Fusion Engineering and Design 87 (2012) 2152–2155

Contents lists available at SciVerse ScienceDirect

Fusion Engineering and Design

journal homepage: www.elsevier.com/locate/fusengdes

A GPU-based real time high performance computing service in a fast plant system controller prototype for ITER

J. Nieto[a,*], G. de Arcas[a], M. Ruiz[a], J. Vega[b], J.M. López[a], E. Barrera[a], R. Castro[b], D.Sanz[a], N. Utzel[c], P. Makijarvi[c], L. Zabeo[c]

[a] Grupo de Investigación en Instrumentación y Acústica Aplicada. Universidad Politécnica de Madrid, Crta. Valencia Km-7, Madrid 28031 Spain
[b] Asociación EURATOM/CIEMAT para Fusión, Madrid, Spain
[c] ITER Organization, CS 90 046, 13067 St. Paul lez Durance Cedex, France

HIGHLIGHTS

- Implementation of fast plant system controller (FPSC) for ITER CODAC.
- GPU-based real time high performance computing service.
- Performance evaluation with respect to other solutions based in multi-core processors.

ARTICLE INFO

Article history:
Available online 1 June 2012

Keywords:
Graphic processing units (GPU)
Data acquisition and processing
Fusion experiments

ABSTRACT

EURATOM/CIEMAT and the Technical University of Madrid UPM are involved in the development of a FPSC (fast plant system control) prototype for ITER based on PXIe form factor. The FPSC architecture includes a GPU-based real time high performance computing service which has been integrated under EPICS (experimental physics and industrial control system). In this work we present the design of this service and its performance evaluation with respect to other solutions based in multi-core processors. Plasma pre-processing algorithms, illustrative of the type of tasks that could be required for both control and diagnostics, are used during the performance evaluation.

1. Introduction

The use of GPUs (graphical processing units) to include high parallel processing power in data acquisition (DAQ) systems for real time data assessment is attracting considerable attention in the scientific community [1–3]. This is of special relevance in the fusion community as the requirements of the data acquisition systems to be used in next generation machines increases due to the need of including new functionalities for real time data assessment. There is a need to include new techniques based on pattern recognition, event detection and data reduction in addition to their conventional tasks of data acquisition and logging [4].

The goal of this work is to evaluate the possible benefits of using GPU based technologies for data preprocessing. In particular, the performance of a GPU running an algorithm representative of the type of operations that would be needed in plasma preprocessing has been evaluated in comparison with that obtained running the same algorithm in a similar state-of-the-art CPU. Also, a methodology to integrate GPUs in distributed control systems based on EPICS [5] is proposed.

The hardware test system consists of a commercial Workstation Hewlett-Packard model Z600, that hosts two Xeon X5550 QuadCore processors at 2.66 Ghz with 4 Gbytes DDR3 RAM, and an Asus ENGTX580 board to run the GPU tests. This NVIDIA Fermi-Architecture board has 1536 Mbyte GDDR5 RAM and can maximize the data transfer rate between the CPU and the GPU using a PCI-Express Gen 2 Slot interface with 16 data lines. This board includes 16 third generation Streaming Multiprocessors and each multiprocessor houses 32 CUDA cores (512 CUDA cores in total). The software running in the host processor has been developed in C language under RHEL v5.5. Table 1 shows the software configuration of the system.

2. Algorithm description

A simple search in the Internet will provide several performance benchmarks, such as [6], reporting the benefits of using GPUs for several data processing functions. These reports usually compare the execution time of typical data processing routines running in a GPU with those obtained in a CPU. Although they provide valuable

* Corresponding author. Tel.: +34 91 336 5227.
E-mail address: jnieto@sec.upm.es (J. Nieto).

0920-3796/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.fusengdes.2012.05.008

Table 1
Software tools and configuration.

Host processor software	
OS	RedHat Enterprise Linux 5.5
Middleware	EPICS 3.14.12 and asynDriver 4.16
Compilers	gcc V4.12.20080704 and nvcc V0.2.1221
CPU libraries	MKL 10.3 Update 9 and IPP 7.0
GPU libraries	NVIDIA SDK 3.2, NVIDIA CUBLAS 3.2, EMPHOTONICS CULA R11

information, they are not fully representative of the possible benefits that can be obtained in a real control application, as they do not include other issues as the impact of data transfers, development of custom processing routines, and system integration issues. Therefore in this work we have selected an algorithm representative of the type of operations that would be needed in plasma pre-processing.

The chosen algorithm is essentially a best fit code for detecting position and amplitude of a spectra composed by a set of Gaussians based on Levenberg–Marquardt method [8]. It is used to calculate ion temperature (T_i) from the charge exchange camera installed at JET. Fig. 1 shows a flowchart describing the algorithm. For each input sequence, a set of initial coefficients pointing to the possible location of the Gaussians must be provided. On completion, the algorithm provides a fitted version of the input sequence, along with the coefficients describing the position and amplitude of the Gaussians. Fig. 2 shows a typical example of the input (input data) and output (fitted data) sequences.

3. Implementation

Another important aspect of this work has been the integration with the distributed control system middleware, EPICS in this case. EPICS is an open source technology that includes a wide set of tools and applications to implement distributed control systems. Using client/server and publish/subscribe techniques to

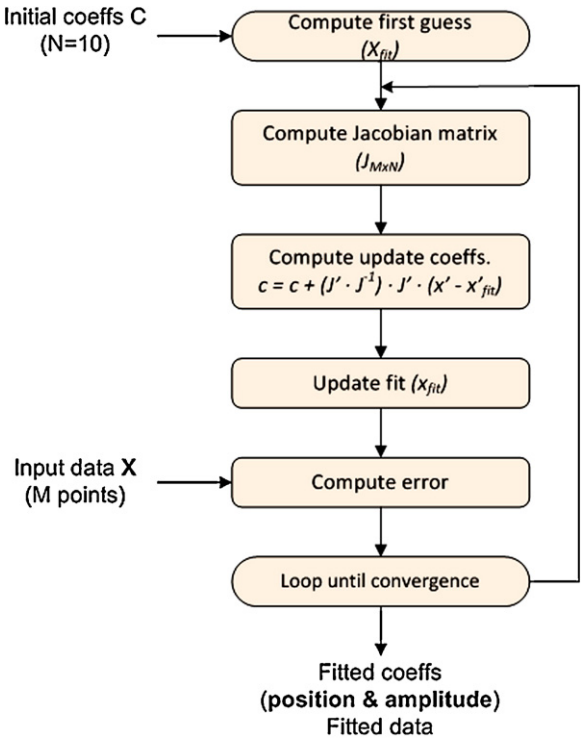

Fig. 1. Algorithm steps in Levenberg–Marquardt algorithm.

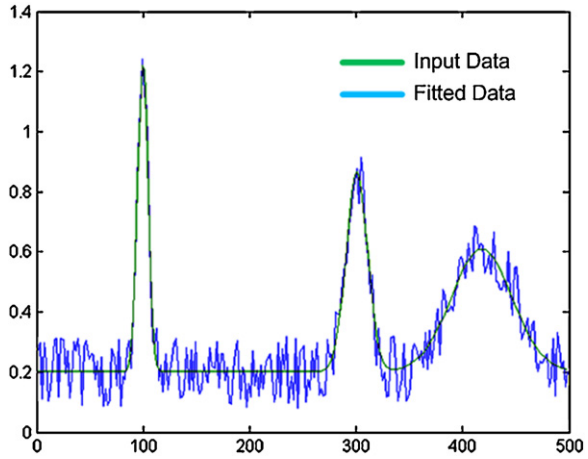

Fig. 2. Input and output sample sequences for TiCameraFit algorithm.

communicate between computers, EPICS not only is able to manage local control tasks in these computers but also is able to publish and monitor their control parameters. Additionally, asynDriver [7] is an extension for EPICS that simplifies and homogenizes interfaces with the different devices to control.

EPICS has an important lack for transferring massive data (from high acquisition rates) between the local tasks of a system. The classic EPICS links between EPICS records are not "quick" enough for this purpose, so a new data block links technology have been implemented between the local concurrent tasks, optimizing data throughput, minimizing system interrupts and minimizing CPU consume. DPD (data processing and distribution) is the complete solution that is compatible with EPICS and asynDriver module (it only uses components and libraries provided by them), includes the commented data block link technology and all necessary routing commands and performance measures.

The system developed runs in a RHEL (RedHat Enterprise Linux) 5.5, with EPICS 3.14.12 and asynDriver 4.16. It has been implemented in C using DPD technology and is composed by the following modules:

- TiC Data Generator. This module simulates the acquisition of the input signal by generating a noisy version of a spectra composed by a set of Gaussians as shown in Fig. 2. The length of the data blocks can be changed during the system configuration phase as it would be done in a real situation.
- EPICS Waveform Monitoring. This module is able to monitor input signals, which are received by its input block links, through EPICS waveform process variables. The module is able to automatically detect the type of data that has been received from a signal, and to generate the corresponding data to be published using an EPICS waveform process variable.
- GPU TiC Data Fit. This module is the GPU implementation of the algorithm described. It has been developed using EMPhotonics' CULA Tools Premium library R11, and NVIDIA's CUBLAS library v3.2. Both libraries support different GPU platforms, so they provide a portable code that can be used with different hardware setups. In addition some custom functions (kernel functions) have been developed, such as the one to compute the Jacobian matrix. The module also includes the data transfers between the CPU and the GPU.
- CPU TiC Data Fit. This module is the corresponding CPU implementation. In this case it has been developed using Intel's Math Kernel Library (Intel® MKL) v10.3 for 64 bits Linux systems.

From the control point of view, the number of points of every data block generated by the "TiC Data Generator" module can be

Table 2
GPU vs CPU (MKL-8 cores) execution times.

Block size	GPU		CPU	
	Time (ms)	SD (%)	Time (ms)	SD (%)
256	2.86	5	2.75	1
512	2.85	1	4.83	7
1024	3.6	0	8.69	5
2048	5.21	0	16.07	39
4096	16.42	0	28.55	0
8192	42.85	0	55.26	3
16384	85.40	0	107.50	1
32768	168.65	0	210.99	3
65536	334.77	0	425.96	0

Table 3
GPU vs CPU (MKL-1 core) execution times.

Block size	GPU		CPU	
	Time (ms)	SD (%)	Time (ms)	SD (%)
256	2.86	5	2.69	1
512	2.85	1	5.09	1
1024	3.6	0	9.21	0
2048	5.21	0	15.93	0
4096	16.42	0	29.09	0
8192	42.85	0	55.44	0
16384	85.40	0	107.92	0
32768	168.65	0	216.97	0
65536	334.77	0	445.74	0

configured using the corresponding EPICS process variable. The rest of modules of the system (CPU and GPU TiC Data Fit, and EPICS waveform monitoring) are able to manage these data blocks independently of the number of values they carry. Additionally, some performance values are monitored using the corresponding EPICS PVs:

- The TiC Data Generator module provides an EPICS PV that measures the throughput of the generated signal in bytes/s.
- Each Data Fit module provides two performance values: an average value of the number of iterations required by the fitting algorithm, which can be used to detect convergence problems; and the execution time of the algorithm.
- The block links that connect the different modules of the system provides two values: percentage of occupancy of the link buffer; and data transfer rate in the link in bytes/s.

Regarding both implementations of the algorithm, it has to be mentioned that standard coding practices for control algorithms have been used, such as pre-allocating memory buffers, etc. Optimization of both algorithms has been based on the use of the optimized primitives provided by the corresponding libraries, but no further optimization techniques, such as pipelining, etc., have been included as the goal of this work was not to develop the best possible code for the algorithm, but to compare the results that both platforms can provide with similar implementations.

4. Results

Different tests have been developed to automate the benchmarking process. All of them have been parameterized, so the impact on the data block sized can be assessed. In order to measure execution times both platforms, GPU and CPU, provide hardware mechanisms that can be used to benchmark both algorithm implementations. The GPU provides a one microsecond resolution timer that can be accessed through specific NVIDIA library functions. This timer has been used during the development phase to analyze the amount of time taken by each algorithm step in order to detect possible bottlenecks and to focus coding efforts. The CPU also provides a hardware timer that can be accessed through the corresponding libraries, but with a much higher resolution as it runs at the microprocessor's clock, in our case 2.26 GHz. Therefore this timer has been used to obtain all performance measurements.

The following tables present a summary of the results obtained. Table 2 shows the execution times obtained when the CPU module is allowed to use all available cores. In this case the GPU is not much faster than the CPU, but it has to be notice that we comparing a low cost GPU, in the order of 500€/board, with a top-of-scale CPU, in the order of 1500€/microprocessor. But even under these circumstances, the GPU is able to overperform the CPU as the input data block size increases. This is normal as it is well known that GPUs are especially interesting for processing high amount of data.

But execution times are not the only important aspect to consider in control algorithms. An even more important aspect is determinism. Table 2 uses the standard deviation of the execution time of each implementation through a set of 100 runs in order to assess its determinism. If all cores are used to obtain a faster implementation, determinism is compromised as the impact of other tasks, including the operation system, is high. This is shown more clearly in Table 3, where the number of cores that the CPU implementation is able to use is restricted to its minimum, or in Fig. 3 where the time evolution of the execution time for each configuration is shown together with the occupancy of the 8 CPU cores. Of course the price paid on the execution time on the CPU is compensated with higher determinism. On the contrary, the determinism of the GPU is almost invariable, as this device is only used to perform this task, and therefore the only source of variability could come from the data transfer process. For the data block sizes used during the tests, the time of the data transfers is small enough such as to obtain an almost ideally deterministic behavior. In our opinion this can be one of the most important benefits of GPUs as they do not require the use of real time extensions of the operating systems to obtain a deterministic performance. Table 4 shown the improvements obtained the GPU versus a CPU execution with 1 and 8 cores versions.

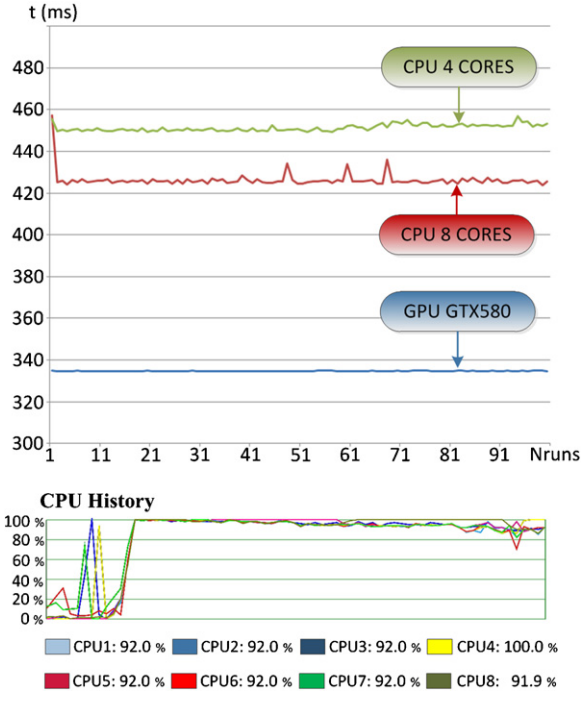

Fig. 3. GPU execution times vs CPU execution times, CPU occupancy 8 cores version.

Table 4
GPU improvements.

Block size	GPU vs CPU-8t	GPU vs CPU-1t
256	−3.85%	−6.36%
512	40.94%	44.02%
1024	58.14%	60.49%
2048	67.54%	67.26%
4096	42.49%	43.57%
8192	22.45%	22.70%
16384	20.56%	20.86%
32768	20.07%	22.27%
65536	21.41%	24.90%

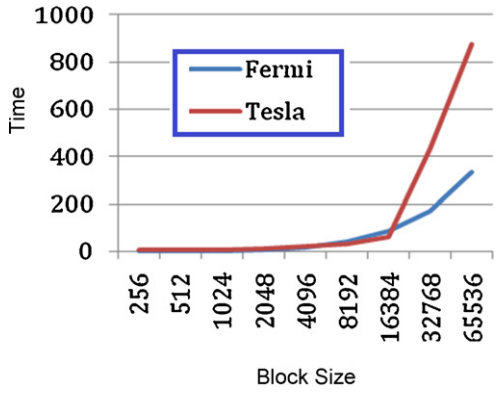

Fig. 4. Execution times in Tesla GPU vs Fermi GPU.

In order to assess the real time capability of a system solution the real time constraints must be defined first. In our case, real time performance is achieved as long as the time needed to run the algorithm for a data block of known length, including data handling and processing, is less than the time it takes to acquire it, minus a guard time included to account for non deterministic behaviors, such as the ones discussed in the previous paragraphs. So once the sampling frequency of the input data is known, the data shown in the following tables permits us to assess the real time capabilities of the implementation for a given guard time.

In order to evaluate the performance over two generations of NVDIA boards, the tests described above were repeated for a GTX580 (Fermi technology) card and a Tesla C1060 card (4 GB GDDR3 RAM with 240 CUDA cores). Fig. 4 shows the execution time on both cards and the improved performance obtained with the new Fermi family, due to the total number of cores in both architectures, 240 against 512 in Fermi. Also the core clock in the new architecture has been updated from 1296 MHz to 1544 MHz. Fermi cards include third 16 generation streaming multiprocessor (SM), each of which contains 32 CUDA processors. Each CUDA processor has a fully pipelined integer arithmetic logic unit (ALU) and a 64 bit floating point unit (FPU) compatible with the IEEE 754-2008 industry standard for floating-point arithmetic solving the problem of limited precision. Each SM on the GTX 580 is able to execute up to 16 double precision fused multiply-add (FMA) operations per clock with a peak throughput of $16 \times 16 \times 1544\,\text{MHz} \times 2 = 790.5$ GFLOPS. These new features introduced in the Fermi cards not only mean a new boost in performance as shown in Fig. 4, but also more suitability for control and data analysis applications, as it solves the problem of limited precision operations.

However, GPUs are probably not the best option for pure control applications, meaning those that deal with small data blocks, due to the overhead included in data handling and the difficulties to solve synchronization issues. Other technologies, such as FPGAs are proving to be much more suitable in these cases [9–12]. But for "real time" data analysis applications, meaning those that deal with medium to large data block sizes and complex algorithms, GPUS are becoming a very interesting alternative, even outperforming FPGAs thanks to their increase in floating point processing power. In our opinion, generally speaking, and as a rule of thumb, FPGAs are specially suited for control and data preprocessing applications, whereas GPUs are becoming a good alternative for solving complex data analysis tasks in real time.

Finally, another important feature to note is the power saving of the GPU-based platforms, as the consumption required to achieve the same performance CPU systems is greater.

Acknowledgments

This work is funded by the Spanish Ministry of Science and Technology under the Projects nos. ENE2009-10280 and ENE2008-08294/FTN and the contract ITER/CT/09/10001533.

This work, supported by the European Communities under the contract of Association between EURATOM/CIEMAT, was carried out within the framework of the European Fusion Development Agreement. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

References

[1] G. Collazuol, G. Lamanna, M. Sozzi, A trigger system based on graphics processing unit (GPU), in: Proceedings of the 17th IEEE-NPSS Real Time Conference, 2010.
[2] S. Chilingaryan, A. Kopmann, A. Mirone, T. dos Santos Rolo, A GPU-based architecture for real-time data assessment at synchrotron experiments, in: Proceedings of the 17th IEEE-NPSS Real Time Conference, 2010.
[3] S. Gorbunov, et al., ALICE HLT high speed tracking and vertexing, in: Proceedings of the 17th IEEE-NPSS Real Time Conference, 2010.
[4] J. Vega, A. Murari, B. Carvalho, G. de Arcas, et al., New developments at JET in diagnostics, real-time control, data acquisition and information retrieval with potential application to ITER, Fusion Engineering and Design 84 (2009) 2136–2144.
[5] "EPICS home web page", http://www.aps.anl.gov/epics.
[6] Tesla C2050 Performance Benchmarks. http://www.microway.com/pdfs/TeslaC2050-Fermi-Performance.pdf.
[7] "asynDriver home web page", http://www.aps.anl.gov/epics/modules/soft/asyn/.
[8] J.J. Moré, The Levenberg–Marquardt algorithm: implementation and theory, Lecture Notes in Mathematics 630 (1978) 105–116.
[9] M. Ruiz, et al., Real time disruptions detection in JET implemented with the ITMS platform using FPGA based IDAQ, in: Proceedings of the 17th IEEE-NPSS Real Time Conference, 2010.
[10] L. Esteban, et al., Continuous plasma density measurement in TJ-II infrared interferometer – advanced signal processing based on FPGAs, Fusion Engineering and Design 85 (3–4) (2010) 328–331.
[11] I. Balboa, B. Huang, G. Naylor, et al., Laser beam combiner for thomson scattering core LIDAR, Review of Scientific Instruments 81 (10) (2010).
[12] V. Svoboda, B. Huang, J. Mlynar, et al., Multi-mode remote participation on the GOLEM tokamak, Fusion Engineering and Design 86 (6–8) (2011) 1310–1334.