2010 Fifth International Conference on Systems and Networks Communications

AES Performance Analysis on Several Programming Environments, Operating Systems or Computational Platforms

Radu Tomoiaga
Faculty of Automatics and Computers
University Politehnica of Timişoara
Timisoara, V. Parvan 2, 1900 Romania
Email: radugam@mailcity.com

Mircea Stratulat
Faculty of Automatics and Computers
University Politehnica of Timişoara
Timisoara, V. Parvan 2, 1900 Romania
Email: mircea.stratulat@cs.upt.ro

***Abstract* —** Advanced Encryption Standard's performance is changing depending on the platform it is run. In doing these benchmarks, we observed the behavior of AES algorithm on several programming environments, on different operating systems and on various computational platforms concluding that not all of them offer the same performance. We noticed, in doing these tests, that the best performance is offered by CUDA environment using the Graphic Processing Unit.

Keywords - AES; benchmark; CUDA; GPU.

I. INTRODUCTION

The quality characteristics of a system are conditioned by the quality attributes, and each attribute is measured through one or more metrics. One or more evaluation elements correspond to a metric.

The performance evaluation is generally made according to characteristics, attributes, metrics and evaluation elements.

The quality of software is determined by the quality attributes. Each attribute is measured through one or more metrics and one or more evaluation elements correspond to a metric.

Luken stated that measuring the performance can be done by measuring time [7]. When we talk about time, we refer to the response time or execution time. So, we evaluate the difference between the start time and end time of an event. Ştefănescu defined the performance, as being (Execution time)$^{-1}$ [15]. In order to measure the execution time, we can use the computer clock, more precisely the period of clock that is measured in nanoseconds or the clock frequency measured in megahertz. An evaluation of an informatics system can be made according to two notions, that are MIPS (Millions Instructions Per Second) and FLOPS/MFLOPS (Floating Point Operations Per Second). As a base measure for performance evaluation remains the time.

Buligiu divided the metrics in more categories: evaluation metrics regarding the processing speed and response time, metrics that study the transfer flux between the system and its components, metrics regarding the systems safety, metrics regarding the systems availability and scalability metrics [16]. In case of the processing speed, one of the most used metrics is the response time of the system. In case of some benchmarks it is indicated to be calculated an average value for the response time. This value is obtained as an average of measured times on a large number of run tests. Regarding the systems that need to be evaluated, the metric is adapted to the requirements of the respective case. In case of the cryptographic tests of the algorithms on large files, the time necessary to finalize a process is great. In the particularly case of symmetric cryptography, where an algorithm of block type is used, each block of the file is read encrypted and written in the output file. Thus, we have a large number of iterations with different inputs, which are processed sequentially. The number of iterations depends on the file size and is obtained as a ratio between the file size and the size of the input block. In this case we obtain a time that reflects more the practical reality of an encrypting application for the large information, which is saved on a memory support or hard disk. This time is theoretically greater than that assumed for the running of the algorithm only on immediate values (e.g., that are stored in RAM). The time for these values is greater than the ones that can be accessed from the internal memory because the reading / writing time from /on the hard disk is added to it. In the second part of the benchmarks, we obtain the running times for the algorithms when the input data are in the memory, not being accessed from an external support. In this case, a simple run of the algorithm is not sufficient. The algorithms are repeated a larger number of times: 100.000 in case [9] and 1.000.000 in the present case. After running, the average will be calculated and thus an average result is obtained, which reflects the necessary time for the running of the algorithm. The testing of the calculation times of the cryptographic function has been made in case [9] on dimensions of 0 bytes, 26 bytes, 62 bytes and 80 bytes, and in the present paper on the dimension of 16 bytes. Another benchmark is that created by Groza in [3]. This material discusses the implementation in Java of an authentication protocol for mobile phone applications. HMAC algorithms as well as the HASH algorithms are tested. The test platform was a mobile phone Nokia 6288 with a multitasking operating system and the number of repetitions was of 100 times. In order for the virtual machine not to be influenced by other tasks, the tests were done in the profile "flight mode", because all communication functions are stopped.

Another type of metric for measuring speed is the latency metric, which measures the waiting or delay time for a system or a component [16].

978-0-7695-4145-7/10 $26.00 © 2010 IEEE
DOI 10.1109/ICSNC.2010.33

In order to understand the project system and to reach a better design, the performance modeling is necessary even from the beginning [9].

II. Related Work

Kahate, in his study regarding the impact of cryptographic algorithms on the performance of the application [5], concludes that, disregarding the algorithm, the time necessary for the encrypting or decrypting is almost the same and the size used for the input does not have a major impact on the time necessary for the computation. The used algorithms were of the type message digest (MD5, SHA1, SHA 512), symmetric algorithms (AES, 3DES, Blowfish) and asymmetric algorithms (RSA). The input length had varied between 14 and 203 characters. In his paper, there are no details regarding the way in which the tests have been done, if the iteration has been used and if so how many times. This experiment refers to data of very small sizes, but doesn't cover large data, which, obviously, influence the computational performance.

During the last years, due to the slow processors evolution, hard computing power application developers oriented towards other type of processors. Graphic Processors were taken in consideration. This were initially designed and developed for 3D rendering, video encoding and decoding and for game engines. Software like this require a big amount of computing power and the processors on personal computers couldn't offer this. Graphic Card developers designed more and more powerful graphical processors and gave software developers the chance to write their own programs to use co processing on CPU and GPU. In [11], a list of NVIDIA video cards is presented. This cards support CUDA environment being able to accelerate tasks. From this list we mention GeForce and Quadro products that can be installed in a PCI Express slot in a personal computer.

Starting from Cooks success in 2005 [1], of implementing a cryptographic algorithm on a GPU, Yeom analyzed the improved performances using DirectX and OpenGL [10], and after finalizing his research he concluded that an Intel Core 2 Quad (QX6850) processor is able of speeds up to 96 GFLOP, while a NVIDIA GeForce 8800GTX is capable of 330 GFLOP. In his tests AES has a 4.5 Gbps and DES 2.8 Gbps performance on this GPU.

Kipper speaks about implementing AES on GPU and concludes that the algorithm is 14.5 faster than on a classic processor [6]. He also says that cracking AES attempts, through brute-force attack types, is unfeasible on a performance gain of this level. In a similar project, Luken speaks about encrypting with AES and DES using GPU hardware acceleration [6]. The tests were done on data volume up to 100 Mb, and the performances were as following: AES is 3.75 faster on GPU than on CPU and DES is 4.5 faster on GPU than on CPU.

Manavski tested CUDA compatibility in hardware acceleration for AES on NVIDIA graphic cards [8]. His best result was on AES 128, for an 8 MB input file, the performance being of 8.28 Gbps. The GPU algorithm was 19, 60 times faster than the CPU algorithm.

A Rijandael computational complexity analysis was done by Graneli and Boato [2]. In [2], the authors compare Rijndael, Camelia and Shacal-2, and conclude that "Rijndael is very good and can be used as reference for benchmarks [2]". In Table 1, the values regarding AES are presented according to the tests done by the authors.

TABLE I. AES Computational complexity

Name	Operations			
	AND	***OR***	***Shift(bytes)***	***Adding 32 bit***
AES General	5836	4254	1336	0
Key expansion	1536	1536	846	144
Encrypt	4912	3624	1188	0
Decrypt	14896	11112	3654	0

Operation	***Algorihtm (key dimension)***		
	128	192	256
AND	7236	8784	10334
OR	5418	6536	7667

AES Computational complexity. Operations [2]

III. The Benchmarks

In this section, the implemented benchmark applications are described. The applications were developed in four programming environments. For the Windows platform (XP SP3 and 2000 SP4) Java/Eclipse and Visual Studio 2008 (Visual C# and Visual Basic) were used. In order to run these applications within the stand alone packet option, the operating system must have the Dot. Net. Framework 3.5 SP1 installed. Classes used in test applications belong to System.Security.Cryptography. In order to test algorithms mentioned in this benchmark in the UNIX operating system the Live CD Kubuntu 9.04 distribution has been selected. The Live CD solution has been chosen because it was the most convenient one and could be ran on all platforms without involving installing a new operating system and risking problems to arise when installing it on one of the platforms. The fact that the files have been read and written on a NTFS partition is a disadvantage due to the fact that the operating system natively runs on EXT2/EXT3, but at the same time an advantage, because the results of the benchmarks can be easily compared with the ones compiled on Windows considering the fact they have also been ran on NTFS partitions.

The tests have been performed in three phases.

The first phase consisted in running the algorithms on files of 1 GB, 2 GB, 3 GB... 10 GB sizes. The algorithms in this phase have been implemented under Windows, using Visual Basic, C# and under UNIX, using OpenSSL libraries.

In the second phase the tests from the first one have been repeated in Visual Basic, C# and Java, for a small amount of data accessed from RAM. In this case the data are transferred not from a hard disk, but more rapidly from the memory. Given the small amount of data the test had to be redone by a certain number of times. In the present case the number of iterations rose to 1.000.000. "On the fly" encryption has been completed by running the algorithms, using recursion as in the example below:

buffer = algorithm_encryption (buffer)

The three phases consist on testing AES algorithm on CUDA.

The time is calculated this way (Visual Basic example):

TimeSpan duration = stopTime - startTime;

duration.TotalMilliseconds

or for small volumes

duration.TotalMilliseconds/ NrIteratii [11].

For Unix the "time" command is used associated with the Open SSL command like in the following example:

time openssl dgst -md5 test1mb.txt (UNIX).

The basic tools for developing cryptographic Java applications are provided by JCA (Java Cryptography Architecture) and JCE (Java Cryptography Extension). These provide the developer with direct access to cryptographic algorithms by using the so called "*factory classes*" [4]. The practical aspect of the above mentioned is that the application appeals some of the internal classes and these will assure the functionality requested by the applications. For the Java development environment the classes used to develop the application are the ones used by [4]: javax.crypto and java.security. For implementing the application Java JDK 1.6.0.17 along with the Eclipse 3.1.2 platform has been used [10].

During the first phase five workstations have been used as test platforms, configured as shown in Table 2.

During the second phase three workstations have been used, configured as shown in Table 3.

The platforms have been chosen so that the algorithms could be tested on multiple types of units and systems.

Regarding the test done on CUDA (the third phase) we used the implementation presented in [14] by Urmas Rosenberg.

TABLE II. PLATFORM DESCRIPTION PHASE 1

PC	Processor	Memory	HDD
PC1	Core(TM)2 Duo P8400 @ 2.26GHz, CPU Wolfdale	Slot 1+ Slot3 Samsung 2048 MBytes (400 MHz)	WD 250 GB 5400 RPM
PC2	Core(TM)2 Duo CPU E6750 @ 2.66GHz ,Conroe	Slot 1+Slot3 Kingston 2048 MBytes (333 MHz)	Seagate 500 GB 7200 RPM
PC3	Pentium(R) 4 CPU 3.00GHz Prescott-2M	XMM1+3 JTAG 256 MBytes 533 MHz	WD 80 GB 7200 RPM
PC4	Dual CPU E2140 @ 1.60GHz CPU Conroe-1M	Slot 1+ Slot3 Nanya 512 MBytes (333 MHz)	Maxtor 250 GB 7200 RPM
PC5	Intel Core 2 Quad Yorkfield	Slot 1+Slot3 Kingston 2048 MBytes (400 MHz)	2 HDD WD 1 TB RAID

Platform description phase 1 (information obtained with System Info for Windows)

TABLE III. PLATFORM DESCRIPTION PHASE 2

PC	Processor	Memory
PC1	Core(TM)2 Duo CPU E6750 @ 2.66GHz Conroe	Slot 1+Slot3 Kingston 2048 MBytes DDR2 (333 MHz)
PC2	ATOM N270 @ 1.60GHz Dothan	Slot 2 Samsung 1024 MBytes (333 MHz)
PC3	Intel Core Duo T2400 @ 1.83GHz Yonah DC L2 Cache Speed 1828.77 MHz	Slot 1+Slot 3 Samsung 512 MBytes (266 MHz)

Platform description phase 2 (information obtained with System Info for Windows)

IV. RESULTS

In Figure 1, we present a comparison of AES results, obtained on the five platforms for large volume data. PC2 has, overall, the best results. PC5 obtains the worst performance, although it has the best hardware configuration. The explanation for this weak performance resides in the mirroring RAID hard drives configuration. It seems that time increases for this type of algorithms, when working with large data.

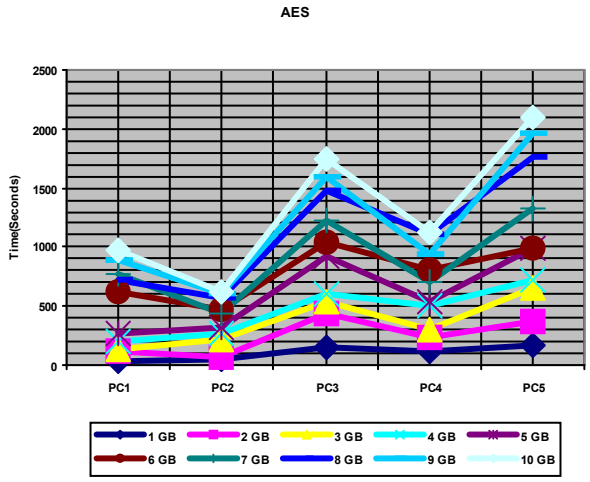

Figure 1. AES Comparison in Visual Basic

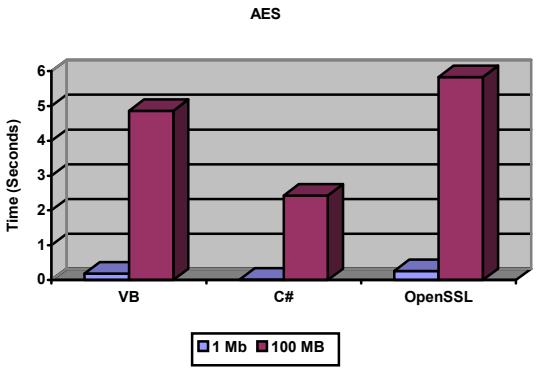

Figure 2. PC2 1 Mb and 100 Mb files

The results for benchmarking AES on PC2 in Visual Basic, C# and OpenSSL for file sizes of 1 Mb and 100 Mb can be seen in Figure 2. C# needs the smallest amount of time while OpenSSL the takes the longest time to finish the

tests. OpenSSL running on NTFS partition could be the reason why it needs more time to compute the given tasks.

In the next figure, doing the tests, again on PC2, but with larger data, we observed that OpenSSL and CUDA maintain a linear growth of time from file to file, while Visual Basic and C# have some exceptions. Also C# has the longest time for file larger than 6 Gb. The reason for this could be that, being run on Windows, the application have the processor for an amount of time, after that the operating system (which is not a true multitasking system) gives it to another process.

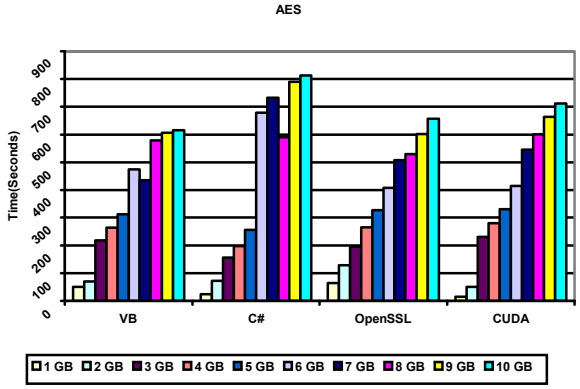

Figure 3. PC2 Large Files

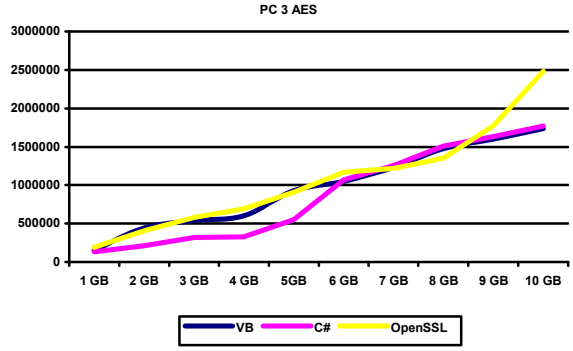

Figure 4. PC3. AES [10]

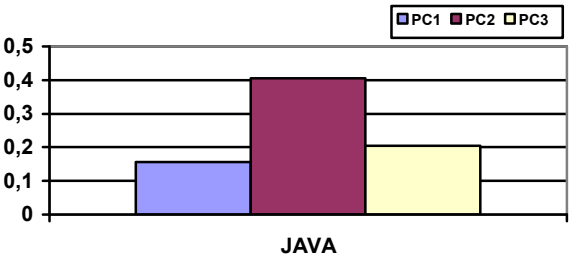

Figure 5. AES.Java

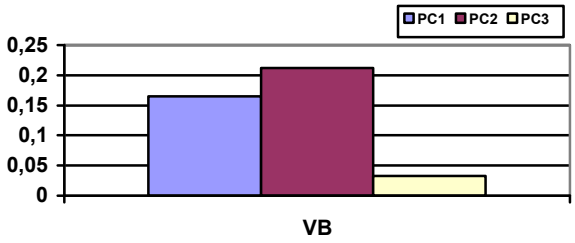

Figure 6. AES.VB

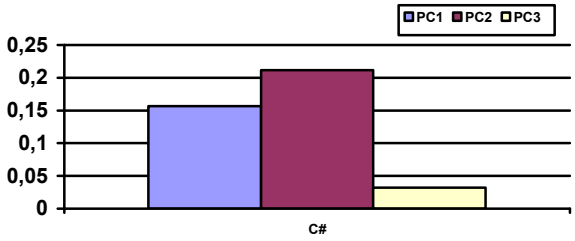

Figure 7. AES.C#

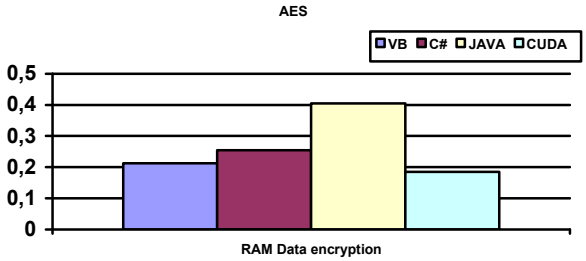

Figure 8. PC1 AES.

V. CONCLUSION AND FUTURE WORK

After analyzing the obtained results of the benchmarks, we concluded that the processors offer performance that varies depending on the size of the input data, the algorithms, the memory characteristics, programming language but also on the operating system. Regarding the tests with the data acquired from RAM, we can say that Java usually had computational time greater than Dot Net, meaning that in this case Java needs more time. Between C# and Visual Basic there were no big differences, the two having similar performances and behavior. At this benchmarks the weakest processor was the Atom N270. Regarding the tests done on big volume data we can say that C# and Visual Basic have almost identical performances beginning from 6 Gb to 10 Gb on AES. Up to 6 Gb C# has an easy advantage (figure 5). OpenSSL has almost the same behavior as Visual Basic up to 9 Gb, over this step, the computing time for OpenSSL unexpectedly increases. This behavior is characteristic for PC3, platform that was one of the slowest from the chosen

ones. The surprise in these tests was PC5, which in Windows 2000 had the worst performances on Dot Net, although the hardware configuration was better than on the other platforms. A crucial factor for these results was the RAID configuration for the two hard drives which was a mirroring RAID. In the case of the platforms that were subject of this benchmark we can easily spot a winner, but the issue is not that simple when discussing about the programming languages and the operating systems and winner couldn't be established, only CUDA seeming to gain a small advantage over the other competitors. Every programming language and applications that were tested get better performances then the others in some points of the tests, been beaten by the others in other tests, or in the same test at another level.

Looking at CUDA environment, at the implementation done in [14] and tested on platform PC2 we can conclude that the results of the tests done with data stored in RAM were better then the ones on the CPU. Comparing to VB, CUDA seems to be just a little faster, but when comparing to C# or Java, CUDA is much faster.

Regarding the tests done using large files, the results were inconclusive, as CUDA was better only on the 1 GB and 2 GB files [11]. On the other files the times CUDA got were higher than the other programming languages, except some files on C# (Figure 3). It seems that the delay caused by reading date from the hard disk were the large files are stored, affects the overall time of CUDA tests. The performance of AES implementation on CUDA is better according to Figure 8. The time for each test increases, when another factor influences the computational time of the entire test (HDD read/write).

Figure 5 presents computational times obtained when benchmarking AES in Java on the three platforms, in Figure 6, computational times obtained when benchmarking AES in Visual Basic, in Figure 7 computational times obtained when benchmarking AES in C#. It can be observed that PC2 (the Atom processor) has the worst performance of the three tested.

Based on our evaluation, CUDA environment was faster in our experiments than the other competitors.

As a future research we will try to implement an own AES encryption algorithm in CUDA that will run on a GPU. The platform on which the tests will be done is PC2, this platform having a CUDA capable graphic card: NVIDIA GeForce 8800 GT. An optimization on the existing algorithm that was tested [14] will be done, or developing new approaches for these algorithms to get better performances than the ones published in different articles. In a second stage, we will try to integrate the algorithm from the previous step in a software application like OpenSSL, so that this can use the algorithm and benefit from the graphic processor acceleration.

ACKNOWLEDGMENT

I am thankful to Mr. Groza, whose help, guidance and support from the initial to the final level enabled me to develop an understanding of the subject.

I offer my regards to all of those who supported me in any respect during the completion of this project.

REFERENCES

[1] D.L. Cook, J. Ioannidis, A.D. Keromytis, and J. Luck. "CryptoGraphics: Secret Key Cryptography Using Graphics Cards." In *RSA Conference, Cryptographer's Track* (CT-RSA), pp. 334–350. 2005.

[2] F. Granelli and G. Boato, "A Novel Methodology for Analysis of the Computational Complexity of Block Cyphers: Rjindael, Camellia and Shacal-2 Compared", Proceedings of 3rd Conference on Security and Network Architectures (SAR'04), La Londe, France, June 21-25, 2004

[3] B. Groza, D. Pop, and I. Silea, "Java Implementation of an Authentication Protocol with Application on Mobile Phones", IEEE-TTTC International Conference on Automation, Quality & Testing, Robotics, AQTR 2008 (THETA 16), pp. 190-195, Cluj-Napoca, Romania, 2008

[4] D. Hook, "Beginning Cryptography with Java", Ed. Wiley Publishing, 2005, ISBN-13:978-0-7645-9633-9

[5] A. Kahate, "Cryptographic Algorithms–Impact On Application Performance", http://www.indicthreads.com/1519/ cryptographic-algorithms-impact-on-application-performance/, 2008

[6] M. Kipper, J. Slavkin, and D. Denisenko," Implementing AES on GPU", University of Toronto, http://www.eecg.toronto.edu/ ~moshovos/CUDA08/arx/AES_ON_GPU_report.pdf, 2009

[7] B. Luken and M. Ouyang "AES and DES Encryption with GPU", Proceedings of the ISCA 22nd International Conference on Parallel and Distributed Computing and Communication Systems, pp 67-70, 2009

[8] S. Manavski, "CUDA Compatible GPU as an efficient Hardware Accelerator for AES Cryptorgraphy" , IEEE International Conference on Signal Processing and Communication, ICSPC 2007, pp. 65–68, Nov. 2007

[9] M. Solga and B. Groza, "Computational performance evaluation for symmetric and asimetric cryptographic functions on Windows and Unix" unpublished.

[10] R. Tomoiaga and M. Stratulat, "Evaluation of DES, 3 DES and AES on WINDOWS and UNIX platforms", in press

[11] R. Tomoiaga and M. Stratulat, "AES Behavior on WINDOWS and UNIX Platforms", in press

[12] Y. Yeom, Y. Cho, and M. Yung "High-Speed Implementations of Block Cipher ARIA Using Graphics Processing Units," in Proceedings of the 2008 International Conference on Multimedia and Ubiquitous Engineering (April 24 - 26, 2008). MUE. IEEE Computer Society, Washington, DC, 271-275. 2008.

[13] http://www.nvidia.com/object/cuda_learn_products.html 17.05.2010

[14] http://math.ut.ee/~uraes/openssl-gpu/ 17.05.2010

[15] http://funinf.cs.unibuc.ro/~gheorghe/curs/arhCalc/lec/l02four.pdf, 17.05.2010

[16] http://revistaie.ase.ro/content/39/I%20Buligiu.pdf, 17.05.2010