Optimization Techniques on GPU: A Survey

Rashmi Sharan Sinha
SBS State Technical Campus,
Ferozepur, Punjab [INDIA]
Email:sinharashmisinha@hotmail.com

Satvir Singh
SBS State Technical Campus,
Ferozepur, Punjab [INDIA]
Email: satvir15@gmail.com

Abstract*—In this paper, we present a comprehensive survey on parallelizing computations involved in optimization problem, on GPU using CUDA. Many researchers have reported significant speedup using CUDA on GPU. Stochastic algorithms, Metaheuristic algorithms and Heuristic algorithms i.e., Mixed Integer Non-linear Programming (MINLP), Central Force Optimization (CFO), Genetic Algorithms (GA), Particle Swarm Optimization (PSO), etc. are exploring/exploiting the processing power of GPU. GPGPU shows tremendous speedups of 6x to 7x in Steady State Genetic Algorithm to 10,000x speedups in CFO. GPU have multithread cores with high memory bandwidth which allow for greater ease of use and also more radially support a layer body of applications.

***Index Terms*—GPU, GPGPU, CUDA, MINLP, PGMOEA, CGA, CFO, Optimization Algorithms.**

I. INTRODUCTION

General Purpose GPU Computing really took off when CUDA and Stream arrived in late 2006 [1]. GPU constitute a tremendous step towards a usable, suitable, scalable and manageable future-proof programming model [2]. Optimization workloads are very parallel, and so GPUs developed as large-scale parallel computation machines [3] [4] [5] [6]. Originally GPGPU processing was done by tricking the GPU by disguising computation loads as graphic loads [7]. With the advent and large availability of General Purpose Graphics Processing Units and the development and straightforward applicability of the Compute Unified Device Architecture platform, several applications are being benefited by the reduction of the computing time [8]. GPGPU-based architecture, aiming at improving the performance of computationally demanding optimizations for identifiable specific mapping parameters, one can reduce total execution time drastically and also, improve greatly the optimization process convergence. Application performance can be significantly improved by applying memory-access pattern-aware optimizations that can exploit knowledge of the characteristics of each access pattern [3]. To evaluate the effectiveness of our methodology, we have created a tool that incorporates our proposed algorithmic optimizations and report on execution speedup using selected benchmark kernels that cover a wide range of memory access patterns commonly found in GPGPU workloads [9]. Graphics Processing Units (GPUs) are widely used among developers and researchers as accelerators for applications outside the domain of traditional computer graphics. In particular, GPUs have become a viable parallel accelerator for scientific computing with low investment in the necessary hardware.

II. MINLP OPTIMIZATION

With the increasing advent of GPGPU using CUDA, the stochastic algorithm of advanced Genetic Algorithm is used to solve non-convex mixed integer Non-linear Programming (MINLP) and non-convex Non-linear Programming (NLP) problems [10]. MINLP refers to mathematical programming algorithms that can optimize both continuous and integer variables, in a context of nonlinearities in the objective function and/or constraints. MINLP problems involve the simultaneous optimization of discrete and continuous variables. These problems often arise where one is trying to simultaneously optimize the system structure and parameters. This is difficult because optimal topology is dependent upon parameter levels and vice versa [10]. In many design optimization problems, the structural topology influences the optimal parameter settings so a simple de-coupling approach does not work: it is often not possible to isolate these and optimize each separately. Finally, the complexity of these problems depends upon the form of the objective function. In the past, solution techniques typically depended upon objective functions that were single-attribute and linear (i.e., minimize cost). However, real problems often require multi-attribute objectives such as minimizing costs while maximizing safety and/or reliability, ensuring feasibility, and meeting scheduled deadlines. In these cases, the goal is to optimize over a set of performance indices which may be combined in a nonlinear objective function. Through this algorithm the intensity of each individual is beamed using entropy measures. The results of the tests shows a significant speedup of 42x with single precision and 20x with double precision over nVidia Fermi C2050 GPU [10].

III. PGMOEA

The general Purpose GPU is efficiently used in optimizing the multiple objective problems. The particle gradient Multiobjective Evolutionary Algorithm (PGMOEA) is used to solve optimization problems. PGMOEA is first experimented on CPU and then after parallelizing the algorithm executed upon GPU which formed a great speedup results [11]. The experiment is conducted upon two different examples. The first example shows a speedup of 9x with nVidia GeForce GTX285 then CPU result, while the second example is 10x faster than that of CPU [11]. The speedup comparison is shown below in Table 1.

Table 1. Speedup Comparison (source [11])

Algorithm	Example 1		Example 2	
	Time(s)	Speedup	Time(s)	Speedup
PGMOEA on GPU	0.97	9.95	0.83	10.64
PGMOEA on CPU	9.01	1.04	8.02	1.10

IV. CELLULAR GENETIC ALGORITHM

Genetic Algorithm have a subclass known as Cellular Genetic Algorithm (cGA) which provides the data of population structured in several specified topologies [12]. The cGA is compared upon CPU, single GPU and multiGPU. The nVidia GTX285 multiGPU test shows a speedup of 8 to 771 times then single GPU [12]. The multiGPU is more prominent in paralleling the algorithm and producing accurate results as there is a need of special maintenance to perform same experiment upon single GPU.

V. DIFFERENTIAL EVOLUTIONARY ALGORITHM

GPGPU is proved to be great architectural unit in reducing the processing time [13]. The Differential Algorithm which is one of the part of Evolutionary Algorithm is implemented upon CPU using C-CUDA. The motivating features of Differential Algorithm are easy for parallelization and convergence properties which intern gives an appropriate result. The algorithm is first tested upon CPU then on nVidia GTX285 with 1GB GDDR3 GPU with the speedup outcomes. GPU gives 20x to 35x faster results which proves GPU is much more effective and efficient than Differential Algorithm on CPU [13]. The Speedup comparison results are shown in Table 1.

VI. CELLULAR AUTOMATA

Cellular Automata have various real life application like physical system modeling, road traffic simulation, artificial life simulation, etc [14], [15], [16]. Cellular automata design evolved from evolutionary algorithm and a part of Genetic Algorithm which is complex in nature. The Algorithm is parallelized and implemented upon GPGPU shows an efficient reduction in execution time. The rules of Cellular Automata take longer time period inevolution in sequential execution. The same Genetic Algorithm shows 31.34x to 314.94x speedup when executed upon nVidia GeForce FX280 GPU which is a significant reduction in execution time [17].

VII. ACCELERATING PSO

PSO is a metaheuristic algorithm works by having a swarm of particles [18]. These particles are moved around in the search-space according to a few simple formulae. The movements of the particles are guided by their own best known position in the search-space as well as the entire swarm’s best known position [19]. When improved positions are being discovered these will then come to guide the movements of the swarm. Particle Swarm Optimization (PSO) is one of the type of Evolutionary Algorithm used to optimize the multiple objective problems. When an optimization problem involves more than one objective function, the task of finding one or more optimal solutions is known as multi-objective optimization [18]. The objects are having random velocities and positions. The algorithm is tested upon three different platforms of C, Matlab and C-CUDA. The parallel implementation of PSO on nVidia GTX 280 gives 17 to 41 times speedup in computing time in C-CUDA as compared with the C and Matlab as Shown in Fig.1 [20].

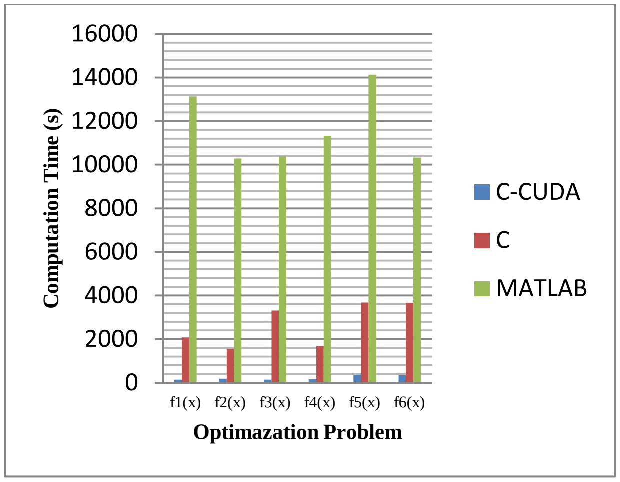

Figure 1. Computing time for C-CUDA, C AND MATLAB (Source [20])

VIII. STEADY STATE GENETIC ALGORITHM ON GPU

The optimization problem is effectively solved by the means of Evolutionary Computing [21]. The steady state Genetic Algorithm used to access optimization algorithms. These algorithms basically have selection for the reproduction and selection of survival implementation with concurrent kernel execution [22]. The study is first performed upon CPU then with nVidia GeForce GTX480 GPU gives a speedup of 3x to 6x then the previous implementation on CPU [12]. The executed time is greatly reduced using general purpose GPU. The population individual data is accessed parallelaly which effectively speedup the process.

IX. BINARY-CODED AND REAL-CODED GENETIC ALGORITHM

Genetic Algorithm is tested and evaluated on parallel implementation on C-CUDA API on the parameters like population size, number of threads, problem size and problem of differing complexities with variation in the population individuals [12]. For an efficient implementation on GPGPU the solution is thoroughly implemented along with the operators like random number generation, initialization, selection operation, and mutation operations [13]. The nVidia GeForce 8800GTX shows overall speedup of 40-400 on three different test problems [23]. Thus parallel implementation is more effective then sequential process as compared with clock time and accuracy.

X. CENTRAL FORCE OPTIMIZATION (CFO)

The metaheurestic algorithm Central Force Optimization (CFO) is implemented upon GPGPU using local neighborhood and implemented CFO concepts [24]. The calculation of CFO is dependent upon the movement of probes which are scattered all over the space. The probes then slowly move towards the probe having highest mass or fitness. PR-CFO is the most evaluated algorithm with the measures of initial position and acceleration vectors, fitness evaluation and probe movements [25]. The test problems are having the dimension of 30 to 100 of four different examples of Pseudo random CFO (PR-CFO). The PR-CFO is tested with four test types i.e. Ring, Standard, CUDA, CUDA Ring. PR-CFO shows a speedup of 4 to 400 using CUDA. PR-CFO ring and PR-CFO CUDA ring on nVidia Tesla C1060 shows 10,000 times faster results as compared with standard PR-CFO algorithm [25].

XI. CONCLUSION

In this paper we present different optimization algorithm with tremendous speedup in the computation time. MINLP archived an overall speedup of 20x to 42x using nVidia Tesla C2050 GPU as compared to intel Core i7 920 CPU processor. The new binary-coded and real-coded Genetic Algorithm using CUDA leads to a performance improvement with the speedup of 40x to 400x. Central Force Optimization (CFO) results in reduction of computing time and a speedup of 10,000x. The Cellular Automata shows 314.97x as compared with the sequential implements.

XII. REFERENCES

[1] K. S. Perumalla, “Discrete-event execution alternatives on general purpose graphical processing units (GPGPUs),” in Principles of Advanced and Distributed Simulation, 2006. PADS 2006. 20th Workshop on, 2006, pp. 74–81. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1630711
[2] J. A. Jablin, P. McCormick, and M. Herlihy, “Scout: High-performance heterogeneous computing made simple,” in Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on, 2011, pp. 2093–2096.
[3] Fast Parallel Markov Clustering in Bioinformatics Using Massively Parallel Graphics Processing Unit Computing, 2010.
[4] Data Processing in Space Weather Physics Models in the Meridian Project, 2010.
[5] Design and Implementation of Remote Parallel Computing System Based on Multi-Platform, 2010.
[6] An efficient parallel algorithm for evaluating join queries on heterogeneous distributed systems, 2009.
[7] Profiling General Purpose GPU Applications, 2009.
[8] D. Luebke, M. Harris, N. Govindaraju, A. Lefohn, M. Houston, J. Owens, M. Segal, M. Papakipos, and I. Buck, “Gpgpu: generalpurpose computation on graphics hardware,” in Proceedings of the 2006 ACM/IEEE conference on Supercomputing. ACM, 2006, p. 208.
[9] Evolving a CUDA kernel from an nVidia template, 2010.
[10] Advanced genetic algorithm to solve MINLP problems over GPU, 2011.
[11] Particle Gradient Multi-objective Evolutionary Algorithm Based on GPU with CUDA, 2010.
[12] A multi-GPU implementation of a Cellular Genetic Algorithm, 2010.
[13] Differential evolution algorithm on the GPU with C-CUDA, 2010.

[14] L. J. Durbeck and N. J. Macias, “The cell matrix: an architecture for nanocomputing,” Nanotechnology, vol. 12, no. 3, p. 217, 2001.

[15] M. Gardner, “Mathematical games: The fantastic combinations of john conways new solitaire game life,” Scientific American, vol. 223, no. 4, pp. 120–123, 1970.

[16] M. Tomassini, M. Sipper, and M. Perrenoud, “On the generation of high-quality random numbers by two-dimensional cellular automata,” Computers, IEEE Transactions on, vol. 49, no. 10, pp. 1146–1151, 2000.

[17] GPU Accelerators for Evolvable Cellular Automata, 2009.

[18] J. Kennedy, R. C. Eberhart, and Y. Shi, “Swarm intelligence. 2001,” Kaufmann, San Francisco, vol. 1, pp. 700–720, 2001.

[19] J. Kennedy and R. Mendes, “Population structure and particle swarm performance,” in Evolutionary Computation, 2002. CEC’02. Proceedings of the 2002 Congress on, vol. 2. IEEE, 2002, pp. 1671–1676.

[20] Swarm’s flight: Accelerating the particles using C-CUDA, 2009.

[21] F. Stentiford, “An evolutionary programming approach to the simulation of visual attention,” in Evolutionary Computation, 2001. Proceedings of the 2001 Congress on, vol. 2, 2001, pp. 851–858. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=934279

[22] R. Arora, R. Tulshyan, and K. Deb, “Parallelization of binary and real-coded genetic algorithms on GPU using CUDA,” in Evolutionary Computation (CEC), 2010 IEEE Congress on, 2010, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5586260

[23] Parallelization of binary and real-coded genetic algorithms on GPU using CUDA, 2010.

[24] A. Stefek, “Benchmarking of heuristic optimization methods,” in MECHATRONIKA, 2011 14th International Symposium. IEEE, 2011, pp. 68–71.

[25] Central Force Optimization on a GPU: A case study in high performance metaheuristics using multiple topologies, 2011.