HYBRID CPU/GPU PLATFORM FOR HIGH PERFORMANCE COMPUTING

Michał Marks[*], Ewa Niewiadomska-Szynkiewicz[*,**]
[*] Research and Academic Computer Network (NASK)
Wawozowa 18, 02-796 Warsaw, Poland
and
[**] Institute of Control and Computation Engineering
Warsaw University of Technology
Nowowiejska 15/19, 00-665 Warsaw, Poland
Email: mmarks@nask.pl, ewan@nask.pl

KEYWORDS

HPC, cluster, GPU computing, OpenCL, cloud computing, Openstack, cryptanalysis

ABSTRACT

High performance computing is required in a number of data-intensive domains. CPU and GPU clusters are one of the most progressive branches in a field of parallel computing and data processing nowadays. Cloud computing has recently emerged as one of the buzzwords in the ICT industry. It offers suitable abstractions to manage the complexity of large data processing and analysis in various domains. This paper addresses issues associated with distributed computational system and the application of mixed GPU&CPU technology to data intensive computation. We describe a hybrid cluster formed by devices from different vendors (Intel, AMD, NVIDIA). Two variants of software environment that hides the heterogeneity of our hardware platform and provides tools for solving complex scientific and engineering problems are presented and discussed. The first solution (HGCC) is a software platform for data processing in heterogenous CPU/GPU clusters. The second solution (HGCVC) is an extension version of the previous one. The cloud technology is incorporated to the HGCC framework. The results of numerical experiments performed for parallel implementations of password recovery algorithms are presented to illustrate the performance of our systems.

INTRODUCTION

Computational-effective High Performance Computing (HPC) is required for efficient transformation of massive data into valuable information and meaningful knowledge. It is obvious that in order to support calculations for fast increasing number of data more sophisticated software and hardware platforms to perform complex operations in a scalable way have to be developed.

In recent years parallel processing has provided a new impetus in systems engineering. The intense research, development and deployment of hardware, software and applications for parallel computers were carried out. During the 1980s and 1990s, software for parallel computers focused on providing powerful mechanisms for managing communication between processors, and environments for parallel machines and computer networks. High Performance Fortran (HPF), OpenMP, Parallel Virtual Machine (PVM) and Message Passing Interface (MPI) were designed to support communications for scalable applications (Karbowski and Niewiadomska-Szynkiewicz, 2009). The application paradigms were developed to perform calculations on shared and distributed memory machines.

In the last decade, clusters, grids and clouds have been identified as important new technologies for massive data processing and large scale computing. In today's computer systems these technologies are often used to solve complex scientific problems as well as to tackle projects in industry and commerce (Marks, 2012; Wang et al., 2011; Niewiadomska-Szynkiewicz and Marks, 2012; Szynkiewicz and Błaszczyk, 2011; Koodziej et al., 2013). The most common operating systems used for building clusters are UNIX and Linux. Clusters should provide scalability, transparency, reconfigurability, availability, reliability, and high performance. There are many software tools for supporting cluster computing, such as SLURM (Yoo et al., 2003), Torque/MOAB (Staples, 2006) or ASimJava (Sikora and Niewiadomska-Szynkiewicz, 2007). A novel approach to perform parallel computations is to use a hybrid cluster – the computational architecture with multicore CPUs working together with multicore GPUs (Kunzman and Kalé, 2011; Wen-Mei, 2011). Graphics Processing Units (GPUs) al-

Proceedings 28th European Conference on Modelling and Simulation ©ECMS Flaminio Squazzoni, Fabio Baronio, Claudia Archetti, Marco Castellani (Editors)
ISBN: 978-0-9564944-8-1 / ISBN: 978-0-9564944-9-8 (CD)

low to perform massively parallel computations. Many operations are natively supported by GPU units, involving maximum instruction throughput and full use of computational resources. Using CUDA or OpenCL software platforms many applications can be easily implemented and significantly faster executed than on multiprocessor or multicore computational systems.

Initially the idea of Grid was to extend parallel computing paradigms from tightly coupled clusters to geographically distributed systems. However, in practice, Grid has been utilized more as a platform for the integration of loosely coupled applications (Berman et al., 2003; Kołodziej et al., 2014). Currently computational Grids enable the sharing and aggregation of a wide variety of geographically distributed computational resources, such as supercomputers, computer clusters, data sources, storage systems, scientific instruments, and present them as a unified, dependable resource for solving large-scale computations and data intensive computing applications. Grids have the potential to integrate as never before - theory, experiment, and computation - and to do so on a global scale.

Clouds are the natural evolution of traditional clusters and data centres (Wang et al., 2011, 2010). They are distinguished by pricing model where customers are charged based on their utilisation of computational resources, storage and transfer of data. The cloud computing emerges as a new computing paradigm that offers reliable, customized and QoS guaranteed dynamic computational environments. Clouds offer services to access hardware, software and data resources in a transparent way. The services are referred to as:

- Platform as a Service (PaaS),
- Software as a Service (SaaS),
- Infrastructure as a Service(IaaS).

Due to abilities to provide flexible computational infrastructure, configurable software services and pay-as-you-use cloud computing seems to be very promising in massive data processing and solving complex computing problems. These emerging services can reduce the cost of computation, application hosting and content storage and delivery by several orders of magnitude.

This paper addresses issues associated with distributed computational systems and the application of mixed GPU&CPU and cloud computing technologies to massively parallel computations. These technologies are very effective in solving complex calculation problems that can be divided up into large numbers of independent parts. The particular improvement should be obtained for problems that can be solved applying SPMD (Single Program Multiple Data) paradigm. The data decryption/encryption and password recovery algorithms are easy adaptable to parallel environments. Therefore, GPU-based computation performed in cluster and cloud infrastructures can be especially exploited in the field of cryptanalysis and cryptography.

The paper is organized as follows. In Section 2 we describe an architecture of our heterogenous cluster formed by two types of CPU and GPU units. The software framework for managing calculations on such type of cluster is presented in Section 3. Finally, in Section 4 we summarize results of numerical experiments. The considered case study is concerned with parallel implementation of selected cryptanalysis algorithms. The paper is concluded in Section 5.

HARDWARE INFRASTRUCTURE

The objective was to develop a computational system composed of heterogenous devices and software framework for massive parallel computations. The expected functionalities were:

- effective computation of applications implementing MapReduce programming model,
- integration of computational devices with different architectures (from different vendors) into one transparent system,
- resistance and easy to use.

Our hybrid hardware platform is composed of 24 nodes that integrates two types of multi-core CPUs and GPUs:

- 12 nodes equipped with: Intel Xeon X5650, 2.66 GHz/3.06 GHz turbo, 6 cores with Hyper-Threading technology/ 12 threads, 6x256 L2, 12 MB L3 cache, and NVIDIA Tesla M2050, 448 CUDA cores, 384-bit memory bus.
- 12 nodes equipped with: AMD Opteron 6172, 2.1 GHz, 12 cores / 12 threads, 12x512 KB L2, 12 MB L3 cache, and AMD FirePro V7800, 1440 stream processors (equivalent of 288 CUDA cores), 256-bit memory bus.

The system architecture is presented in Fig. 1. All computational nodes equipped with CPU and GPU devices are supported by a dedicated master and storage nodes providing access to disk arrays and management capabilities. Communication in the cluster presented in Fig. 1 is organized using different interconnects: InfiniBand 4x QDR, 10 GbE and 1 GbE. Such excess network configuration allows us to separate communication connected

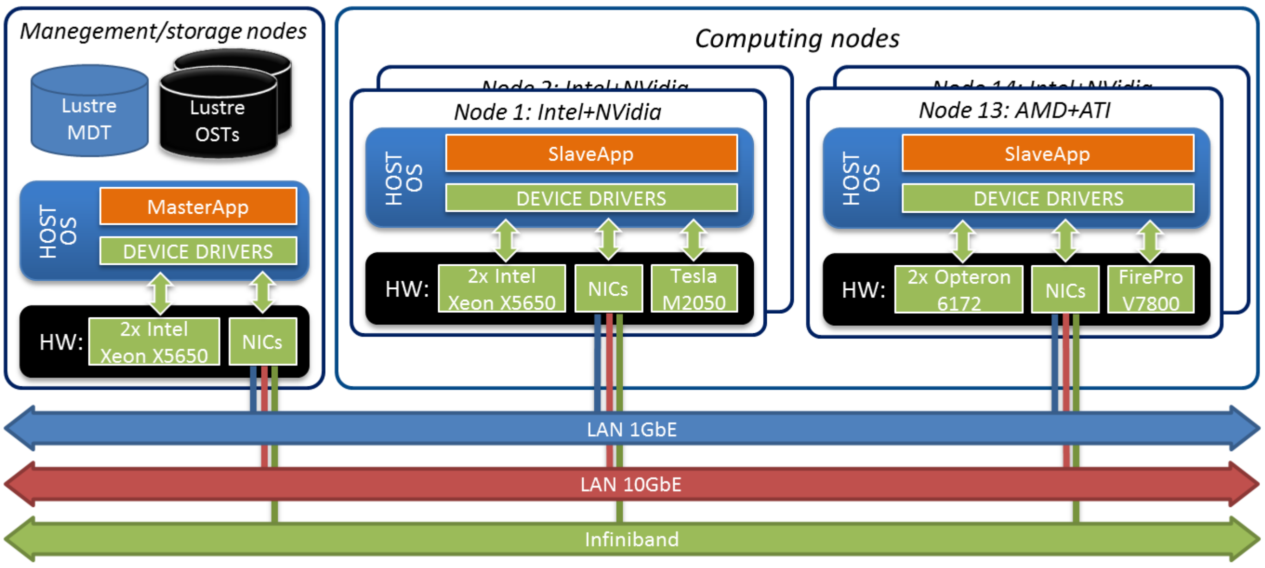

Figure 1: Hardware platform (Intel CPU + NVIDIA GPU and AMD CPU + AMD GPU nodes).

with IO operations from computational traffic. The current configuration assumes utilizing 10 GbE network for providing access to data storage. InfiniBand and the 1 GbE Ethernet are used for computational purposes.

SOFTWARE INFRASTRUCTURE

The novelty of our solution is not only the proposed hybrid architecture of our hardware platform but new software framework that can support the potential user in a task execution. The objective of this framework is to provide environments for parallel calculations that are performed on a hardware platform formed by heterogenous CPU and GPU devices. The main functionality is to hide a heterogeneity of the computational nodes and minimize user's effort during the design, implementation and execution of the applications. We developed and implemented the software framework HGCC (Hybrid GPU/CPU Cluster) for management parallel calculations that can be performed on the hardware platform presented in Fig. 1. Moreover, this framework can be executed in virtual environment. HGCVC (Hybrid GPU/CPU Virtualized Cluster) is an extended version of HGCC utilizing virtualization and cloud computing. Application of Openstack orchestrator increases flexibility, functionality and robustness of HGCC software.

HGCC Framework

When designing the HGCC system we have assumed that from the potential user's perspective, the computational system should serve as one supercomputer. The concept was to allow applications developed by users to transparently utilize many CPU and GPU devices, as if all the devices were on the local computer. A single system image model was implemented. Finally, in our framework all servers' resources such as CPU, GPU or memory are seen by the user as one unique machine. In order to take advantage of GPUs from different vendors, we decided to use OpenCL (Bainville, 2010). It is a low level programming toolkit for writing programs that execute across heterogeneous platforms consisting of CPUs, GPUs, DSPs (Digital Signal Processors), FPGAs (Field-Programmable Gate Arrays) and other processors.

In general, the goal of the HGCC framework is to divide the data into separate domains, allocate the calculation processes to cluster nodes, manage calculations and communication, and provide a store for all data objects. Hence, four groups of services are supplied: user interface services, calculation management services, communication services and data repository services.

HGCC is composed of several components. The most important are `MasterApp` (master node application) and `SlaveApp` (computational node application), Fig. 1. `MasterApp` is the main component that is responsible for the user-system communication and calculation management. `SlaveApp` is responsible for calculations that are performed by the assigned server.

It is assumed that each computational node in HGCC contains some number of resources. Two types of such resources are distinguished: CPUs and GPUs. The computational resource can be in one of three states:

- `waiting` – ready for loading a new task to execution,
- `working` – occupied, calculations are executed,
- `lost` – lost because of the node failure.

Our framework implements *master-slave* communication model. An XML-based communication protocol based on the TCP/IP protocol and BSD sockets is used to perform communication between master and slave nodes.

Each computational task should be defined in the `task descriptor` and implemented in an object oriented style. The XML Schema specification for building XML files with task description is provided in HGCC. The task descriptor contains: a type of the task, an algorithm, a destination platform and device. All these parameters are mandatory. Rest of this file is filled by parameters specific to a given task.

HGCC operates as follows. A computational task implemented by the user is sent to the `MasterApp` component. All parameters defined in the `task descriptor` are parsed inside `MasterApp`. The task is divided into smaller subtasks which are allocated to the slave nodes with free resources. Next the `SlaveApp` application is initialized. A plugin list is loaded from a plugin descriptor file, and a socket is opened for `MasteApp`'s connection. The plugin descriptor file contains information about all plugins currently available in the system. Whenever a slave node gets a new set of subtasks to execute it looks for available valid plugin, and loads it to the memory. Next, the control flow inside `SlaveApp` splits, and the newly spawned thread launches calculations stored in the loaded plugin.

HGCVC Framework

The HGCC framework is prepared to be executed in form of `MasterApp` and `SlaveApp` daemons working directly in the operating system. This solution is quite effective and comfortable when computational resources are dedicated to perform tasks supported by HGCC. However, in practice there are many situations with the need of assigning computational power to perform calculations on behalf of other projects. These situations often imply the necessity of preparing different environment and as a consequence it may lead to changes disrupting normal HGCC operations.

In order to overcome these problems we decided to prepare an extended solution utilizing the power of virtualization and cloud computing. The new environment is called HGCVC - Hybrid CPU/GPU Virtualized Cluster and is based on KVM Hipervisor and Openstack orchestrator. The comparison of computational node architectures for HGCC and HGCVC is presented in Figure 2. In case of HGCC framework the `SlaveApp` is run by the host operating system (the only one available operating system) using hardware available through devices drivers. In case of HGCVC the `SlaveApp` is run by the guest operating system hosted on one of the Virtual Machines. The virtual machines are run by the hypervisor which provides the set of devices modules – a hardware abstraction. This property which is very useful in majority of cloud systems applications is a drawback in case of utilizing cloud computing for HPC. That's why in HGCVC solution the PCI pass-through property is exploited to provide a direct access to GPUs for guest operating systems (one guest OS in the same time). This solution allows us to minimize the overhead caused by virtualization and, in the same, allows us to get cloud benefits like scalability, reliability and utility of cluster infrastructure.

The whole process of orchestration and system management is organized using Openstack modules. In Fig. 2 only cloud agent (Openstack nova compute module) is presented – as this is the only module which needs to be run on computing nodes. The rest of Openstack functionality like network, storage, identity and images management is provided by the controller node which plays for Openstack a similar role like `MasterApp` for HGCC software.

CASE STUDY RESULTS: PASSWORD RECOVERY

However, the presented computational system composed of CPU and GPU devices can be used to any massively parallel and intensive-data computations we used it to develop efficient cryptanalysis and cryptography. The results of evaluations of selected encryption and decryption algorithms (DES, 3DES, AES) are described in (Niewiadomska-Szynkiewicz et al., 2012). In this paper we present and discuss the efficiency of parallel implementations of selected password recovery algorithms. The only reasonable technique for recovering a password from hash is to scan all potential password, compute their hash, and test the coincidence (Paar et al., 2010; Li et al., 2009). In general, cryptographic hash functions include integer and binary operations such as: addition modulo power of two, bit shift and rotation, bitwise xor, bitwise or, bit negation and words permutation. All those operations are natively supported by GPU processors. Three main approaches to password strength validation that are usually considered are: brute-force, rainbow-table and

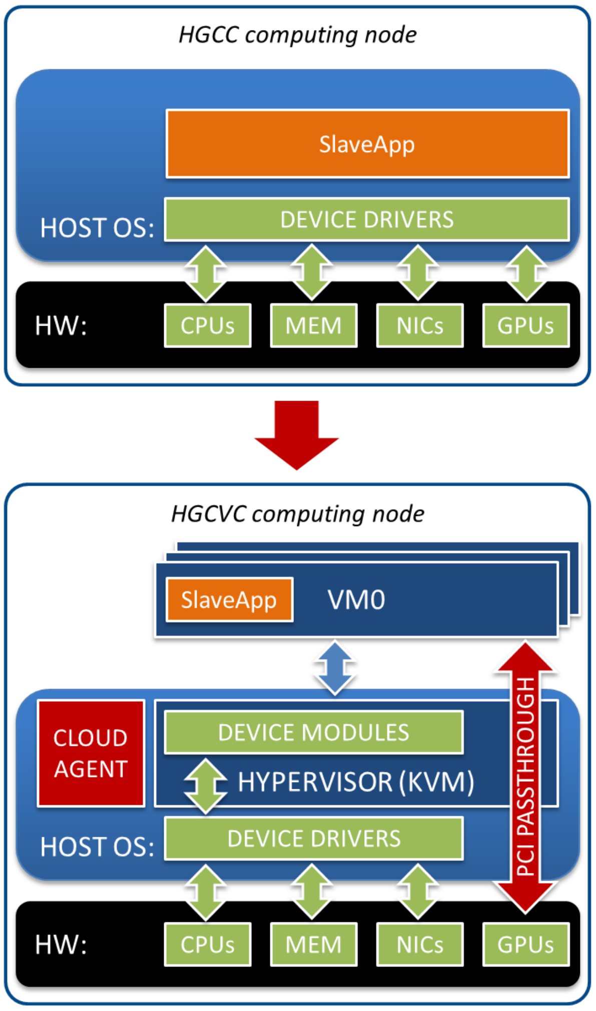

Figure 2: HGCC and HGCVC computational nodes.

dictionary test. The first two of these are very suitable for porting to GPU. Brute-force, thus not very sophisticated, is the most common approach used together with GPUs.

Multiple tests were performed for parallel implementations of password recovery from MD5, SHA-1 and SHA-2 hashes utilizing brute-force technique, and hybrid CPU&GPU computing. The aim of the first series of tests was to compare the performance achieved by Intel Xeon and AMD Opteron central processing units. Tables 1 - 3 present scalability of the CPU devices. Tables 1 and 2 collect the number of hashes generated per second using multi-threaded versions of respectively, MD5 and SHA-1 algorithms. The parameter *th_num* in all tables denotes the number of executed threads. The best results were obtained for Intel Xeon X5650 both in case of MD5 and SHA-1. In most tests two Intel Xeon processors with

th_num	1	2	4	8	16	24
AMD	20.1	39.8	79.6	157.7	315.2	465.4
Intel	37.8	74.5	147.9	284.2	384.7	407.5

Table 1: Number of generated MD5 hashes per second (in millions).

th_num	1	2	4	8	16	24
AMD	5.0	9.9	19.7	39.4	78.7	117.5
Intel	10.1	19.9	39.5	76.1	119.5	147.8

Table 2: Number of generated SHA-1 hashes per second (in millions).

6 cores each plus Hyper-Threading technology generated more hashes than two AMD Opteron processors with 12 cores each. Next, the scalability of both CPU technologies was compared. The results are presented in Table 3. It can be observed that both processors scale up very well with a certain predominance of the AMD device.

th_num	2	4	8	12	16	24
AMD (MD5)	2.0	4.0	7.9	11.8	15.7	23.2
Intel (MD5)	2.0	3.9	7.5	9.8	10.2	10.8
AMD (SHA-1)	2.0	4.0	7.9	11.8	15.8	23.5
Intel (SHA-1)	2.0	3.9	7.5	10.8	11.8	14.6

Table 3: Scalability of AMD Opteron 6172 and Intel Xeon X5650 processing units for MD5 and SHA-1 algorithms.

The goal of the second series of experiments was to compare the performance of CPU-based and GPU-based algorithms for password recovery. Three techniques for hash generation were considered: MD5, SHA-1 and four versions of SHA-2 (a-224 bit, b-256 bit, c-384 bit, d-512 bit). The number of hashes generated per second running MD5, SHA-1 and SHA-2 algorithms on Intel Opteron and NVIDIA Tesla processors are presented in Fig. 3. The results of the same tests performed on AMD Opteron and AMD FirePro processors are depicted in Fig. 4. As it can be seen in figures 3 and 4 the GPU-based implementations give better results than the CPU-based ones. In general, the best results were obtained for the OpenCL versions of the hash functions executed on AMD FirePro V7800.

Finally, we tested the scalability of the parallel implementations of AMD5, SHA-1 and SHA-2 algorithms in the cluster. The aim was to present the efficiency of our hybrid computational system. The results, i.e., number of generated hashes per second are collected in Table 4. We present the results for subclusters composed of Intel Xeon, AMD Opteron, NVIDIA Tesla and AMD FirePro

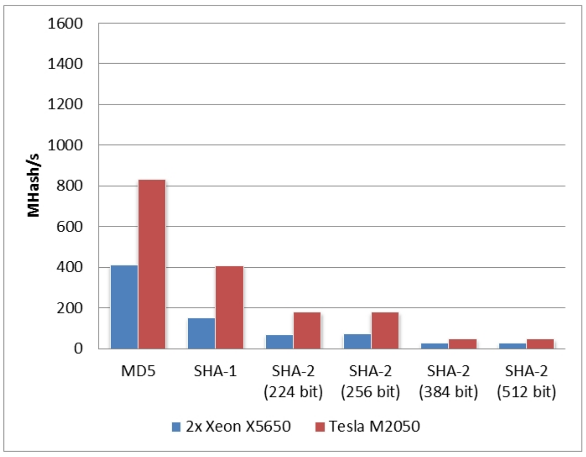

Figure 3: Number of generated hashes per second (in millions); Intel Xeon & NVIDIA Tesla.

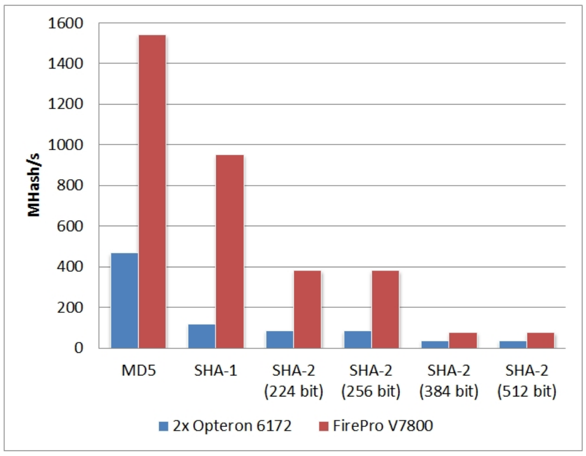

Figure 4: Number of generated hashes per second (in millions); AMD OPteron & AMD FirePro.

processing units. The last column contains the total results for the whole cluster. The results presented in Table

Algorithm	AMD Opteron	Intel Xeon	AMD FirePro	NVIDIA Tesla	Total
MD5	5518	4792	16599	10024	36933
SHA-1	1340	1750	11206	4952	19248
SHA-2a	945	791	4571	2094	8401
SHA-2b	940	793	4548	2094	8375
SHA-2c	417	297	859	532	2105
SHA-2d	410	298	870	529	2107

Table 4: Number of generated MD5, SHA-1 and SHA-2 hashes per second (in millions); the whole cluster.

4 show that our parallel implementations of password recovery algorithms scales up very well in the cluster composed of CPU and GPU devices.

SUMMARY AND CONCLUSION

In this paper we presented the short overview of the hybrid computational platform integrating CPU and GPU technologies. We focused on two variants of the software framework that integrate computational devices with different architectures into one transparent system. Our system has already proved to be very useful for massively parallel computing. The experimental results presented in the paper demonstrate its effectiveness and scalability. As a final observation we can say that cryptanalysis algorithms are natural candidates for massively parallel computing in computational platforms integrating CPU and GPU units. In our future research we plan to use HGCC and HGCVC to broad range of problems requiring large data processing.

REFERENCES

Bainville, E. (2010). Opencl multiprecision tutorial @ONLINE.

Berman, F., Hey, A., and Fox, G. (2003). *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley & Sons, Inc., New York, NY, USA.

Karbowski, A. and Niewiadomska-Szynkiewicz, E. (2009). *Parallel and distributed computing (in Polish)*. WUT Publishing House.

Kołodziej, J., Khan, S., Wang, L., Kisiel-Dorohinicki, M., Madani, S., Niewiadomska-Szynkiewicz, E., Zomaya, A., and Xu, C.-Z. (2014). Security, energy, and performance-aware resource allocation mechanisms for computational grids. *Future Generation Computer Systems*, 31:77–92.

Koodziej, J., Khan, S. U., and Talbi, E.-G. (2013). Scalable optimization in grid, cloud, and intelligent network computing foreword. *Concurrency and Computation: Practice and Experience*, 25(12):1719–1721.

Kunzman, D. M. and Kalé, L. V. (2011). Programming heterogeneous clusters with accelerators using object-based programming. *Sci. Program.*, 19:47–62.

Li, C., Wu, H., Chen, S., Li, X., and Guo, D. (2009). Efficient implementation for md5-rc4 encryption using gpu with cuda. In *Anti-counterfeiting, Security, and Identification in Communication, 2009. ASID 2009. 3rd International Conference on*, pages 167–170. IEEE.

Marks, M. (2012). Enhancing wsn localization robustness utilizing hpc environment. In *Proc. of the European Conference on Modelling and Simulation (ECMS 2012)*, pages 167–170. European Council for Modelling and Simulation.

Niewiadomska-Szynkiewicz, E. and Marks, M. (2012). Software environment for parallel optimization of complex systems. In *Applied Parallel Scientific Computing K. Jonasson*, volume LNCS 7133, pages 86–96. Springer-Verlag.

Niewiadomska-Szynkiewicz, E., Marks, M., Jantura, J., and Podbielski, M. (2012). A hybrid cpu/gpu cluster for encryption and decryption of large amounts of data. *Journal of Telecommunications and Information Technology*, 3(2):55–64.

Paar, C., Pelzl, J., and Preneel, B. (2010). *Understanding Cryptography: A Textbook for Students and Practitioners*. Springer.

Sikora, A. and Niewiadomska-Szynkiewicz, E. (2007). A federated approach to parallel and distributed simulation of complex systems. *International Journal of Applied Mathematics and Computer Science ACMS*, 17(1):99–106.

Staples, G. (2006). Torque resource manager. In *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, SC '06, New York, NY, USA. ACM.

Szynkiewicz, W. and Błaszczyk, J. (2011). Optimization-based approach to path planning for closed chain robot systems. *International Journal of Applied Mathematics and Computer Science ACMS*, 21(4):659–670.

Wang, L., Ranjan, R., and Chen, J.and Beanatallah, B. e., editors (2011). *Cloud computing methodology, system, and applications*. CRS Press, Taylor & Francis.

Wang, L., von Laszewski, G., Younge, A., He, X., Kunze, M., Tao, M., and Fu, C. (2010). Cloud computing a perspective study. *New Generation Comput.*, 28(2):137–146.

Wen-Mei, W., H. (2011). *GPU Computing Gems Emerald Edition*. Morgan Kaufman.

Yoo, A., Jette, M., and Grondona, M. (2003). Slurm: Simple linux utility for resource management. In Feitelson, D., Rudolph, L., and Schwiegelshohn, U., editors, *Job Scheduling Strategies for Parallel Processing*, volume 2862 of *Lecture Notes in Computer Science*, pages 44–60. Springer Berlin Heidelberg.

AUTHOR BIOGRAPHIES

MICHAŁ MARKS received his M.Sc. in computer science from the Warsaw University of Technology, Poland, in 2007. Currently he is a Ph.D. student in the Institute of Control and Computation Engineering at the Warsaw University of Technology. Since 2007 with Research and Academic Computer Network (NASK). His research area focuses on wireless sensor networks, global optimization, distributed computation in CPU and GPU clusters, decision support and machine learning. His e-mail is `mmarks@nask.pl`

EWA NIEWIADOMSKA-SZYNKIEWICZ DSc (2005), PhD (1995), professor of control and information engineering at the Warsaw University of Technology, head of the Complex Systems Group. She is also the Director for Research of Research and Academic Computer Network (NASK). The author and co-author of 3 books and over 140 papers. Her research interests focus on complex systems modeling, optimization, control and simulation, parallel computation and computer networks. Her email is `ewan@nask.pl`.