Parallel particle swarm optimization using GPGPU

Ing. Iliana Castro Liera, Dr. Marco Antonio Castro Liera, M.C. Jesús Antonio Castro

***Abstract*—This work presents a parallelization method for the Particle Swarm Optimization algorithm using a low-cost architecture: a General Purpose Graphics Processing Unit (GPGPU). The strategies to better suit the architecture main characteristics are addressed along success rates and convergence times for the optimization of Rastrigin's and Ackley's functions on a 30-dimensional search space, and compared with results previously obtained using a cluster implementation.**

Index Terms—CUDA (Compute Unified Device Architecture), Evolutionary computation, Parallel architectures, Particle swarm optimization.

I. Introducción

The present work deals with a parallel implementation of a population-based optimization algorithm known as Particle Swarm Optimization (PSO).

Nowadays, parallel applications development is not confined to big and expensive equipments anymore. In recent years practically all new models of personal computers are built around multi-core processors. In some cases, those PCs are equipped with advanced graphic processors with the ability of performing general purpose computation (GPGPU). [1][2].

Particle swarm optimization is a heuristic, population-based local search algorithm first proposed by Russ C. Eberhart and James Kennedy in 1995 [3].

PSO is based on the behavior of insect swarms and bird flocks when searching for food or escaping from predators. Some of the attractive features of PSO include the ease of implementation and the fact that no gradient information is required [3][4].

Each particle in the swarm has memory of its current position, speed and fitness, and his best historical position and best historical fitness.

The algorithm starts with a randomly generated set of particles called a swarm, and then each particle calculates his speed using (1) and his new position using (2) for each dimension d through a previously chosen number of generations.

$$v_{i,d}(t+1) = w v_{i,d}(t) + c_1 r_1 (x_{hd} - x_{id}) + c_2 r_2 (x_{gd} - x_{id}) \quad (1)$$

Where x_h represents each particle's historical best position, w an inertial weight, c_1 the confidence on self information and c_2 it's confidence on social information. r_1,r_2 are randomly generated numbers on the interval [0,1].

$$x_{i,d}(t+1) = x_{i,d}(t) + v_{i,d}(t+1) \quad (2)$$

If global (gbest) strategy is employed, x_g represents the best position of the entire swarm, whereas for local strategy (lbest) x_g represents the neighborhood's best position. In general, gbest strategy tends to converge faster, but is more likely to become trapped on local optimum regions than lbest. Additionally gbest can be viewed as a special case of lbest where the neighborhood size is the entire swarm [4].

II. GPGPU Implementation

A. Compute Unified Device Architecture (CUDA)

CUDA is a GPGPU technology developed by NVIDIA in 2006 with the launch of the GeForce GTX 8800 card, few months later, the CUDA C compiler was released to the public enabling programs to perform parallel general calculations over advanced graphics processing cards. CUDA architecture is represented on Fig. 1.

The elements of the CUDA architecture are: *Threads*, that execute functions designed to run simultaneously and have their own memory space, *Blocks* that are formed by a set of threads and include a shared memory area for their threads communication, *Grid*, which includes all the blocks executed on a single *Device* and includes memory areas common to all blocks. (Fig. 1)

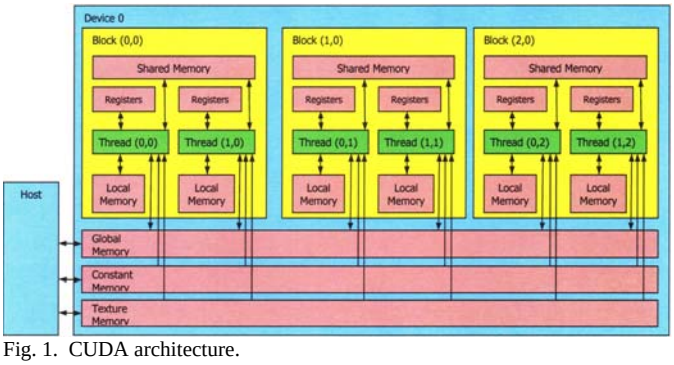

Fig. 1. CUDA architecture.

A function running in the GPU is called a *kernel.* When a kernel is launched, is necessary to inform the GPU how many blocks (BLK), and threads (THR), will be running on the GPU using the following syntax:

kernel_name<<<BLK,THR>>>(parameters);

The GPU organizes work on its multiprocessors as shown

on fig. 2. It is important to use a multiple of the available amount of multiprocessors as the BLK parameter to avoid unused resources.

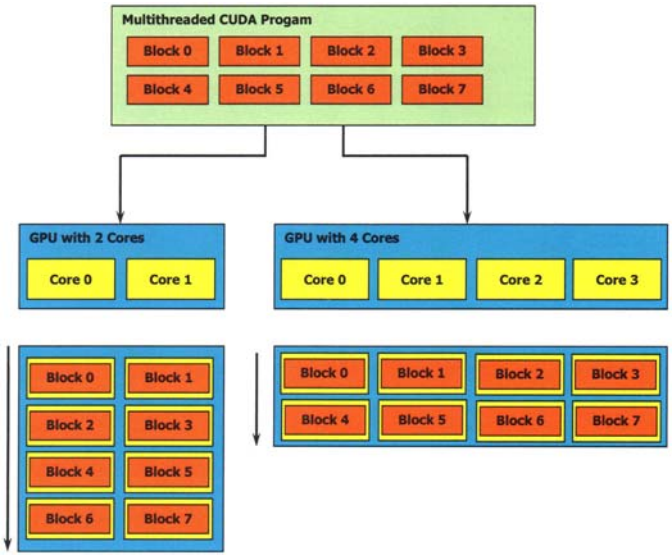

Fig.2. More multiprocessors execute in less time same amount of blocks

To better suit this particular architecture, a smaller swarm size is assigned on each block launched on the GPU. A thread is responsible to process a single particle of a swarm as depicted on Fig 3.

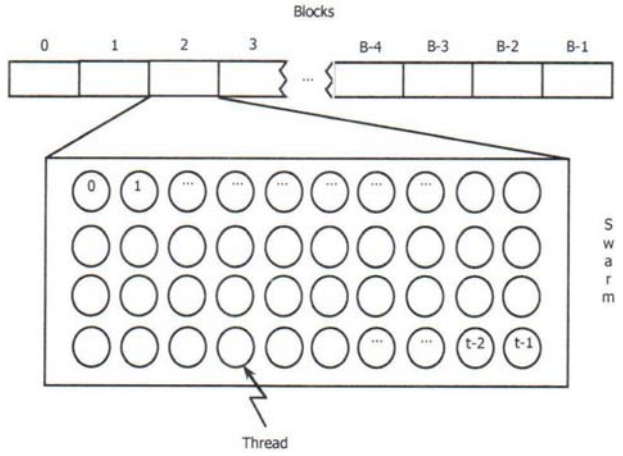

Fig. 3. A t particles swarm was processed on a block. Single particle was assigned to a thread.

A migration process occurs every PMB generations, where the best position of each swarm is informed with a ring topology on block launched as seen on Fig. 4.

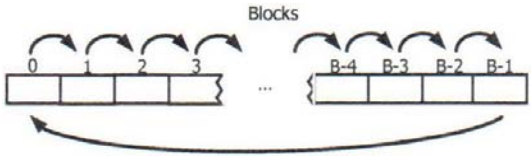

Fig. 4. Best position is informed to next block every PMB generations.

The following pseudo-code illustrates the task assignation and shows which parts are executed on to CPU and GPU respectively:

```
CPU:
Launch kernel to Create BLK swarms
for i = 1 to GMAX/PMB
    Launch kernel Pso
    Launch kernel Migration
receive the best position from BLK swarms
Display the best overall position as the
solution of the optimization problem

GPU:
G = 0;
KERNEL Create
for i=1 to P
  initialize Xi, Vi and Fi

KERNEL Pso
while (G<PMB)
  G = G+1
  for i=1 to Swarm_size
    calculate Xi fitness F(Xi)
    if F(Xi) is better than F(Xhi) then
        Xhi = Xi
  for i = 1 to Swarm_size
    choose gbest form swarm
    update Xi speed Vi
    update Xi position

KERNEL Migration
Inform best position to (BlockId+1) mod BLK
```

B. *Test Functions*

Rastrigin's and Ackley's functions, two of the Competence of Evolutionary Computation (CEC'05) problems [4], where used to test the proposed algorithm.

Rastrigin's function (3) is a scalable, separable, multi-modal problem, it has a known global optimum on the origin and a huge number of local optima's, the defined search space was $[-5,5]^D$ in accordance to the CEC'05

$$f(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\Pi\ x_i) + 10) \quad (3)$$

Ackley's function (4) is a multi-modal non-separable and scalable problem, it has a known global optimum on the origin and very small decreasing area around the optimum, the defined search space was $[-32,32]^D$ in accordance to the CEC'05, where D represents the number of dimensions of the search space.

$$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\Pi\ x_i)\right) + 20 + e \quad (4)$$

C. *Parameter Determination*

Several works [5]-[7] have dealt with the PSO parameter determination problem, confidences and inertial weights of c_1=c_2=1.62, w=0.8 are recommended and produced good results for all our tests.

Based on results obtained by Castro et. al.[8], a migration period of 1% of the total generations number was used.

III. RESULTS

A 30-dimension Rastrigin's function (3) was minimized using 21 swarms with a swarm size of 128 particles over 68,000 generations Table I shows that 100% success rate results were obtained for all admissible error limits using the

680 generations migration period (1% of the total generations number).

TABLE I
SUCCESS RATE PERCENTAGES OF RASTRIGIN'S

Generations	Success Rate %(1e-6)	Execution Time (seg)
48000	99.40	3.00
55000	99.67	4.00
60000	99.80	4.00
65000	99.93	5.00
68000	100.00	5.00

Ackley's function, (4) was optimized for a 30-dimensional search space, using 70 swarms with a swarm size of 8 particles over 30,000 generations. Table II shows that 100% success rate was obtained using the 300 generations migration period (1% of the total generations number) which is consistent with the previous case.

TABLE II
SUCCESS RATE PERCENTAGES OF ACKLEY'S

Generations	Success Rate %(1e-6)	Execution Time (s)
20000	98.93	1.00
25000	99.80	2.00
30000	100.00	3.00

Tests were conducted using version 3.2 of CUDA C Libraries on Visual Studio 2008 API, using Windows 7 32 bits on computer equipped with Intel Core i5 650 @ 3.20GHz, 4 GB RAM, 1333 MHz, and GPU NVIDIA GeForce GTX 460 1GB total memory, 7 multiprocessors and 336 CUDA cores.

The best result reported by Castro et. al. [8] on a cluster using a fast-ethernet switch with 16 2.8GHz Intel E7400 2 cores processor computers, using Fedora 14 and PVM 3.4.5 with gcc 4.5.1.4 for Rastrigin Function using 32 swarms of 100 particles over 25,000 generations, migration period of 250 and lbest strategy with neighborhood size of 20%, was: Success rate percentage (1e-6), 100; Execution Time, 8.2s. And for Ackley's function using 32 swarms of 100 particles over 6,000 generations, migration period of 60 generations and lbest strategy with neighborhood size of 20%, was: Success rate percentage (1e-6), 100; Execution Time, 2.0s with 16 computers, 3.95s with 8 computers.

IV. CONCLUTIONS AND FUTURE WORK

The proposed implementation is a very good option for parallel processing of PSO, considering it is not as expensive as other alternatives (clusters or supercomputers) and for the Rastrigin problem it's faster than a 16 computers cluster and for the Ackley problem it's faster than a 8 computer cluster according to the results obtained by Castro et. al.[8].

Currently, work is being done on the implementation of Genetic Algorithms on this architecture and comparing its efficiency against PSO for different problems.

This year, a new competence for real-world numeric optimization problems is been held and the proposed algorithms can be modified to solve the proposed problems [9].

REFERENCES

[1] Nvidia Corp, "CUDA Architecture Introduction & Overview", Nvidia Corp, Santa Clara, California Version 1.1, april 2009, 3pp.

[2] Sanders, J. and kandrot, E. "Cuda by Example. An Introduction to General-Purpose GPU Programming", 1 ed. 2010, Ann Harbor, Michigan, Pearson, 290 pp.

[3] Kennedy, J. and Eberhart, R. (1995), "Particle Swarm Optimization", Proc. 1995 IEEE Intl. Conf. on Neural Networks, pp. 1942-1948, IEEE Press.

[4] Sugatan, P.N. et al "Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization", KanGAL, Tech. Rep 2005005, Nanyang Technological University, Singapore., May 2005.

[5] Van de Berg, F. "An Analysis of Particle Swarm Optimizers", PhD Dissertation Faculty of Agricultural and Natural Science, Pretoria University, Pretoria South Africa, November 2001.

[6] Clerc, M. "Particle Swarm Optimization", 1st English Ed. Newport Beach ISTE, 2006

[7] Mussi, L Daolio, F and Cagnoni, S., "GPU-based Road Sign Detection Using Particle Swarm Optimization", 9th International Conference on Intelligent Systems Design and Applications, 2009

[8] Castro, M.A. Morales, J.A. Castro, I. Castro, J.A. and Cárdenas L.A., "Distributed Particle Swarm Optimization Using Clusters and GPGPU", send for publication on I Congreso Internacional de Ingeniería Electrónica y Computación 2011. Minatitlán, Veracruz, México.

[9] Swagatam, D, and Sugatan, P.N "Problem Definitions and Evaluation Criteria for the CEC 2011 Competition on Testing Evolutionary Algorithms on Real World Optimization Problems", Dept. of Electrical Communications Engg, Jadavpur University, Kolkata 700 032, India and School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore., December 2010.

Iliana Castro Liera Is a full time titular professor at the Technological Institute of La Paz, she obtained his engineering degree on Computer Systems from the Technological Institute of La Paz, and is currently on a sabbatical year working on a masters degree thesis regarding implementation of parallel population-based optimizations algorithms using GPGPU. iliana.castro@gmail.com

Marco Antonio Castro Liera Obtained his Ph.D from the Central University "Marta Abreu" of Las Villas, in Santa Clara, Cuba.

His employment experience includes the Transport and Communications Secretary, Baja California Sur Autonomous University and currently holds a titular position at the Postgraduate Studies and Research Division of the La Paz Technological Institute in La Paz, México. Professor Castro has published his work on Genetic Algorithms, Distributed Computing, and Systems Identification on several international events and arbitrated and indexed periodicals. mcastroliera@gmail.com

Jesús Antonio Castro Obtained his masters degree from UNAM, and currently holds a titular professor position at the Postgraduate Studies and Research Division of the La Paz Technological Institute where he teaches Distributed Processing and Systems Software. castroli48@gmail.com.