Paper

A Hybrid CPU/GPU Cluster for Encryption and Decryption of Large Amounts of Data

Ewa Niewiadomska-Szynkiewicz[a,b], Michał Marks[a,b], Jarosław Jantura[b], and Mikołaj Podbielski[b]

[a] *Institute of Control and Computation Engineering, Warsaw University of Technology, Warsaw, Poland*
[b] *Research and Academic Computer Network (NASK), Warsaw, Poland*

Abstract—The main advantage of a distributed computing system over standalone computer is an ability to share the workload between cores, processors and computers. In our paper we present a hybrid cluster system – a novel computing architecture with multi-core CPUs working together with many-core GPUs. It integrates two types of CPU, i.e., Intel and AMD processor with advanced graphics processing units, adequately, Nvidia Tesla and AMD FirePro (formerly ATI). Our CPU/GPU cluster is dedicated to perform massive parallel computations which is a common approach in cryptanalysis and cryptography. The efficiency of parallel implementations of selected data encryption and decryption algorithms are presented to illustrate the performance of our system.

Keywords—AES, computer clusters, cryptography, DES, GPU computing, parallel calculation, software systems.

1. Introduction

Data encryption and decryption are generally complex problems and involve cumbersome calculations, especially when consider processing of large amounts of data. The restrictions are caused by demands on computer resources, i.e., processor and memory. However, in many cases the calculations performed by cryptography algorithms can be easily partitioned into large number of independent parts and carried out on different cores, processors or computers. It was observed that parallel implementation based on MapReduce programming model improves the efficiency of the algorithm and speeds up a calculation process.

CPU and GPU clusters are one of the most progressive branches in a field of parallel computing and data processing nowadays, [1], [2]. A cluster is a set of computers working together so that in many aspects they can be viewed as a single system. Typical cluster consists of homogenous Central Processing Units (CPUs). A new model for parallel computing based on using CPUs and GPUs (Graphics Processing Units) together to perform a general purpose scientific and engineering computing was developed in the last years, and used to solve complex scientific and engineering problems. Using CUDA or OpenCL programming toolkits many real-world applications can be easily implemented and run significantly faster than on multi-processor or multi-core systems [3].

In this paper we describe and evaluate a hybrid cluster system HGCC that integrates two types of multi-core CPUs, i.e., Intel and AMD processors equipped, adequately, with NVIDIA and AMD graphical units. We have designed and developed a dedicated software framework for parallel execution of computing tasks which aim is to hide a heterogeneity of the cluster – from the user's perspective, the cluster system serves as one server. The objective of this software is to divide the data into separate domains, allocate the calculation processes to cluster nodes, manage calculations and communication.

The remainder of this paper is organized as follows. We present a brief survey of modern GPU clusters in Section 2. The architecture of our cluster and software framework that manages calculations are described in Section 3. Finally, in Section 4 we briefly summarize results of tests for selected types of data encryption and decryption algorithms. The paper concludes in Section 5.

2. Survey of CPU and GPU Clusters

Every year in June and November the TOP500 list is published. The announcement of the list is not only the chance to find out what are the most powerful supercomputers but also a great opportunity to observe new trends in the HPC technologies. In June 2012, as compared to November 2011 list, when there was no turnover in the Top10, this time around, there are six brand new machines, plus one, Jaguar, that has benefitted from an upgrade to faster processors. The majority of these new machines is built using latest IBM solution called Blue Gene Q. Only one of new supercomputers is equipped with GPU. However it doesn't mean that the interest in applying GPU technology in supercomputers is falling down. Many of the most powerful supercomputer centers are waiting for new accelerators from NVIDIA, AMD and Intel. For example a new supercomputer Titan, which will be a successor of Jaguar and is currently being built in Oak Ridge National Laboratory, will be equipped with almost 15 000 of NVIDIA cards from "Keppler" family.

When we look at the whole Top500 list we can observe a rising significance of GPU accelerators. In June 2012 ranking, there are 58 machines that are equipped with GPU

accelerators, up from 17 only one year ago – see Fig. 1. It is worth noting that 53 of them use NVIDIA Tesla GPU co-processors while only two of them are equipped with IBM's Cell coprocessors and other two with AMD's Radeon cards. Moreover, a new product of Intel utilizing Intel MIC accelerator had its debut on TOP500 in an experimental cluster with pre-production Knights Corner chips. MIC chip will be available in the end of 2012, as Intel Xeon Phi coprocessor.

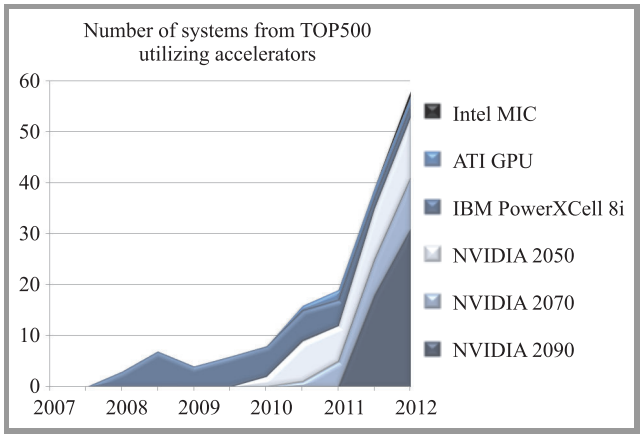

Fig. 1. Number of systems from TOP500 utilizing accelerators.

The most common operating systems used for building clusters are UNIX and Linux. Clusters should provide following features: scalability, transparency, reconfigurability, availability, reliability and high performance. There are many software tools for supporting cluster computing. In the beginning of XXI century, the common idea was to provide a view of one supercomputer for a cluster built from a group of independent workstations. The SSI (Single System Image) clusters were designed and developed. In this approach all servers' resources such as disks, memory, processors are seen by a user as one unique machine. The whole cluster is identified from outside by one IP address. The popular systems that implement the idea of SSI are Mosix (http://www.mosix.org) that does not cover all SSI features, and two comprehensive clustering solutions offering full SSI environments: OpenSSI (http://openssi.org) and Kerrighed (http://www.kerrighed.org). A brief overview and comparative study of stability, performance and efficiency of Mosix, OpenSSI and Kerrighed systems is presented in literature [4].

Other commonly used systems that can be applied to high performance data processing and calculations in cluster systems are software frameworks that perform job scheduling. Commonly used Portable Batch System PBS (www.pbsworks.com) provides mechanisms for allocating computational tasks to available computing resources. Various versions of the system are available, open source and commercial: OpenPBS, MOAB with Torque, PBS Professional.

Most of the presented cluster systems are mature solutions. However, they have some limitations. Mosix, OpenSSI and Kerrighed systems focuss on load balancing. The idea is to implement an efficient load balancing algorithm which is triggered when loads of nodes are not balanced or local resources are limited. In general, processes are moved from higher to less loaded nodes. Unfortunately, migration of processes involves extra time for load calculation and overhead in communication. Moreover, Mosix, OpenSSI, Kerrighed systems were designed for CPU clusters.

Currently, users are provided with software environments that allow to perform calculations on a single GPU device. There are only a few software tools for running applications on GPU clusters. Virtual OpenCL VCL (www.mosix.org/txt_vcl.html) is a software platform for GPU clusters. It can run unmodified OpenCL applications on Linux clusters with or without the Mosix system. VCL provides a view of one superserver for cluster built from a group of GPU units. The components of VCL, its performance and applications are presented in [5].

Our goal was to develop a software framework that allows unmodified OpenCL applications to transparently and concurrently run on multiple CPU and GPU devices in a cluster. In case of our application we need a simple functionality, i.e., a calculation speed up, resistance and ease of use. We perform static decomposition of the problem in calculation startup, hence the dynamic load balancing is superfluous. Our software framework is quite similar to VCL platform [5], however, in our solution it is possible to utilize both CPUs and GPUs on computational nodes.

3. HGCC System Overview

3.1. Hardware Components of HGCC

The aim of our work on utilizing a cluster composed of CPUs and GPUs in cryptography and complex data analysis is providing the system which functionality allow us to perform: effective computing of applications implementing MapReduce programming model, comparison of performance of CPUs and GPUs from many vendors along with comparison of different interconnects performance.

We have built a heterogenous cluster system with multi-core CPUs working together with many-core GPUs. The system consists of 24 nodes and integrates two types of CPUs: 12 servers with two Intel Xeon processors each and 12 servers with two AMD Opteron processors each. All servers are equipped with advanced GPUs, adequately, NVIDIA Tesla and AMD FirePro units. It is worth to note that FirePro V7800 has a peak performance in the single precision almost two times better than NVIDIA Tesla M2050, and a peak performance in the double precision equal 0.8 of Tesla's performance. Moreover, AMD GPU is approximately four times cheaper than NVIDIA GPU. However, effective programming of GPU based on Very Long Instruction Word (VLIW5) architecture – AMD FirePro V7800 – is not a simple task and not every application is capable to achieve performance close to the peak one.

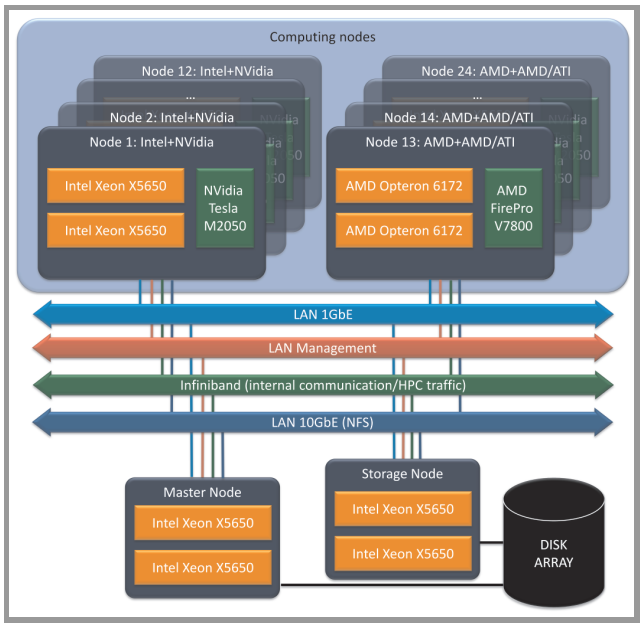

Fig. 2. Hybrid system architecture with Intel+NVIDIA and AMD+ATI/AMD nodes.

The system architecture is depicted in Fig. 2. The specification of components that form the HGCC cluster is as follows:

CPU Intel : Intel Xeon X5650, 2.66 GHz/3.06 GHz turbo, 6 cores / 12 threads, 6x256 L2, 12 MB L3 cache.

CPU AMD : AMD Opteron 6172, 2.1 GHz, 12 cores / 12 threads, 12x512 KB L2, 12MB L3 cache.

GPU NVIDIA : NVIDIA Tesla M2050, 448 CUDA cores, 384-bit memory bus.

GPU AMD : AMD FirePro V7800, 1440 stream processors (equivalent of 288 CUDA cores), 256-bit memory bus.

The computing nodes are supported by a dedicated master and storage nodes providing access to disk arrays and management capabilities. Communication between nodes is organized using different interconnects: InfiniBand 4x QDR, 10GbE and 1GbE. Such redundant network configuration allows us to verify the impact of selected interconnects on computation efficiency. Moreover, it is possible to separate communication connected with IO operations from computational traffic. The current configuration assumes utilizing InfiniBand network for providing access to data storage. 10GbE Ethernet and the 1GbE Ethernet are used for computational purposes.

3.2. HGCC Software Framework

The HGCC software framework provides an environment for parallel calculations that are performed on a cluster formed by heterogenous CPU and GPU devices. The goal was to hide a heterogeneity of the cluster and minimize the user's effort during the design, implementation and execution of the application. From a user's perspective, the cluster system should serve as one server. So, it allows a user to focus only on the numerical part of his application. 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 is implemented – all servers' resources such as CPU, GPU or memory are seen by the user as one unique machine. Therefore, applications written for HGCC benefit from the reduced programming complexity of a single computer, the availability of shared memory and multi threads, as in OpenMP (http://openmp.org/wp), and a concurrent access to cluster nodes and their devices, as in MPI (http://mpi-forum.org).

In order to take advantage of GPU accelerators from different vendors, we decided to use OpenCL, which is a low level GPU programming toolkit, where developers write GPU kernels entirely by themselves with no automatic code generation [6]. OpenCL is an industry standard computing library developed in 2009 that targets not only GPUs, but also CPUs and potentially other types of accelerator hardware. In OpenCL, an efficient implementation requires preparation slightly different codes for different devices, however, it is much less complicated than writing code in many native toolkits for NVIDIA and AMD devices.

The facilities of the HGCC system are provided in the form of four groups of services. These are: user interface, calculation management, communication services and data repository services. *User interface services* provide a consistent user interface supporting the process of defining an application, processing of the calculation results and providing communication with the user. *Calculation management services* allocate the calculation processes into cluster nodes and manage execution of the user's application. *Communication services* manage communications between running processes and system kernel, and finally *data repository services* provide a store for all data objects.

HGCC Architecture. The cluster framework consists of several components presented in Fig. 3. The most important are: *MasterApp* – master node application, the main component that is responsible for the user-system communication and calculation management and *SlaveApp* – the computational node application, the component that is responsible for calculations that are performed by the assigned server. Each computational node contains some number of resources. In our framework we distinguish and collect information about two types of such resources: CPUs – central processing units and GPUs – graphics processing units. The computational resource can be in one of the following states: *waiting* – ready for loading a new task to execution, *working* – occupied, calculations are executed and *lost* – lost because of the node failure.

Inter-process Communication. The system implements the master-slave communication scheme. An XML-based communication protocol based on the TCP/IP protocol and BSD sockets is used to perform communication between

master and slave nodes. Our goal was to develop a simple, flexible and failure resistance mechanism.

HGCC System Operation. A user implements the computational task in an object oriented way and defines his problem in the *task descriptor*. 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. The rest of this file is filled by parameters specific to a given task. The cluster framework can handle any computational task which was implemented by the user. A committed task is sent to the *MasterApp* component. All parameters defined in the *task descriptor* are parsed inside *MasterApp*. Next, the task is divided into smaller subtasks. *MasterApp* creates the list of such subtasks. They are allocated to the slave nodes, which contain any free resources. Two operations are performed after *SlaveApp* initialization: a plugin list is loaded from a plugin descriptor file, and a socket is opened and wait 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.

4. Case Study Results: Parallel Cryptography

Cryptanalysis and cryptography techniques are natural candidates for massively parallel computations. The algorithms for encryption and decryption of large amounts of data can be easily decomposed and executed in parallel. The popular schemes using symmetric ciphers were found to give a significant speed up when ported to GPU, especially such schemes like Data Encryption Standard (DES), Advanced Encryption Standard (AES) [7]–[10] or BlowFish [11]–[13]. GPU-based implementations of algorithms using asymmetric ciphers (RSA, ECC, NTRU and GGH) are described in the following papers [14]–[18]. In this paper we present the evaluation of the performance of our HGCC-based implementations of symmetric DES and AES cryptography algorithms.

4.1. DES and AES Implementations in HGCC

The HGCC cluster is the general purpose hardware and software system that can be used to solve any complex computing problems that require a processing of large amounts of data (see [19], [20]). In our research, which results are presented in this paper we used HGCC to efficient cryptanalysis and cryptography. The evaluation of selected techniques of cryptanalysis, i.e., the password recovery from hashes are described in [21]. In this paper we focus on effective cryptography working on CPU and GPU units. The numerical results of extensive tests of our implementations of DES and AES algorithms are presented and discussed.

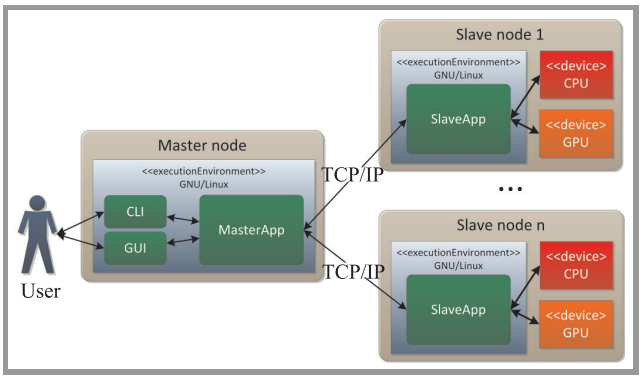

Fig. 3. Core components of the cluster framework.

We performed four series of experiments. The aim of the first series was to test the efficiency of parallel implementations of the DES and AES algorithms on the cluster formed only by the CPU units. Various modes of the algorithms operation were compared. The current modes of operation listed in Table 1 are specified in http://csrc.nist.gov/index.html and in [22]. The simplest of the encryption modes is the electronic codebook (ECB) mode. The message is divided into blocks and each block is encrypted separately. In CBC (cipher-block chaining) mode, each block of plain text is XORed with the previous cipher text block before being encrypted. This way, each cipher text block depends on all plain text blocks processed up to that point. The PCBC (propagating cipher-block chaining) mode was developed to cause small changes in the ciphertext to propagate indefinitely both when decrypting and encrypting. The CFB (cipher feedback) mode that is relative to CBC makes a block cipher into a self-synchronizing stream cipher. The OFB (output feedback) mode makes a block cipher into a synchronous stream cipher. The CTR (counter) mode has similar characteristics to OFB, but also allows a random access property during decryption. It should be pointed that several of listed modes are suited to parallel implementation (see Table 1).

Table 1
Modes of operations of symmetric ciphers

Mode	Parallel encryption	Parallel decryption
ECB	Yes	Yes
CBC	No	Yes
PCBC	No	No
CFB	No	Yes
OFB	No	No
CTR	Yes	Yes

Table 2 and Fig. 4 demonstrate the performance of the single thread CPU-based implementations of the block ciphers: DES, 3DES and AES. The table collects the

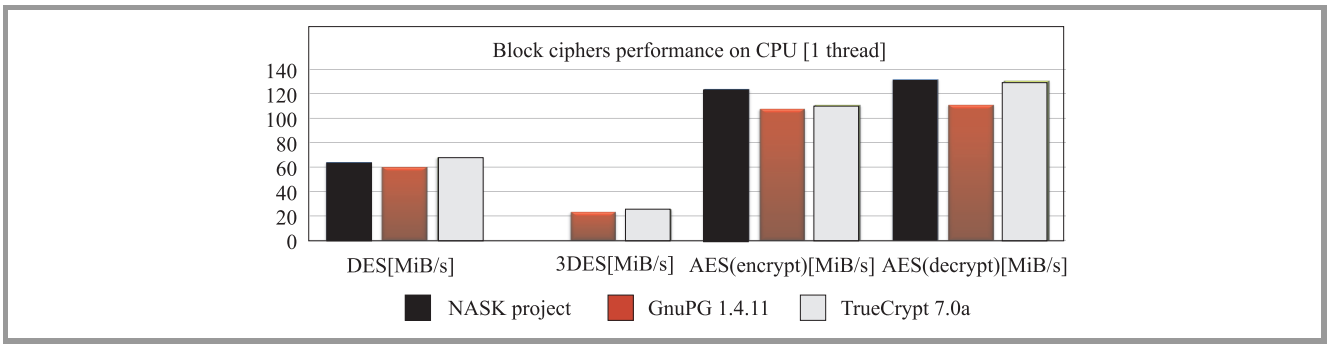

Fig. 4. Block ciphers performance on CPU (1 thread).

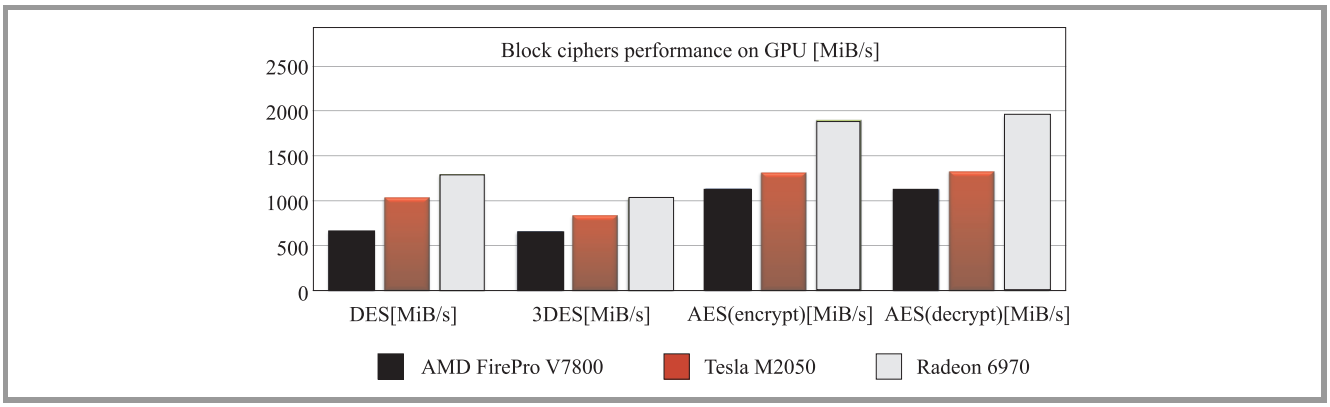

Fig. 5. Block ciphers performance on GPU.

amounts of data in MiB/s (MiB/s = 1,048,576 bytes/s) that were encrypted/decrypted per second in case of all tested algorithms. The efficiency of our implementations of DES and AES (NASK) were compared with the results presented in the Internet: GNU Privacy Guard GnuPG 1.4.11 (http://www.gnupg.org/download/) and free open-source disk encryption software TrueCrypt 7.0a (http://www.truecrypt.org/).

Table 2
Block ciphers performance on CPU (1 thread) in MiB/s

Algorithm	Implementation		
	NASK	GnuPG 1.4.11	TrueCrypt 7.0a
DES	63.82	60.01	67.95
3DES		23.10	25.48
AES(encrypt)	123.70	107.50	110.60
AES(decrypt)	131.80	110.80	130.70

The presented results show that the efficiency of our implementations of DES and AES on CPU is similar to the results provided by other projects. It is worth to mention that both GnuPG and TrueCrypt are widely used products of teams that have extensive experience in cryptography.

The aim of the second series of experiments was to compare the efficiency of the DES and AES implementations on different GPUs provided by various vendors. The calculations were carried out on three types of GPU units: AMD FirePro V7800, NVIDIA Tesla M2050 and AMD Radeon 6970. Table 3 and Fig. 5 present the amounts of data in MiB/s that were encrypted/decrypted per second in case of all tested algorithms and decryption and encryption operations.

Table 3
Block ciphers performance on GPU in MiB/s

Algorithm	Graphics Processing Unit		
	AMD FirePro V7800	NVIDIA Tesla M2050	AMD Radeon 6970
DES	660.73	1038.02	1295.60
3DES	658.11	837.67	1031.73
AES(encrypt)	1135.96	1316.82	1901.25
AES(decrypt)	1129.54	1329.62	1963.91

In general, the GPU-based implementations of all algorithms were much more efficient than implementations working only on the CPU unit. The best results were obtained for the Radeon 6970 graphics processing unit.

The aim of the next series of experiments was to compare the performance of two implementations of the AES algorithm. The original AES implementation was compared with implementation utilizing an Advanced Encryption Standard – New Instruction Set (AES-NI) extension. New Instruction Set is an extension to the x86 instruc-

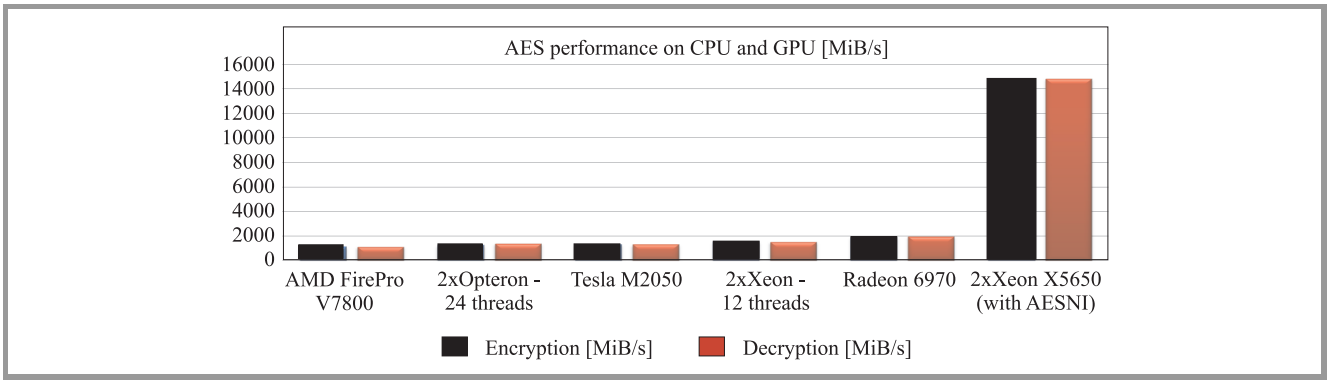

Fig. 6. Performance of the AES algorithm implementations with/without AES-NI extension; CPU and GPU.

tion set architecture for microprocessors from Intel and AMD (http://ark.intel.com/). The purpose of this instruction set is to improve the speed of applications performing encryption and decryption using AES, which is an industrial standard nowadays. In our cluster only Intel Xeon X5650 Westmere processors provides AES-NI extension. Unfortunately none of our processors provides support for Advanced Vector Extensions (AVX), so we were not able to assess impact of AVX instruction set on AES performance. Table 4 and Fig. 6 demonstrate the performance of AES executed on three types of GPU and two types of CPU units. We can see that the application of the new instruction set causes massive acceleration of the AES algorithm.

Table 4
Performance of the AES algorithm implementations with/without AES-NI extension; CPU and GPU

Processing Unit	AES-NI extension	Encryption [MiB/s]	Decryption [MiB/s]
AMD FirePro V7800	no	1135.96	1129.54
NVIDIA Tesla M2050	no	1316.82	1329.62
AMD Radeon 6970	no	1901.25	1963.91
2xOpteron – 24 threads	no	1272.00	1394.40
2xXeon – 12 threads	no	1546.80	1533.60
2xXeon – 12 threads	yes	14848.80	14841.60

Table 5
Scalability of the AES algorithm

Processor	Speedup		
	1 node	2 nodes	4 nodes
Xeon X5650	1	1.95	3.71
Opteron 6172	1	1.93	3.73

The aim of the last series of experiments was to present the efficiency of our cluster system. In this paper we present the evaluation of the AES implementation in two subclusters: the first formed by four Opteron processing units and the second one formed by four Xeon processing units. As it can be seen in Table 5 the AES algorithm scales up very well – the speed up value for four nodes is between 3.71 and 3.73.

5. Summary and Conclusion

The paper provides a short overview of the components of our heterogenous cluster system integrating CPU and GPU devices from various vendors. We described the hardware architecture and the software framework that form our cluster. The cluster system was designed to be powerful, effective, scalable, flexible, and easy to use. It is especially useful in complex calculations and parallel processing of large volumes of data in which a speed of calculation and data decomposition are of essence. Cryptography algorithms are natural candidates for massively parallel computations in GPU/CPU clusters. Our experimental results presented in this paper demonstrate the effectiveness and scalability of the HGCC cluster system, and confirm that the direction to speed up cryptography techniques is to port them to GPU units. As a final observation we can say that heterogeneous computing systems offer a new opportunity to increase the performance of parallel HPC applications on clusters, by combining traditional CPU and general purpose GPU devices.

Acknowledgment

The work was supported by the National Centre for Research and Development (NCBiR) under grant number O R00 0091 11.

References

[1] F. Berman, G. Fox, and A. J. G. Hey, *Grid Computing: Making the Global Infrastructure a Reality*. New York: Wiley, 2003.

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

[3] Wen-Mei W. Hwu (Ed.), *GPU Computing Gems Emerald Edition*. Morgan Kaufman, 2011.

[4] R. Lottiaux, B. Boissinot, P. Gallard, G. Vallee, and C. Morin, "Openmosix, openssi and kerrighed: a comparative study", in *Proc. IEEE Int. Symp. Cluster Comput. and the Grid CCGrid'05*, Cardiff, UK, 2005, vol. 2, pp. 1016–1023.

[5] A. Barak and A. Shiloh, "The mosix virtual opencl (VCL) cluster platform", in *Proc. Intel Eur. Res. Innov. Conf.*, Leixlip, Ireland, 2011, p. 196.

[6] V. Kindratenko, J. Enos, G. Shi, M. Showerman, G. Arnold, J. Stone, J. Phillips, and W. Hwu, "GPU clusters for high-performance computing", in *Proc. Worksh. Paral. Programm. Accelerator Clust. PPAC'09*, New Orleans, LA, USA, 2009.

[7] A. Di Biagio, A. Barenghi, G. Agosta, and G. Pelosi, "Design of a parallel AES for graphics hardware using the CUDA framework", in *Proc. 23rd IEEE Int. Parallel Distrib. Proces. Symp. IPDPS 2009*, Rome, Italy, 2009, pp. 1–8.

[8] J. W. Bos, D. A. Osvik, and D. Stefan, "Fast implementations of AES on various platforms", Tech. rep., Cryptology ePrint Archive, Report 2009/501, 2009 [Online]. Available: http://eprint.iacr.org

[9] D. Le, J. Chang, X. Gou, A. Zhang, and C. Lu, "Parallel aes algorithm for fast data encryption on GPU", in *Proc. 2nd Int. Conf. Comp. Engin. Technol. ICCET 2010*, Chengdu, China, 2010, vol. 6, pp. V6–1.

[10] C. Mei, H. Jiang, and J. Jenness, "CUDA-based aes parallelization with fine-tuned GPU memory utilization", in *Proc. IEEE Int. Symp. Parallel & Distrib. Proces., Worksh. and Phd Forum IPDPSW 2010*, Atlanta, Georgia, USA, 2010, pp. 1–7.

[11] C. Li, H. Wu, S. Chen, X. Li, and D. Guo, "Efficient implementation for MD5-RC4 encryption using GPU with CUDA", in *Proc. 3rd Int. Conf. Anti-Counterfeiting, Secur., Identif. Commun. ASID 2009*, Hong Kong, China, 2009, pp. 167–170.

[12] Z. Wang, J. Graham, N. Ajam, and H. Jiang, "Design and optimization of hybrid MD5-blowfish encryption on GPUs", in *Proc. Int. Conf. Paral. Distrib. Proces. Techn. Appl. PDPTA'11*, Las Vegas, Nevada, USA, 2011, pp. 18–21.

[13] G. Liu, H. An, W. Han, G. Xu, P. Yao, M. Xu, X. Hao, and Y. Wang, "A program behavior study of block cryptography algorithms on GPGPU", in *Proc. 4th Int. Conf. Frontier Comp. Sci. Technol. FCST'09*, Shanghai, China, 2009, pp. 33–39.

[14] S. Fleissner, "GPU-accelerated montgomery exponentiation", in *Proc. Int. Conf. Computat. Sci. ICCS 2007*, Beijing, China, 2007, pp. 213–220, 2007.

[15] O. Harrison and J. Waldron, "Efficient acceleration of asymmetric cryptography on graphics hardware", in *Proc. 2nd Int. Conf. Cryptol. in Africa AFRICACRYPT 2009*, Gammarth, Tunisia, 2009, pp. 350–367.

[16] A. Moss, D. Page, and N. P. Smart, "Toward acceleration of RSA using 3D graphics hardware", in *Proc. 11th IMA Int. Conf. Cryptography and Coding*, Springer, 2007, pp. 364–383.

[17] J. Hermans, F. Vercauteren, and B. Preneel, "Speed records for NTRU", in *Proc. Topics in Cryptol. CT-RSA 2010*, San Francisco, CA, USA, 2010, pp. 73–88.

[18] J. Hermans, F. Vercauteren, and B. Preneel, "Implementing NTRU on a GPU", Darmstadt University of Technology, Darmstadt, Germany, 2009.

[19] M. Marks, "Enhancing wsn localization robustness utilizing HPC environment", in *Proc. Eur. Conf. Model. Simul. ECMS 2012*, Koblenz, Germany, 2012, pp. 167–170.

[20] W. Szynkiewicz and J. Błaszczyk, "Optimization-based approach to path planning for closed chain robot systems", *Int. J. Appl. Mathema. Comp. Sci. ACMS*, vol. 21, no. 4, pp. 659–670, 2011.

[21] M. Marks, J. Jantura, E. Niewiadomska-Szynkiewicz, P. Strzelczyk, and K. Gózdz, "Heterogenous GPU/GPU cluster for high performance computing in cryptography", *Comp. Sci.*, vol. 14, no. 2, pp. 63–79, 2012.

[22] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. New York: CRC Press, 1996.

Ewa Niewiadomska-Szynkiewicz D.Sc. (2005), Ph.D. (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 three books and over 120 journal and conference papers. Her research interests focus on complex systems modeling and control, computer simulation, global optimization, parallel computation, computer networks and ad hoc networks. She was involved in a number of research projects including EU projects, coordinated the Groups activities, managed organization of a number of national-level and international conferences.
E-mail: ens@ia.pw.edu.pl
Institute of Control and Computation Engineering
Warsaw University of Technology
Nowowiejska st 15/19
00-665 Warsaw, Poland

E-mail: ewan@nask.pl
Research and Academic Computer Network (NASK)
Wąwozowa st 18
02-796 Warsaw, Poland

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.
E-mail: M.Marks@ia.pw.edu.pl
Institute of Control and Computation Engineering
Warsaw University of Technology
Nowowiejska 15/19
00-665 Warsaw, Poland

E-mail: M.Marks@nask.pl
Research and Academic Computer Network (NASK)
Wawozowa st 18
02-796 Warsaw, Poland

Jarosław Jantura received his M.Sc. in Electrical Engineering from the Kielce University of Technology, Poland, in 2004. Currently he works as Technical Project Leader at NASK. His interests focus on software engineering and computer graphics.

E-mail: jaroslawj@nask.pl
Research and Academic Computer Network (NASK)
Wąwozowa st 18
02-796 Warsaw, Poland

Mikołaj Podbielski received his M.Sc. in Telecommunications from the Warsaw University of Technology, Poland, in 2010. Currently he works as software engineer at NASK. His interests focus on software engineering and GPGPU.

E-mail: mikolajp@nask.pl
Research and Academic Computer Network (NASK)
Wąwozowa st 18
02-796 Warsaw, Poland