Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 113 (2017) 496–501

Procedia Computer Science

www.elsevier.com/locate/procedia

International Workshop on Data Mining on IoT Systems (DaMIS17)

A virtualized software based on the NVIDIA cuFFT library for image denoising: performance analysis

Ardelio Galletti[a], Livia Marcellino[a,*], Raffaele Montella[a], Vincenzo Santopietro[a], Sokol Kosta[b]

[a]Department of Science and Technology, University of Naples Parthenope, Centro Direzionale Isola C4, 80143, Naples Italy
[b]Aalborg University Copenhagen The Faculty of Engineering and Science The Technical Faculty of IT and Design Department of Electronic Systems Infrastructure, Services and Entrepreneurship Center for Communication, Media and Information Technologies

Abstract

Generic Virtualization Service (GVirtuS) is a new solution for enabling GPGPU on Virtual Machines or low powered devices. This paper focuses on the performance analysis that can be obtained using a GPGPU virtualized software. Recently, GVirtuS has been extended in order to support CUDA ancillary libraries with good results. Here, our aim is to analyze the applicability of this powerful tool to a real problem, which uses the NVIDIA cuFFT library. As case study we consider a simple denoising algorithm, implementing a virtualized GPU-parallel software based on the convolution theorem in order to perform the noise removal procedure in the frequency domain. We report some preliminary tests in both physical and virtualized environments to study and analyze the potential scalability of such an algorithm.

Peer-review under responsibility of the Conference Program Chairs.

Keywords: Hybrid infrastructures, Virtualization, High Performance Computing, FFT, GPGPU, image denoising;

1. Introduction

Analysis and processing of very large datasets, or big data, poses a significant challenge. Massive datasets are collected, studied and interpolated in numerous domains, from mathematical and engineering sciences[5,15,11,12,13,16] to social networks[19], from biomolecular research, commerce and security[23,31] to IoT applications[3,4,7,8,9,10]. In order to obtain the best performance in analyzing big data, the use of computing machines with high processing power became necessary[20]. Virtualization technologies are currently widely deployed, as their use yields important benefits such as resource sharing, process isolation, and reduced management costs[1,21,26]. Thus, it is straightforward that the usage of virtual machines (VMs) in HPC is an active area of research. From a computational point of view, the hierarchical and heterogeneous high performance computing paradigm, that emerged in the last decade, delivers enough power to process spatial big data leveraging on massive multicore CPUs, general purpose graphic processing units (GPGPUs) and, recently, field-programmable generic arrays (FPGAs) and supported by solid state storage skyrocketing the long

* Corresponding author. Tel.: +39-081-5476676.
E-mail address: livia.marcellino@uniparthenope.it

1877-0509 © 2017 The Authors. Published by Elsevier B.V.
Peer-review under responsibility of the Conference Program Chairs.
10.1016/j.procs.2017.08.310

term memory access performance[25,30]. In order to mitigate the computational infrastructures total cost of ownership, and accelerate the tools availability to a large users footprint, the use of high performance cloud computing resources pushes to minimize the big data based products time to market. While CPU virtualization and elastic storage is a common practice in public, private and hybrid clouds[22,28], the same techniques are not widely available due to the fact that often the accelerators exploit closed technologies.
In this paper we demonstrate how is possible to use the GPGPU resource leveraging on the RAPID GVirtuS[25,27] GPGPU virtualization service presenting a specific case related to the image denoising process. As is well-known, image denoising algorithms have a very high computational cost, especially when dealing with large-scale images. An effective approach to solve this problem is to use a parallel algorithm. There are many research efforts in this field, using different parallel computing architectures[2,6,17], especially on hybrid environment[14].
Here we present a denoising algorithm, based on a frequency domain convolution leveraging the CUDA environment, which exploits the computational power of the NVIDIA cuFFT library[1]. Our aim is to highlight the benefits in using virtualization without loss of efficiency. We used this simple case study in order to analyze how the overall virtualization cost affects the execution time. To be specific, starting by the Fourier Transform (FT) approach, we use the *convolution theorem* to perform the image denoising process as a pointwise product of FTs on GPU device. Therefore, we report some tests in both physical and virtualized environments and compare them.

The rest of the paper is organized as follows: in section 2 we recall the image denoising problem, the FT decomposition and the convolution theorem; in section 3 we present the cuFFT library of CUDA and how to use it efficiently: moreover a description of the GPU-parallel algorithm and of the virtualization approach with GVirtuS is also provided; the evaluation is carried out in section 4. Finally, in section 5 we conclude the paper.

2. The image denoising in frequency domain

Reconstruction of a signal from a noisy one is a well-known inverse problem. From a mathematical point of view a noisy image can be defined as a piecewise function $f(x, y) = C[u(x, y)]$, where $u(x, y)$ is the image function (non-corrupted by noise) and C is a function that defines the noise we wish to eliminate. Estimation of an unknown signal u from the available noisy data f, is an ill-posed problem that involves to find the noisy-free image, by preserving useful information. Noise in digital image may arise during the acquisition step, caused by poor illumination and/or high temperature. It can be either additive or multiplicative. Some noise models are also called *white*: the noise is spatially uncorrelated and identically distributed. Another often observed noise model is the so-called *salt and pepper* noise, which can be reduced with a median filter. In this work we deal with additive white Gaussian noise, defined as $p_G(u) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(u-\mu)^2}{2\sigma^2}}$ where u represents the image gray level and μ and σ are the Gaussian distribution mean and standard deviation, respectively. Generally, the simplest method to perform image denoising, in the spatial domain, is based on a convolution operation between the function f and the Gaussian function. The related algorithm uses a finite size mask (or kernel) which scannes across the image. The value of each pixel on the output image is the weighted sum of the input pixels within a window where the weights are the values of the mask. This algorithm has a high computational complexity, therefore this naive approach can become very slow for big inputs (requiring too much time for large images). A more efficient approach exploits the computational power of the Fourier Transform (FT) and the related frequency domain operations.

The Fourier transform of a function of time itself is a complex-valued function of frequency, defined as follows:

$$F(\eta) = \int_{-\infty}^{+\infty} f(t) e^{-2\pi i \eta t} dt.$$

The numerical approach for computing FT approximates a non periodic function as sum of a certain number of sine and cosine functions. The component frequencies of the sine a cosine functions that spread across the spectrum are represented as peaks in the frequency domain. Starting by the FT concept, the image denoising process in the frequency domain uses the *Convolution Theorem* for which: *under suitable conditions the FT of a convolution is the*

[1] https://developer.nvidia.com/cufft

pointwise product of FTs. Therefore, it is possible to write:

$$f(t) \otimes h(t) \iff F(\eta) \cdot H(\eta)$$

Then, to deal with the noise removal problem by avoiding a convolution in the spatial domain, we use the more efficient way which consists in performing a FT, computing an element-wise product and finally the related IFT. (inverse Fourier Transform). Following section describes the computational kernel of our parallel software.

3. GPU-Parallel Algorithm description and Virtualization approach

In order to develop an efficient parallel software we propose to use the cuFFT library for (direct and inverse) Fast Fourier Transforms computation in the GPU-CUDA environment. NVIDIA-CUDA Fast Fourier Transform (cuFFT) library provides a simple interface for computing parallel FFTs on an NVIDIA GPU environment. The library allows users to exploit the floating-point power and parallelism of the GPU without having to develop a custom GPU-based FFT implementation. By using cuFFT we gained a massive performance improvement with respect to the sequential CPU algorithm. Recently, cuFFT has been added to the CUDA libraries supported by GVirtuS.
GVirtuS[2] is an open source project with Apache License v2.0. Born in the University of Naples Parthenope (RAPID project)[18,24], the project is intended to provide a virtual version of computer hardware platforms, storage devices, and computer network resources. The GVirtuS scheme is based on the split-device driver model: device access control split between the frontend driver in the guest VM and backend driver in the host machine: GVirtuS frontend runs in a guest user space; GVirtuS backend is a server application that handles concurrent requests by multiple clients and runs in host user space. Each plug-in module of GVirtuS is composed by a frontend layer and a backend layer. The choice of the hypervisor deeply affects performances. GVirtuS project is paired to the KVM/QEMU hypervisor, which allows to reach high performance guest/host communication. For testing purposes, we can avoid the hypervisor layer and use TCP Unix sockets as communicator[29].
In particular, GVirtuS allows HPC on low-power devices, as GPUs and supports the CUDA libraries cuFFT, CUBLAS and CUDNN (work in progress). A GVirtuS offloading example is shown in Figure 1 on the right.

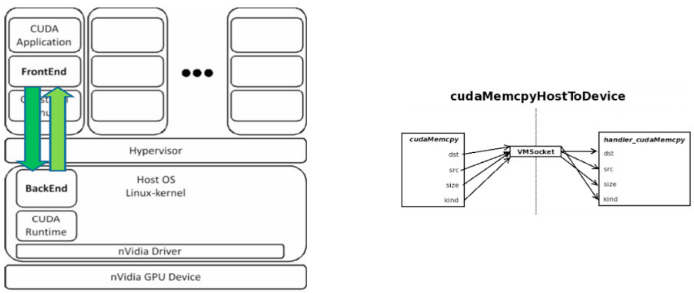

Fig. 1. Left: GVirtuS for GPU. Right: GVirtsuS offloading example.

The Guest OS calls the frontend definition of `cudaMemcpy` and sends the parameters to the remote machine trough the TCP Socket(Communicator). Of course, host pointers must be translated. Device pointers, instead, do not need any translation. The GVirtuS-CUDA stack allows to accelerate any virtual machine with a small impact on overall performance compared to a pure host/gpu setup.

[2] https://github.com/raffmont/GVirtuS

4. Performance test

In this section we report some experiments. We have tested two versions of the denoising GPU-parallel software: the first one in the GPU-CUDA environment, the second one in the GPU-CUDA environment with GVirtuS. Both versions, together with the sequential version, have been tested on a CPU: Intel Xeon E5-2609 v3 - RAM: 64 GB with a GPU: Nvidia GeForce Titan X. First, we compared the efficiency of our GPU-parallel software with respect to the sequential CPU version. The execution times are shown in Figure 2. We have observed a significant reduction in terms of execution times, mainly due to the use of the optimized cuFFT library.

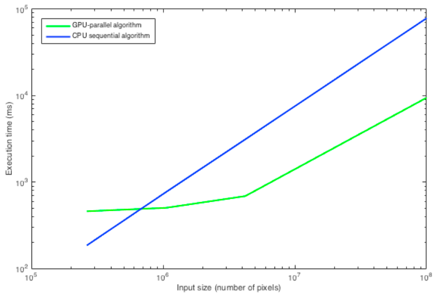

Fig. 2. Execution times (ms): in blue the CPU sequential code, in green the GPU-parallel code.

In order to run our CUDA software with GVirtuS, it is not necessary any modification; the code must be compiled on a machine with `nvcc` and the executable must link the NVIDIA shared libraries. Then, the compiled binary file must be moved to the VM with GVirtuS libraries, exporting some environment variables, in order to link GVirtuS shared libraries. Finally, the binary executable can be run.
In Table 1 we report the execution times of the GPU-parallel version without and with GVirtuS. The same results are also shown in Figure 3.

Table 1. Code execution times in ms

Image size	**GPU code with GVirtuS**	**GPU code**
512×512	565.856	461.781
1024×1024	703.875	505.825
2048×2048	1257.755	690.869
10000×10000	21302.798	9420.8

Since GVirtuS transfers data to the host machine, it is interesting to estimate the cost function due to the latency introduced by the communicator. Standard CUDA times and GVirtuS times of routine `cudaMemcpy` are reported in Table 2.

Table 2. The `cudaMemcpy` execution times in ms

Image size	**GPU code with GVirtuS**	**GPU code**
512×512	92.416	21.796
1024×1024	212.869	59.668
2048×2048	587.348	201.057
10000×10000	7303.230	4614.778

In order to study the virtualization cost, we assume that T_c is the time of the `cudaMemcpy` executed in a standard CUDA environment and T_{gv} is the time of the `cudaMemcpy` executed in the GVirtuS usage. With these notations, we can define a function that gives us an idea of the overall virtualization cost function, i.e. $L_v = |T_c - T_{gv}|$.

The Figure 3 (right) shows the obtained results. Note that the function grows linearly as the size of the problem increases and follows the trend of the curves in Figure 3 (left).

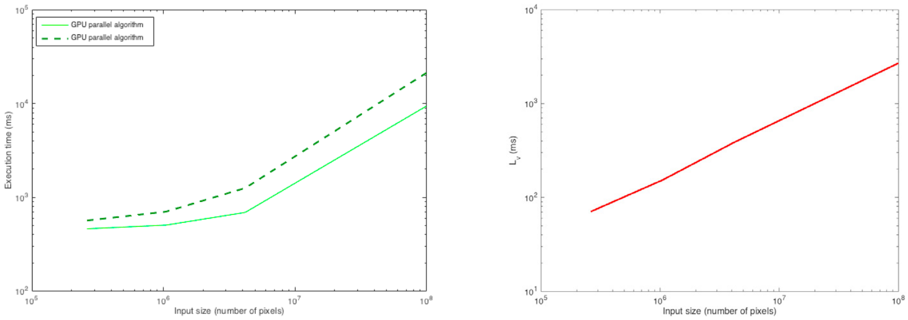

Fig. 3. Left: execution times (ms); the continuous green line is related to the GPU code without GVirtuS, the dotted green line is related to the GPU code with GVirtuS. Right: the virtualization cost function.

5. Conclusions

In this paper, we have presented a virtualized GPU-parallel algorithm in order to analyze the performance that can be obtained. To be specific, we have studied how the usage of the optimized cuFFT library of nVIDIA can be useful and governed in a virtualized environment. As case study we have considered a frequency domain denoising algorithm, reporting some preliminary tests in different environments. We have noticed that GVirtuS is a good solution for enabling GPGPU on Virtual Machines, or low powered devices, even for this kind of problem. This property depends on the fact that a massive optimized GPU code wins over a CPU implementation. Of course, the latency due to the TCP communicator grows linearly as the size of the problem increases, but it can be reduced by using a faster communicator (or network). This issue probably needs further investigations in a future work.

References

1. Isabella Ascione, Giulio Giunta, Patrizio Mariani, Raffaele Montella, and Angelo Riccio. A grid computing based virtual laboratory for environmental simulations. *Euro-Par 2006 Parallel Processing*, pages 1085–1094, 2006.
2. Daniel Castaño-Díez, Dominik Moser, Andreas Schoenegger, Sabine Pruggnaller, and Achilleas S Frangakis. Performance evaluation of image processing algorithms on the GPU. *Journal of structural biology*, 164(1):153–160, 2008.
3. Angelo Chianese, Fiammetta Marulli, Vincenzo Moscato, and Francesco Piccialli. A" smart" multimedia guide for indoor contextual navigation in cultural heritage applications. In *Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–6. IEEE, 2013.
4. Angelo Chianese, Francesco Piccialli, and Giuseppe Riccio. Designing a smart multisensor framework based on beaglebone black board. *Computer Science and its Applications*, pages 391–397, 2015.
5. S. Cuomo, G. De Pietro, R. Farina, A. Galletti, and G. Sannino. A novel O(n) numerical scheme for ECG signal denoising. In *Procedia Computer Science*, volume 51, pages 775–784, 2015.
6. Salvatore Cuomo, Pasquale De Michele, Ardelio Galletti, and Livia Marcellino. A GPU parallel implementation of the local principal component analysis overcomplete method for DW image denoising. In *Computers and Communication (ISCC), 2016 IEEE Symposium on*, pages 26–31. IEEE, 2016.
7. Salvatore Cuomo, Pasquale De Michele, Ardelio Galletti, and Francesco Piccialli. A Cultural Heritage case study of visitor experiences shared on a Social Network. In *10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC 2015, Krakow, Poland, November 4-6, 2015*, pages 539–544, 2015.
8. Salvatore Cuomo, Pasquale De Michele, Ardelio Galletti, and Giovanni Ponti. Visiting Styles in an Art Exhibition Supported by a Digital Fruition System. In *11th International Conference on Signal-Image Technology and Internet-Based Systems, SITIS 2015, Bangkok, Thailand, November 23-27, 2015*, pages 775–781, 2015.
9. Salvatore Cuomo, Pasquale De Michele, Ardelio Galletti, and Giovanni Ponti. *Data Management Technologies and Applications: 4th International Conference, DATA 2015, Colmar, France, July 20-22, 2015, Revised Selected Papers*, volume 584 of *Communications in Computer and Information Science*, chapter Classify Visitor Behaviours in a Cultural Heritage Exhibition, pages 17–28. Springer International Publishing, 2016.
10. Salvatore Cuomo, Pasquale De Michele, Ardelio Galletti, and Giovanni Ponti. *Intelligent Interactive Multimedia Systems and Services 2016*, volume 55 of *Smart Innovation, Systems and Technologies*, chapter Influence of Some Parameters on Visiting Style Classification in a Cultural Heritage Case Study, pages 567–576. Springer International Publishing, 2016.
11. Salvatore Cuomo, Ardelio Galletti, Giulio Giunta, and Livia Marcellino. A class of piecewise interpolating functions based on barycentric coordinates. *Ricerche di Matematica*, 63(11):87–102, 2014.
12. Salvatore Cuomo, Ardelio Galletti, Giulio Giunta, and Livia Marcellino. A novel triangle-based method for scattered data interpolation. *Applied Mathematical Sciences*, 8(134):6717–6724, 2014.

13. Salvatore Cuomo, Ardelio Galletti, Giulio Giunta, and Livia Marcellino. Piecewise Hermite interpolation via barycentric coordinates. *Ricerche di Matematica*, 64(2):303–319, 2015.
14. Salvatore Cuomo, Ardelio Galletti, Giulio Giunta, and Livia Marcellino. Toward a multi-level parallel framework on GPU cluster with PetSC-CUDA for PDE-based Optical Flow computation. *Procedia Computer Science*, 51:170–179, 2015.
15. Salvatore Cuomo, Ardelio Galletti, Giulio Giunta, and Livia Marcellino. Reconstruction of implicit curves and surfaces via rbf interpolation. *Applied Numerical Mathematics*, 116:157–171, 2017.
16. Salvatore Cuomo, Ardelio Galletti, Giulio Giunta, and Alfredo Starace. Surface reconstruction from scattered point via RBF interpolation on GPU. In *Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on*, pages 433–440. IEEE, 2013.
17. L. D'Amore, L. Marcellino, V. Mele, and D. Romano. Deconvolution of 3D fluorescence microscopy images using graphics processing units. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7203 LNCS(PART 1):690–699, 2012.
18. R. Di Lauro, F. Giannone, L. Ambrosio, and R. Montella. Virtualizing general purpose GPUs for high performance cloud computing: an application to a fluid simulator. In *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*, pages 863–864. IEEE, 2012.
19. A. Galletti, G. Giunta, and G. Schmid. A mathematical model of collaborative reputation systems. *International Journal of Computer Mathematics*, 89(17):2315–2332, 2012.
20. G. Giunta, R. Montella, P. Mariani, and A. Riccio. Modeling and computational issues for air/water quality problems: A grid computing approach. *Nuovo Cimento C Geophysics Space Physics C*, 28:215, 2005.
21. Giulio Giunta, Patrizio Mariani, Raffaele Montella, and Angelo Riccio. pPOM: A nested, scalable, parallel and Fortran 90 implementation of the Princeton Ocean Model. *Environmental Modelling & Software*, 22(1):117–122, 2007.
22. Giulio Giunta, Raffaele Montella, Giuseppe Agrillo, and Giuseppe Coviello. A GPGPU transparent virtualization component for high performance computing clouds. In *European Conference on Parallel Processing*, pages 379–391. Springer, 2010.
23. R. Montella, G. Giunta, and A. Riccio. Using grid computing based components in on demand environmental data delivery. In *Proceedings of the second workshop on Use of P2P, GRID and agents for the development of content networks*, pages 81–86. ACM, 2007.
24. R. Montella, D. Kelly, W. Xiong, A. Brizius, J. Elliott, R. Madduri, K. Maheshwari, C. Porter, P. Vilter, M. Wilde, et al. FACE-IT: A science gateway for food security research. *Concurrency and Computation: Practice and Experience*, 27(16):4423–4436, 2015.
25. Raffaele Montella, Giuseppe Coviello, Giulio Giunta, Giuliano Laccetti, Florin Isaila, and Javier Blas. A general-purpose virtualization service for HPC on cloud computing: an application to GPUs. *Parallel Processing and Applied Mathematics*, pages 740–749, 2012.
26. Raffaele Montella, Diana Di Luccio, Pasquale Troiano, Angelo Riccio, Alison Brizius, and Ian Foster. WaComM: A parallel Water quality Community Model for pollutant transport and dispersion operational predictions. In *Signal-Image Technology & Internet-Based Systems (SITIS), 2016 12th International Conference on*, pages 717–724. IEEE, 2016.
27. Raffaele Montella, Carmine Ferraro, Sokol Kosta, Valentina Pelliccia, and Giulio Giunta. Enabling Android-Based Devices to High-End GPGPUs. In *Algorithms and Architectures for Parallel Processing*, pages 118–125. Springer International Publishing, 2016.
28. Raffaele Montella and Ian Foster. Using hybrid grid/cloud computing technologies for environmental data elastic storage, processing, and provisioning. In *Handbook of Cloud Computing*, pages 595–618. Springer, 2010.
29. Raffaele Montella, Giulio Giunta, Giuliano Laccetti, Marco Lapegna, Carlo Palmieri, Carmine Ferraro, and Valentina Pelliccia. Virtualizing CUDA enabled GPGPUs on ARM clusters. In *Parallel Processing and Applied Mathematics*, pages 3–14. Springer International Publishing, 2016.
30. Raffaele Montella, Giulio Giunta, Giuliano Laccetti, Marco Lapegna, Carlo Palmieri, Carmine Ferraro, Valentina Pelliccia, Cheol-Ho Hong, Ivor Spence, and Dimitrios S Nikolopoulos. On the virtualization of CUDA based GPU remoting on ARM and X86 machines in the GVirtuS framework. *International Journal of Parallel Programming*, pages 1–22, 2017.
31. A Riccio, A Ciaramella, G Giunta, S Galmarini, E Solazzo, and S Potempski. On the systematic reduction of data complexity in multimodel atmospheric dispersion ensemble modeling. *Journal of Geophysical Research: Atmospheres*, 117(D5), 2012.