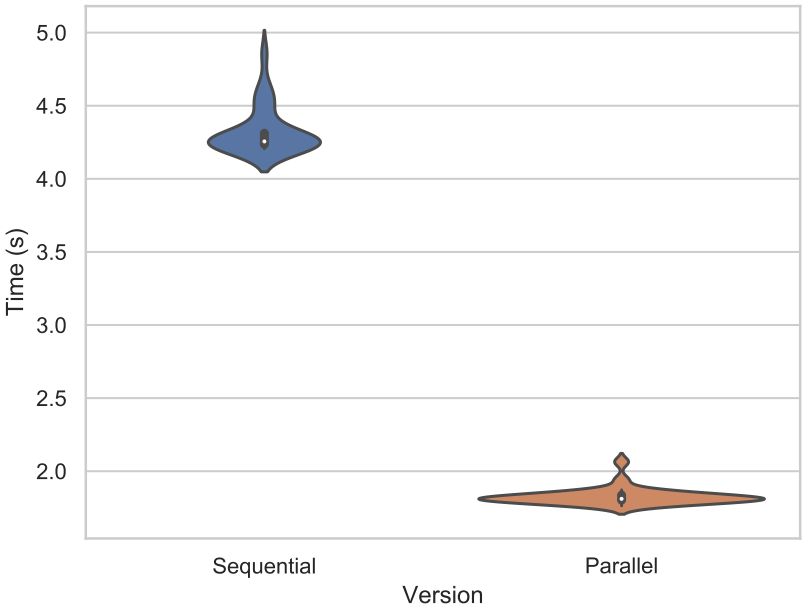

Fig. 1. Comparison of the performance of sequential and parallel versions of the algorithm.

Chu and Beasley proposed a Genetic Algorithm for the Multidimensional Knapsack Problem [1]. This work introduces a heuristic as a repair operator. Our work makes no optimization with regards to the problem domain, making it more generic and supporting other problems.

When using BibTeX references, I suggest using DBLP[2], leaving Google Scholar as a backup, since DBLP has more correct and detailed information about research papers.

7 Conclusions

Here you should resume the major conclusions taken from discussion. Ideally, these should align with the objectives introduced in the introduction.

You should also list the future work, i. e., tasks and challenges that were outside your scope, but are relevant.

Acknowledgements

First Author wrote the part of the program implemented the phasers. Second Author implemented the MergeSort in parallel.

Both authors wrote this paper, with First Author focusing on the introduction, related work and conclusions while the Second Author focused on approach and evaluation.

Each author spent around 30 hours on this project.

References

1. Chu, P.C., Beasley, J.E.: A genetic algorithm for the multidimensional knapsack problem. J. Heuristics **4**(1), 63–86 (1998). https://doi.org/10.1023/A:1009642405419, https://doi.org/10.1023/A:1009642405419

[2] https://dblp.org