GECCO 2012 GPGPU competition
The EASEA parallelization platform

Pierre Collet
Strasbourg University, ICUBE, BFO
Illkirch, France
pierre.collet@unistra.fr

Ogier Maitre
Strasbourg University, ICUBE, BFO
Illkirch, France
ogier.maitre@unistra.fr

Frédéric Krüger
Strasbourg University, ICUBE, BFO
Illkirch, France
frederic.kruger@unistra.fr

ABSTRACT

1. INTRODUCTION

This entry is not for a standard application of evolutionary computation that can maximally exploit the parallelism provided by low-cost consumer graphical card, but for a platform that automatically parallelizes an evolutionary algorithm onto one or several NVIDIA GPU cards and over one or several potentially heterogeneous computers.

The EASEA (for EAsy Specification of Evolutionary Algorithms) parallelizing platform [2] can take `.ez` specification files for Genetic Algorithms, Evolution Strategies, Memetic Algorithms, Genetic Programming and CMA-ES evolutionary engines. In the current version available on the web site (`http://easea.unistra.fr`), it only parallelizes the evaluation stage over one or several GPU cards.

In case of a standard evolutionary algorithm, the evaluation function is compiled thanks to `nvcc` and individuals are sent on the GPU card to evaluate, exploiting the SPMD parallelism of the GPU card in the hope that the evaluation function is not too divergent.

In case of Genetic Programming, an interpreter executes individuals on the card. Because each individual is different, parallelization is done over the learning set that will ideally contain more than 32 values. Then, several individuals are also assigned to a 32 core SIMD multi-processor for maximum efficiency spatio-temporal parallel evaluation.

Typically, observed speedups range from x100 to x300 on 2012 hardware, depending on the CPU and GPU cards.

2. SUPPLIED CODE

The supplied code contains the complete parallelization platform to implement an evolutionary algorithm contained in a `.ez` file for execution on CPU or on GPU.

In order to get the best results, please use a machine that is running under Linux, with CUDA installed correctly and ideally, Java, for an interactive display during the run.

First, unzip the provided `easea.zip` file into a directory (called `easea`, for instance), `cd` into it and type `make`.

This should install the EASEA parallelizing platform. Then, follow the steps to make the EASEA platform functional (mainly, make sure that the environment variable `EZ_PATH` is set to the directory that contains the EASEA compiler, ending with a "/", by typing `export EZ_PATH=‘pwd‘/` while in the `easea` directory for instance.).

3. TESTING THE SUPPLIED EXAMPLE ON ONE MACHINE

In the `examples` directory provided under the `easea` directory, `cd` into the `weierstrass` directory.

This directory contains the `weierstrass.ez` file that contains the code to solve the Weierstrass benchmark, that can be modified using a standard ascii editor.

The `weierstrass.ez` file can be compiled into C++ (or nvcc) source files by typing:
`$ easea weierstrass.ez` // for a sequential execution on CPU, or
`$ easea weierstrass.ez -cuda` // for a parallel execution over GPU cards.

EASEA will automatically detect how many GPU cards are in the computer (beware, if you have a small NVIDIA GPU card for display and a larger NVIDIA GPU card for computation, EASEA will detect both and may execute half the evaluations on each of the cards, resulting in suboptimal speedup). Ideally, in order to maximize speedup, the computer should be equipped with identical GPU cards.

Once the source code is produced, a makefile is provided that allows to compile it by simply typing `make`.

One can then start an execution by typing:
`./weierstrass`.

If Java is installed, a Java windows should pop up to display the best and average individuals along generations, otherwise, start the run by typing:
`./weierstrass -plotStats 0`

Depending on the GPU and CPU hardware, the code generated by EASEA with the `-cuda` option should run several hundred times faster than without this option, cf. Fig. 3.

4. TESTING THE SUPPLIED EXAMPLE ON SEVERAL MACHINES

A file called `ip.txt` is provided in the `weierstrass` directory. By default, this file contains several lines referring to the local IP number of the machine with a port number appended to it.

This allows to parallelize the execution on several CPU

GECCO '12 Philadelphia, USA

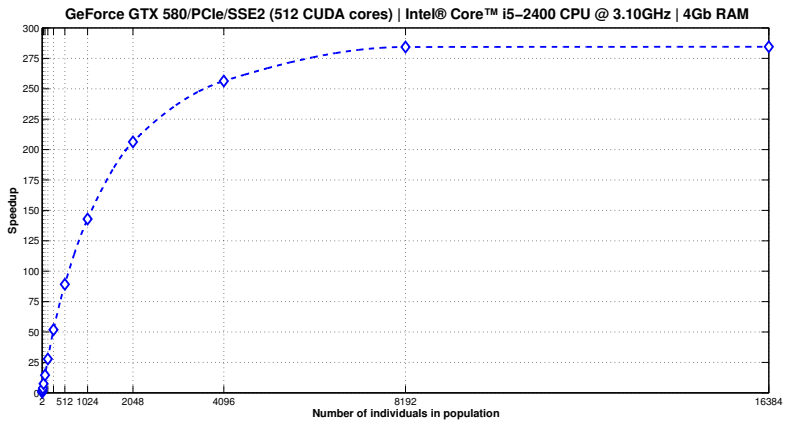

Figure 1: Obtained speedup of $\times 284.5$ on Weierstrass function between core i5 CPU and GTX580 NVIDIA card depending on population size.

cores of the same machine using an island model, by launching the execution in different windows and specifying different port numbers as in:

```
$ ./weierstrass -serverPort 2930
$ ./weierstrass -serverPort 2931
```

This will start two runs that will periodically exchange individuals. If the population size is too big for a quick CPU execution, a smaller population can be specified on the command line by typing:

```
$ ./weierstrass -serverPort 2930 -popSize 100
-nbOffspring 100
```

If several GPU machines are available, then, one can compile for GPU with the `-cuda` option and specify as many IP numbers as there are involved machines (don't forget to append the default port number :2930 to the IP number) in the `ip.txt` file.

The island model is scalable and allows to use n machines on a single problem at once with or without GPU cards.

5. CONCLUSION

The algorithms implemented into the EASEA platform have all been published extensively [3, 4, 6] and are quite efficient (the computed GPU utilization on Weierstrass is well over 90%, which allows for very large speedups), both for standard artificial evolution (GA and ES) and for Genetic Programming, which is more tricky to get right.

A big effort has been put into making the algorithms available to all the scientific community through the EASEA parallelization platform, that can produce code for CPU or GPU out of the same specification file.

Thanks to the island model, the EASEA platform presented here scales well over many machines and allows to obtain linear and supra-linear speedup on complex problems.

In 2010, the EASEA platform in Strasbourg made of 20 Intel Xeon 5500 2.57Ghz Core i7 CPU with 8GB Ram with each a GTX275 240 cores NVIDIA card allowed to contribute to find a new Zeolite crystalline structure in around 40 hours rather than 11 years, thanks to a speedup of $20 \times 120 = 2400$. This work was extensively described over two pages of the Supplied Online Material of the 2011 Science paper in which this new Zeolite structure was published [1, 5].

An EASEA cluster of 18000 GPU cores is used on an everyday basis in Strasbourg (20 machines with each 2x GTX560TI cards) and a new cluster of 30000 GPU cores (20 machines with each a 1536 cores GTX680 card) is currently under assembly, for a computing power of 60 TFlops.

These machines are used to compute massive inverse problems both for chemistry, biology and aerodynamics.

A French national project called EASEA-CLOUD has been secured in 2012 ($500K over 2 years) to turn the EASEA platform into an industrial grade solver that will run on a GRID and a CLOUD of computers.

6. REFERENCES

[1] J.Jiang, J. L. Jorda, J. Yu, L. A. Baumes, E Mugnaioli, M. J. Diaz-Cabanas, U. Kolb, A. Corma, "Synthesis and Structure Determination of the Hierarchical Meso-Microporous Zeolite ITQ-43", *Science*, 26 August 2011 : Vol. 333 no. 6046 pp. 1131-1134 DOI: 10.1126/science.1208652

[2] Collet, P., Lutton, E., Schoenauer, M., Louchet, J.: Take It EASEA. Proceedings of the 6th International Conference on Parallel Problem Solving from Nature. **PPSN VI** (2000) 891–901

[3] Maitre, O., Baumes, L., Lachiche, N., Corma, A., Collet, P.: Coarse Grain Parallelization of Evolutionary Algorithms on GPGPU Cards with EASEA. 11th Annual Conference on Genetic and Evolutionary Computation. GECCO 2009 403–1410

[4] Krüger, F., Maitre, O., Jimenez, S., Baumes, L., Collet, P.: Speedups between x70 and x120 for a generic local search (memetic) algorithm on a single GPGPU chip. EvoNum 2010. **LNCS** (2010)

[5] Baumes, L.A., Krüger, F., Jimenez, S., Collet, P., Corma, A.: Boosting theoretical zeolitic framework generation for the determination of new materials structures using GPU programming. Phys. Chem. Chem. Phys. **Vol. 13** (2011) 4674-4678

[6] O. Maitre, F. Krüger, S. Querry, N. Lachiche, P. Collet. "EASEA: Specification and Execution of Evolutionary Algorithms on GPGPU," *Soft Computing* - A Fusion of Foundations, Methodologies and Applications, pp. 1–19, may 2011.