PSO-GPU: Accelerating Particle Swarm Optimization in CUDA-Based Graphics Processing Units

Daniel Leal Souza[1,2,3]
daniel.leal.souza@gmail.com

Tiago Carvalho Martins[3]
tiagocm@ufpa.br

Victor Alexandrovich Dmitriev[3]
victor@ufpa.br

Glauber Duarte Monteiro[1,2]
glauberbcc@gmail.com

Otávio Noura Teixeira[1]
onoura@gmail.com

ABSTRACT

This work presents a PSO implementation in CUDA architecture, aiming to speed up the algorithm on problems which has large amounts of data. PSO-GPU algorithm was designed to customization, in order to adapt for any problem that can be solved by a PSO algorithm. By implementing PSO using CUDA architecture, each processing core of the GPU will be responsible for a portion of the overall processing operation, where each one of these pieces are handled and executed in a massive parallel enviroment, opening the possibility for solving problems that require a large processing load in considerably less time. In order to evaluate the performance of PSO-GPU algorithm two functions were used, both global optimization problems, where without constraints (Griewank function) and other considering constraints, the Welded Beam Design (WBD).

Categories and Subject Descriptors

I.2.8 [**Computing Methodologies**]: Artificial Intelligence – *Heuristic Methods.*

D.1.3 [**Software**]: Concurrent Programming – *Parallel Programming.*

General Terms

Algorithms, Performance, Parallelism, CUDA

Keywords

CUDA, Parallel Computing, Particle Swarm Optimization, Late-Breaking Abstracts

1. INTRODUCTION

Particle Swarm Optimization is a stochastic optimization technique initially designed for nonlinear and continuous functions. It was developed by James Kennedy and Russel Eberhart in 1995 [1] and inspired by Frank Heppner´s researches about social behavior in some species of birds [2]. And according to the mechanics of PSO, it is inspired by collaborative behavior and swarming of biological populations like bee swarms, bird flocks and fish schools.

One of the main difficulties in the PSO algorithm lies in the fact that, depending on the complexity of the problem, the number of particles and/or iterations shall be larger enought to increase the probability of obtaining a good result, thereby increasing the execution time of the algorithm, which is a problem in certain cincunstances (e.g., real-time applications).

GECCO'11, July 12–16, 2011, Dublin, Ireland.
ACM 978-1-4503-0690-4/11/07.

This work presents PSO-GPU, a generic and customizable implementation of a PSO algorithm under the CUDA architecture, that takes advantage from thousands of *threads* present in the GPU, by reducing the runtime and increasing performance using parallel processing.

In order to validate the results, tasks envolving Griewank function (unconstrained) and WBD function (constrained) were executed in PSO-GPU for speedup and convergence tests. The convergence results obtained in WBD function were compared with another results, found in [5], [6], [7] and [8]. The results shows that this approach has a great appeal in the way to solve more quickly any kind of optimization problems.

2. PSO-GPU: PARTICLE SWARM OPTIMIZATION UNDER CUDA ARCHITECTURE

PSO is a metaheuristic with high capacity for parallelization. The only operation that could not be performed concurrently is the operation of finding the best global best.

PSO-GPU was designed to work with parallelism 1 to 1 (one *thread*, one particle), where each element of the particle is treated individually in the thread, allowing an efficient data parallelism, without risk of starvation or race conditions. Figure 1 shows the PSO flow task in CPU and Figure 2 shows the PSO flow task in GPU.

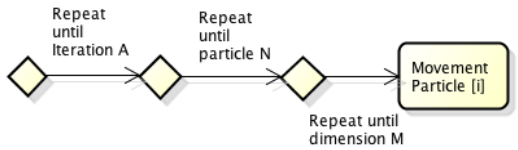

Figure 1. Activity diagram of moving particle process in CPU

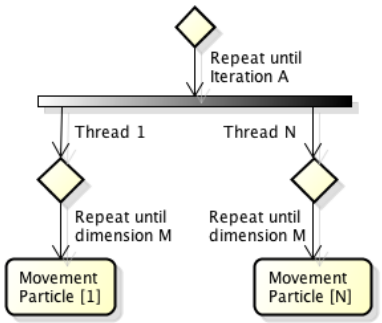

Figure 2. Activity diagram of moving particle process in GPU

3. SIMULATION AND RESULTS

The experiments were performed on a machine using NVIDIA GeForce GT 330M with 256 MB of VRAM memory. Two experiments were made, both of them, with different configurations:

Experiment One (E1):

- **Number of Particles**: 512;
- **Number of Tests Performed Per Function**: 30;
- **Number of Iterations**: 10,000.

Experiment Two (E2):

- **Number of Particles**: 1,024;
- **Number of Tests Performed Per Function**: 30;
- **Number of Iterations**: 100,000.

All these simulations were made based on Griewank function and WBD function with 30 and 4 dimensions, respectively.

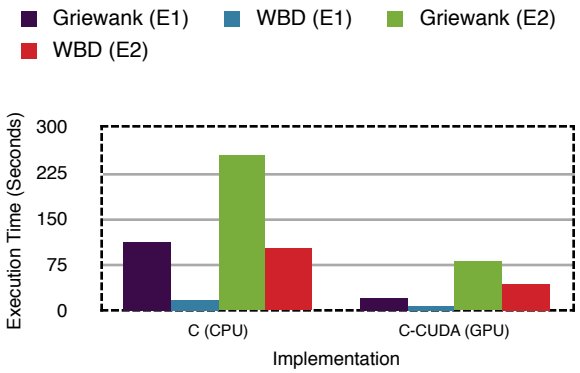

Figure 3. Computing time between functions in Experiment One

Table 1: Computing Time

Function	Station	Time (Seconds)	SpeedUp
Griewank (E1)	C (CPU) CUDA-C (GPU)	113.514 22.602	5.022x
WBD (E1)	C (CPU) CUDA-C (GPU)	19.381 8.872	2.184x
Griewank (E2)	C (CPU) CUDA-C (GPU)	254.431 81.92	3.105x
WBD (E2)	C (CPU) CUDA-C (GPU)	102.927 45.449	2.264x
Total			3.143x

Table 2: Convergence Results and Comparission With Other Results

	PSO-GPU	[4]	[5]	[6]	[7]
Griewank	0	N/A	N/A	N/A	N/A
WBD	1.45888	1.72802	1.72822	1.74830	2.43311

4. BRIEF DISCUSSION AND CONCLUSION

This paper has presented an implementation of the PSO in the CUDA architecture. With the results obtained through experiments, we can conclude that the exploration of the CUDA benefits applied to bioinspired metaheuristics, where the purpose is parallelize massively tasks which demand a very high processing time brings great benefits, not only for being faster, but to have a implementation which is portable, easy to read and adapt to other problems. Another goal is the WBD results, that is better than the other results.

As future improvements, we can mention the use of shared memory in each thread block, in order to increase speed in data access. Another implementation in progress, is the use of GPU to perform multiple PSOs with same or different configurations, in a cooperative system of multi-PSOs.

5. REFERENCES

[1] Kennedy, J. and Eberhart, R. Particle Swarm Optimization. *Proceedings of the IEEE International Conference on Neural Networks,* (Perth, Australia), (1995), 1942-1948. IEEE Press.

[2] Heppner, F. and Grenader, U. *A Stochastic Nonlinear Model For Coordinated Bird Flocks: The Ubiquity of Chaos*. AAAS Publications, Washington, DC, 1990.

[3] Kennedy, J. and Eberhart, R. A Discrete Binary Version of The Particle Swarm Algorithm. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics,* (Piscataway, New Jersey, USA), (1997), 4104-4108. IEEE Press.

[4] He, Q., Wang, L. An Effective Co-evolutionary Particle Swarm Optimization for Constrained Engineering Design Problem. Engineering Aplications of Artificial Intellifence, (2007), 89-99, Elsevier Press.

[5] Coello, C. A. C., Montes, E. M. Constraint-handling in Genetic Algorithms through the use of Dominance-based Tournament Selection. *Advanced Engineering Informatics* 16, (2002), 193-203.

[6] Coello, C. A. C. Use of a Self-adaptive Penalty Approach for Engineering Optimization Problemas. *Computers in Industry* 41, (2000), 113-127.

[7] Deb, K. GeneAS: a robust optimal design technique for mechanical component design. *Dasgupta, D., Michalewicz, Z (Eds.), Evolutionary Algorithms in Engineering Applications.* (Springer, Berlin), (1997), 497-514.

[1]Laboratório de Computação Natural (LCN), Área de Ciências Exatas e Tecnologia (ACET), Centro Universitário do Estado do Pará (CESUPA), Av. Governador José Malcher, 1963 – São Brás – 66.060-230, Belém – Pará – Brasil. 55-91-4009-9145. URL: http://www.lcn-cesupa.org.

[2]Programa de Pós-Graduação em Ciência da Computação (PPGCC), Instituto de Ciências Exatas e Naturais (ICEN), Universidade Federal do Pará (UFPA), CEP – 66.075-110, Belém – Pará – Brasil. 55-91-3201-7103. URL: http://www.ufpa.br/ppgcc/ppgcc/.

[3]Laboratório de Nanofotônica e Nanoeletrônica, Instituto de Tecnologia (ITEC), Universidade Federal do Pará (UFPA), Caixa Postal 8619 – 66.075-900, Belém – Pará – Brasil.