THE IMPORTANCE OF ESTABLISHING A METHODOLOGY FOR ANALYSIS OF AN ALGORITHM BETWEEN ARCHITECTURES

SILVA, B. C. [1]*; LIMA, M. E.* [1]*; TEICHRIEB, V.* [1]*; CARVALHO, Q. G. da S*[2].
[1] Federal University of Pernambuco, Computer Science Center, Cidade Universitária, 50740-540, Recife – PE, Brazil.
[2] Esp. in security Work, resident of the 2ºyear in specialty Hematology and Hemotherapy. SES-PE.

ABSTRACT

Lately, computer games provide highly sophisticated features and therefore demand great computational power, being comparable to some scientific processing algorithms, well known as applications with a high demanded computational performance. The scientific community, perceiving the potential of the graphics cards demanded by these games, that provided high performance parallel vectorial processing, started to use them for general purpose scientific calculations. CUDA is a programming language similar to C language, with some modifications that allow the elaboration of algorithms in a simple way and with a good performance in GPU. To analyze an architecture technology is a task that requires a great bibliographic review. Create a methodology and analyze an algorithm along various architectures is important to decide which is the best technology to run a specific program. This paper shows the first steps taken in order to create such a methodology, which initially will be used to compare the CPU and GPU architectures when running a seismic related algorithm.

1. INTRODUCTION

Lately, computational applications have been demanding even more high performance processing. In fact, these Requirements already existed, but they were supplied by using mainframes, supercomputers or computer clusters. High performance problems solution is an area in the world-wide market supplied by companies using technologies such as Cell, GPGPU (General-Purpose computation on Graphics Processing Units), cluster, among other architectures, with the intention of offering the best solution, in the most diverse application domains: financial analysis, data mining, image processing, scientific computation, computational modeling, etc.[1].

A way to disencumber this great demand for processing in the CPU is to use a device that is common good at the hands of the users of games, the graphic cards. They have evolved and reached a high degree of performance efficiency in the data processing task aiming more complex games. Moreover, its cost is small if compared to dedicated specialized systems for high performance problems solving. Currently, GPUs are able to keep the order of teraflops of processing, which is the measure applied to high performance devices [2].

On the other hand, already for two decades the use of programmable logical devices, such as FPGAs (Field Programmable Gate Arrays), is consolidated for the solution of problems through Prototyping. The use of FPGAs is advantageous when the cost to manufacturing the chip of a specific solution is not viable, or when the product of the solution changes constantly compromising financially the final value. In addition, they have being used for HPC solutions mainly because of the low power consumption what does not happen in the GPU case. Beyond, they provide a significant performance improvement due to some characteristics, such as its intrinsic parallelism, larger bandwidth for memory access and customization flexibility [3].

This paper describes a methodology for analysis of a high performance algorithm. The algorithm models in 2D problems related to seismic. The analysis will give indications about the advantages and disadvantages of executing the algorithm on a GPU (using CUDA (Compute Unified Device Architecture) technology) and a CPU.

2. METHODOLOGY

In this section we describe the environment, information collected for the benchmark, and show the first steps of the methodology for analysis performance and their calculations.

2.1 Development environment

The implementation of the algorithm and the subsequent analysis have been made using a notebook with Intel Core 2 Duo P8600 2.4G, Front Side BUS of 1066MHz, RAM DDR2 800 memory of 4GB, NVIDIA GeForce 9800M GS graphics card with video memory of 1GB DDR3 VRAM. The operational system was Windows Vista Home Premium of 32 bits with SP1, and the Eclipse with MinGW GCC Compiler.

The algorithm, versions in C and CUDA, will be developed and tested using the computer described above. The programs will be tested in an idle Windows Vista Home.

2.2 Benchmark

GPUs allow clock change in order to improve efficiency in relation to the power used, or allow defining its constant speed. In this project, the NVIDIA graphics card offers the nTune and RivaTuner tools to correct the clock.

In many graphics cards an overclock of 5% is already enough to destabilize the equipment and not get any gradual result in overclock. But, at this moment the intent is not making an overclock but estimate performance, not allowing that our test environment suffers alterations at each test.

Notebooks provide the NVIDIA PowerMizer Technology. This technology allows notebooks reducing and increasing efficiency so that the battery is minimized and drawn out for more time. Some CUDA examples were executed, and using the programs GPU-Z (performance measuring program specific for GPUs) and Lavalys Everest (general performance measuring program) none inferior clock limit was observed both on the GPU and the CPU. Even being a notebook, the computer used in this work (and described above) offers all features found on a workstation.

The algorithm tackled in this paper uses a huge data matrix, demanding a great amount of memory that impacts its performance. So more closer the data stay from block processing occurs less latency. Soon, the amount of memory in the DEVICE influences the algorithm performance and the architecture that will be chosen for the solution.

To supply this requirement our system provides 4GB of RAM in the CPU architecture, and 1GB of GDDR3 memory in the GPU one.

Our measure methodology performs efficiently when dealing with algorithms that do not make use of the interference of the user and pass most of the time executing in the processor. This type of algorithm is called CPU-bound [4].

The performance tests will be made in a CPU-bound way. Therefore, after the exits of the algorithms are validated the function will be annulled. Otherwise, we would lose the characteristics of cpu-bound and could not adequately perform the benchmark.

Clock counting is more precise than time counting. In case a time counting is used, the result is given in seconds, and if the processor varies the algorithm performance result changes as well. Using a cycle of clock counting approach limits this error, allowing the generation of information for comparison purposes. In order to perform such time measurement a simple equation can be used that is valid both in the CPU and GPU cases, with cycles of the divided clock frequency of the processor we have an experimental value of time. Follows the equation:

$$\boxed{S(\text{ seconds }) = \frac{\text{total cycles}}{\text{processor frequency}}}$$

2.3 Methodology for performance analysis

Our algorithm generates two models in an iterative and incremental way. Firstly, the algorithm was implemented using the C language, and defined as the reference algorithm. Its exit is considered standard, and is reference for the CUDA algorithm exit.

After that, a CUDA version of the algorithm was implemented and verified against the reference model in C.

When translating the algorithm from the C based CPU architecture to the CUDA architecture, some modifications to the algorithm were necessary. This was important in order to optimize the algorithm for running on the GPU, taking advantage of its massive parallelism.

It is important to say that identify the bottlenecks of the algorithm and know how to use wisely the architecture in order to respect restrictions and take advantage of its features is a process that requires a deep understanding of the architecture and the algorithm, as well as a profound bibliographic review.

2.4 Performance calculation

Each time the initial and final clock value is calculated, it will be collected, respectively, the value of the accountant before and after the execution of the main process, and then they will be subtracted and the value will be kept in a set. After a series of calculations an analysis of these values is performed by computing its median value. The median value has been adopted because it shows a measure of central trend, in a way that the chosen value separates in 50% the population above and below it.

3. RESULTS

The results gotten through the use of the methodology described in this paper are part of an undergraduate work developed at the Computer Science Center of the Federal University of Pernambuco.

It is expected that through this methodology an effective experimental comparison between architectures will be possible. In each architecture the algorithm will show advantages and disadvantages, and the methodology should help developers chose which architecture fits better to the seismic problem.

In parallel to this work, another team is developing an FPGA version of the algorithm tackled in this paper, for later comparison. Therefore, not only a CPU and GPU comparison will be possible, but an extension to a CPU vs. GPU vs. FPGA comparison.

5. REFERENCES

[1] *GPU Programming Guide GeForce 8 and 9 Series*. Available in: http://developer.download.nvidia.com/GPU_Programming_Guide/GPU_Programming_Guide_G80.pdf. Access in: 22/03/09.

[2] *Nvidia closing in on 2 teraflops with graphics card*. Available in: http://www.computerworld.com.au/article/272446/nvidia_closing_2_teraflops_graphics_card. Access in: 13/04/09

[3] CORPORATION, NVIDIA CUDA. Programming Guide 1.0. *NVIDIA*, 2007.

[4] U*sing the rdtsc instruction for performance monitoring*. Available in: http://cs.smu.ca/~jamuir/rdtscpm1.pdf. Access in: 08/07/2008.