HPCLatAm 2013, pp. 113-121 (full paper)
Session: GPU Architecture and Applications
C. Garcia Garino and M. Printista (Eds.)
Mendoza, Argentina, July 29-30, 2013

Performance Analysis of a Symmetric Cryptography Algorithm on GPU and GPU Cluster

Adrian Pousa[1], Victoria María Sanz[1,2], Armando De Giusti[1,3]

[1] Instituto de Investigación en Informática LIDI – School of Computer Science – National University of La Plata

[2] Fellow, CONICET, [3] Principal Researcher, CONICET

{apousa, vsanz, degiusti}@lidi,info.unlp.edu.ar

Abstract. This article presents a performance analysis of the symmetric encryption algorithm AES (Advanced Encryption Standard) on a machine with one GPU and a cluster of GPUs, for cases in which the memory required by the algorithm is more than that of a GPU. Two implementations were carried out, based on C language, that use the tool CUDA in the case of a single GPU and a combination of CUDA and MPI in the case of the cluster of GPUs. The experimental work carried out shows how communications in the GPU cluster negatively affect algorithm total computation time.

Keywords: GPU, GPU Cluster, AES, CUDA, Algorithm performance.

1 Introduction

In recent years, GPUs (Graphic Processing Unit) [1] have gained significance due to the high performance achieved when running general-purpose applications. However, a large number of GPUs have limited global memory capacity, which is hard to increase, unlike other computation architectures.

Traditionally, computing power and memory volume in high performance computing are increased by connecting several machines through a communication network in a cluster-like architecture. If, in addition to this, a GPU is added to each machine in the cluster (cluster of GPUs), computation power is markedly increased.

Those applications that require exploiting a GPU cluster architecture must be programmed using two different tools: one for message passing through the communication network in the cluster (e.g., MPI [2]), and another one to program the portion of the application that is run on the GPU (e.g., CUDA [3]).

On the other hand, the volume of data that are transmitted through the networks has increased considerably. This information is occasionally sensitive, so it is important that it is encoded to send it securely through a public network such as the Internet. Data encryption and decryption requires additional computation time, which can be considerable depending on data size.

AES (Advanced Encryption Standard) is a symmetric block encryption algorithm that became a standard in 2002 [4], and is currently widely used to encode information. In 2003, the government of the United States announced that the algorithm was secure enough and that it could be used for the national protection of the information [5]. So far, no efficient attacks are known, the only known attacks are those known as side-channel attacks[1][6].

This algorithm is characterized for being simple, fast, and consuming little resources. However, the time required to encrypt and decrypt large amounts of data is significant; the possibilities offered by multicore architectures can be exploited to reduce this time. In previous work [7] [8] the great efficiency achieved when running a version of this algorithm on a GPU was shown, both regarding computation time and energy consumption, in relation to versions of the algorithm that were adapted to other multicore architectures (multicore and multicore cluster).

The purpose of this paper is comparing the execution time of the AES algorithm on a machine that has a single GPU and a cluster of GPUs, for those cases in which the amount of memory required by the algorithm is more than that of the *device.* To this end, two versions of AES were implemented:

- AES-GPU: it fragments the data to encrypt and invokes the kernel several times to process the data in the GPU of a single machine. For this version, the tool CUDA was used.
- AES-GPUCluster: it divides the data to be encrypted among the various machines in the cluster, and the kernel is invoked in each of them to process the assigned portion on the machine's GPU. For this version, the tools MPI and CUDA were used in combination.

The experimental work carried out shows that communications through the network have a negative impact when using a cluster of GPUs for this algorithm in particular.

2 Overview of the AES Algorithm

AES (Advanced Encryption Standard) is characterized for being a block-encryption algorithm. The data to be encrypted are divided in fix-sized blocks (128 bits), where each block is represented as a matrix of 4x4 bytes called *state*, as shown in Figure 1.

[1] A side channel attack does not attack the encryption algorithm, it rather exploits implementation vulnerabilities that can reveal data as the encryption is carried out.

AE	03	1F	2A	1E	3F	01	7A	21	04	CF	7A	1C	33	11	27

128-bit block

AE	1E	21	1C
03	3F	04	33
1F	01	CF	11
2A	7A	7A	27

State

Fig. 1. AES State.

Each *state* goes through eleven rounds of transformation, each of them formed by a set of operations. The eleven rounds can be classified into three types: one initial round, nine standard rounds, and one final round.

Since AES is a symmetric algorithm, it uses the same key to encrypt and decrypt the data; the size of this key is 128 bits as indicated by the standard. This key is called *initial key*, and it is used to generate ten more keys by means of a mathematical procedure. The ten resulting keys, together with the *initial key*, are called *subkeys,* and they are used one in each of the rounds.

The initial round performs only one operation:

AddRoundKey: a byte by byte XOR between the *state* and the initial key is performed.

Each of the following nine rounds, called *standard* rounds, applies 4 operations in this order:

SubBytes: each state byte is replaced by another one taken from a byte-substitution table whose elements are pre-computed. The replacement byte is obtained by accessing the table taking the first 4 bits of the byte to be substituted as the row index and the last 4 bits as column index. The size of the table is 16x16 bytes.

ShiftRows: with the exception of the first state row, which is not modified, the bytes on the remaining rows are cyclically rotated to the left: once for the second row, twice for the third, and three times for the fourth.

MixColumns: a linear transformation is applied to each state column, and the column is substituted by the result of this operation.

AddRoundKey: this is the same as the initial round, but using the following subkey.

The final round is formed by 3 operations:

SubBytes: similar to the standard rounds.

ShiftRows: similar to the standard rounds.

AddRoundKey: the same as in the previous rounds, but using the last subkey.

Figure 2 shows a scheme of the algorithm.

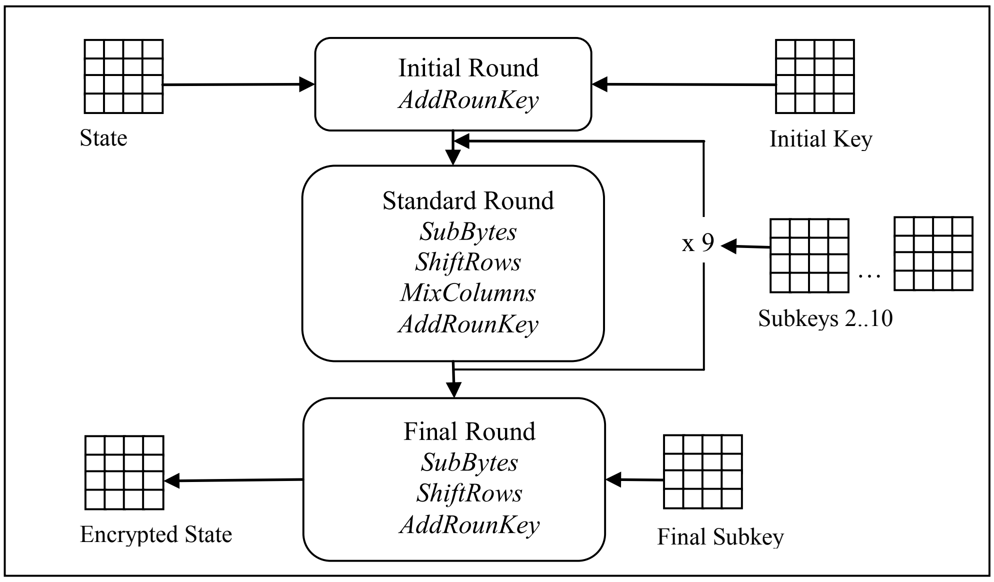

Fig. 2. Rounds of the AES algorithm on a state.

3 Implementations of the AES Algorithm

In this paper, we consider cases in which the amount of memory required by the algorithm is greater than the memory of a GPU.

There were two implementations of the algorithm:

- AES-GPU: the tool CUDA and a single GPU were used for encrypting all data.
- AES-GPUCluster: MPI is used to divide the data to be encrypted among the machines in the cluster, and then each machine uses its own GPU to carry out computation.

3.1 Implementation of AES-GPU

Since the amount of memory required by the algorithm is greater than that available in the *device,* the data to be encrypted have to be divided into fragments. To process each fragment, the following must be done: copying fragment data from the CPU to the GPU, invoking the algorithm on the GPU to encrypt the data, and then recovering the encrypted data from the GPU memory by copying them to the CPU memory.

The factors that should be taken into account are the time required for CPU-GPU and GPU-CPU memory transfers for each fragment, and the effect of fragment size on total time, since it could affect the number of calls to CUDA *kernel.*

3.2 Implementation of AES-GPUCluster

The data to be encrypted are proportionally distributed among the machines in the cluster, so that each machine uses its local GPU to perform the actual encryption. It is assumed that each GPU has sufficient memory capacity to encrypt the portion of data it receives, so no fragment division is required as in the previous case.

The factors to be taken into account are the time required to copy the data from the CPU to the GPU, the time required to retrieve the encrypted data from the GPU to the CPU, and the time required for communications when distributing data and retrieving encrypted data.

3.3 Description of AES Implementation Using CUDA

The AES algorithm subkey calculation is done at the host, since execution time for this task is negligible, leaving only the encryption procedure to the *device.*

The host copies the subkeys and the byte substitution table to the constant memory of the *device,* since these will be read only by the threads. It then copies the data to be encrypted to the global memory of the *device.*

Next, it invokes the kernel specifying both the number of blocks and the number of threads per block.

The threads belonging to a same CUDA block will work on consecutive states; each will be responsible for encrypting one state (16-byte block). Since access to global memory is very expensive, before the state encryption stage, each thread cooperates with the other threads in its block to load the information that they have to encrypt to the *shared* memory. These accesses are done in a coalescent manner. Once the encryption stage is finished, the threads cooperate in a similar way to move the data from the shared memory to the global memory.

4 Results

To carry out the experimental work, a cluster of 8 machines connected through a Gigabit Ethernet network was used; each machine has a 1-GB-RAM Nvidia Geforce

GTX 560TI graphics card [9] with 384 SPs, distributed in 8 SMs (48 SPs each). The AES-GPU algorithm uses a single machine of the cluster.

The execution times presented here correspond only to encryption time, decryption time was not considered on account of its similarity.

As already mentioned, in the case of the AES-GPU algorithm it is assumed that the memory required by the algorithm is greater than the memory of a single GPU, and, therefore, the data to be encrypted have to be divided into fragments. Even though the memory of the *device* is 1GB, the implemented AES algorithm allows fragments that are no larger than 256MB. Tests were carried out to verify if fragment size affects execution time; to this end, various input sizes were considered for the data to be encrypted (256MB, 512MB, 1GB), dividing them in fragments of 128MB (Table 1) and 256MB (Table 2). Each table shows total CPU-GPU copying time for all the fragments used, total fragment encryption time, and total GPU-CPU copying time to retrieve the encrypted fragments.

Table 1. Times obtained with AES-GPU (in seconds) with various input sizes, divided in 128-MB fragments.

	256MB (2 fragments)	512MB (4 fragments)	1GB (8 fragments)
Total *CPU-GPU* copying time	0.048	0.095	0.18
Total encryption time	0.81	1.63	3.26
Total *GPU-CPU* copying time	0.074	0.11	0.21
Total time	**0.932**	**1.835**	**3.65**

Table 2. Times obtained with AES-GPU (in seconds) with various input sizes, divided in 256-MB fragments.

	256MB (1 fragment)	512MB (2 fragments)	1GB (4 fragments)
Total *CPU-GPU* copying time	0.048	0.093	0.18
Total encryption time	0.81	1.63	3.26
Total *GPU-CPU* copying time	0.1	0.14	0.23
Total time	**0.958**	**1.863**	**3.67**

As it can be seen, fragment size does not affect total time. For instance, total time is almost the same with an input of 1 256-MB data fragment and dividing those 256MB in two 128-MB fragments.

Table 3 shows the times obtained (in seconds) when running AES-GPUCluster for various data input sizes (256MB, 512MB, 1GB) for a 4-machine cluster. These figures include network communication times (sending the data to be encrypted and retrieving the encrypted data) and CPU-GPU copying time, encryption time, and GPU-CPU copying time for each node in the cluster. Table 4 shows these times for an 8-machine cluster.

Table 3. Times obtained (in seconds) with a 4-GPU cluster.

	256MB (64MB per GPU)	512MB (128MB per GPU)	1GB (256MB per GPU)
Communication time, sending	1.76	3.53	7.07
CPU-GPU copying time, per node	0.012	0.024	0.048
Encryption time, per node	0.2	0.4	0.81
GPU-CPU copying time, per node	0.025	0.051	0.1
Communication time, reception	1.76	3.53	7.06
Total time including communications	**3.757**	**7.535**	**15.088**
Total time excluding communications	**0.237**	**0.475**	**0.958**

Table 4. Times obtained (in seconds) with an 8-GPU cluster.

	256MB (32MB per GPU)	512MB (64MB per GPU)	1GB (128MB per GPU)
Communication time, sending	2.04	4.06	8.1
CPU-GPU copying time, per node	0.007	0.012	0.025
Encryption time, per node	0.1	0.2	0.4
GPU-CPU copying time, per node	0.012	0.026	0.051
Communication time, reception	2.03	4.14	8.28
Total time including communications	**4.189**	**8.438**	**16.856**
Total time excluding communications	**0.119**	**0.238**	**0.476**

It can be seen that for each input size, total computation time excluding communications for the AES-GPUCluster implementation improves that of AES-GPU linearly when 4- and 8-machine clusters are used (Figure 3). If we consider network communication time, a considerable time increase is observed that affects algorithm total time, which worsens as more machines are added to the cluster (Figure 4).

It can also be seen that the computation/communications ratio is of the order of 6.3% with 4 GPUs and 2.8% with 8 GPUs. This is evidence of two facts: the problem is "communication bounded," and this ratio worsens as the number of GPU boards increases.

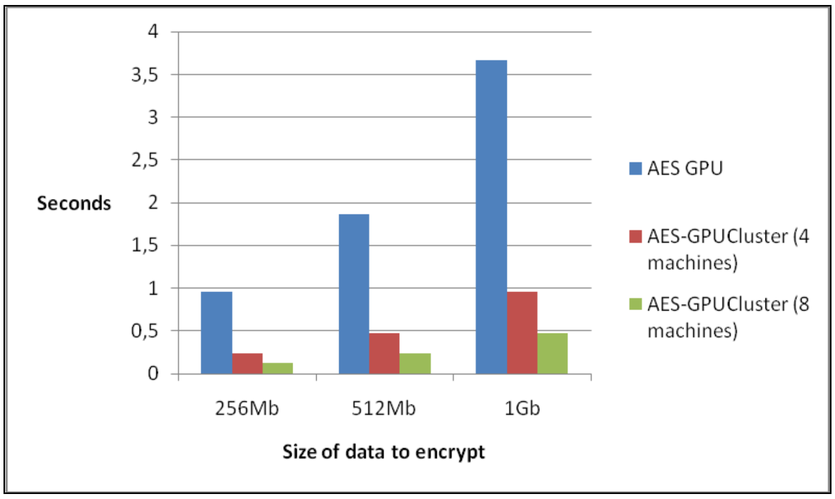

Fig. 3. AES-GPU time vs. AES-GPUCluster time (4 and 8 machines) excluding communications.

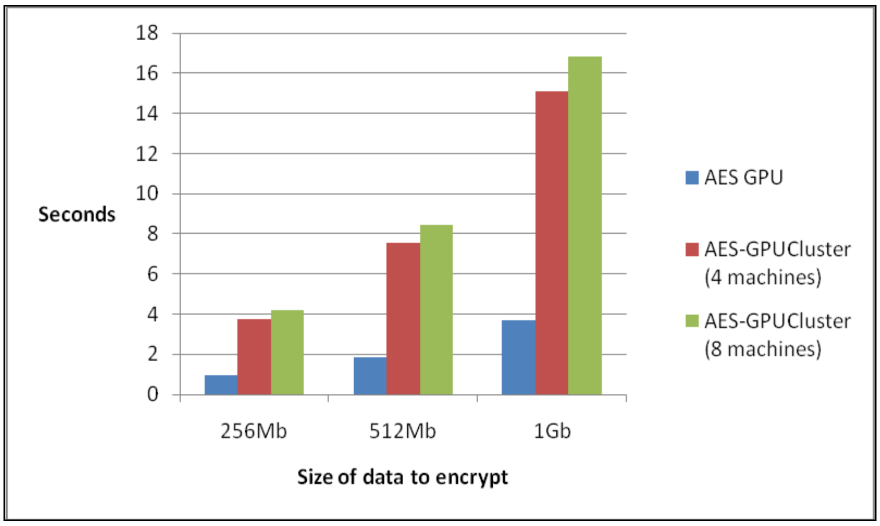

Fig. 4. AES-GPU time vs. AES-GPUCluster time (4 and 8 machines) including communications.

5 Conclusions and Future Work

A performance analysis was presented for the block-based symmetric encryption algorithm AES for scenarios in which the memory required by the algorithm is greater than GPU capacity.

To this end, two implementations were carried out: one using CUDA running on a single GPU, dividing the data to be encrypted into fragments and invoking the *kernel* several times, and another one using a combination of MPI and CUDA running on a cluster of GPUs.

The analysis of execution times with various data input sizes shows that cluster communications introduce a significant increase in total computation time, and these communications are therefore critical for a GPU cluster system considering available technologies. Consequently, for this algorithm in particular, it is better to divide the data into fragments and encrypting each of these with a single GPU, rather than using a cluster of GPUs.

We are currently analyzing applications for clusters of GPUs that are "processing bounded," in which the impact of communications is lower. Additionally, future lines of work include the optimization of algorithms for hybrid cluster architectures.

References

1. General-Purpose Computation on Graphics Hardware http://gpgpu.org/.
2. MPI Specification http://www.mpi-forum.org/docs/mpi-2.2/mpi22-report.pdf.
3. Cuda Home Page http://www.nvidia.com/object/cuda_home_new.html.
4. FIPS PUB 197: the official AES Standard http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.
5. Lynn Hathaway (June 2003). "National Policy on the Use of the Advanced Encryption Standard (AES) to Protect National Security Systems and National Security Information"http://csrc.nist.gov/groups/ST/toolkit/documents/aes/CNSS15FS.pdf.
6. D.J. Bernstein - Cache-timing attacks on AES (2005) http://cr.yp.to/antiforgery/cachetiming-20050414.pdf.
7. Pousa A., Sanz V., De Giusti A. Performance Analysis of a Symmetric Cryptographic Algorithm on Multicore Architectures. Computer Science & Technology Series - XVII Argentine Congress of Computer Science - Selected Papers. Edulp 2012.
8. Romero F., Pousa A., Sanz V., De Giusti A. Consumo Energético en Arquitecturas Multicore. Análisis sobre un Algoritmo de Criptografía Simétrica. Proceedings of the XVIII Congreso Argentino de Ciencias de la Computación. ISBN 978-987-1648-34-4
9. Nvidia Geforce GTX 560TI Specifications http://www.nvidia.com/object/product-geforce-gtx-560ti-us.html.