Encryption as a Service using Parallel Computing Frameworks

Alexandru Costin STĂNIMIR
IT&C Security Master
Department of Economic Informatics and Cybernetics
The Bucharest University of Economic Studies
ROMANIA
alex.stanimir@live.com

Abstract: In this article I present a study of an implementation named Encryption as a service which is a web service that can be deployed on a various number of devices and that can take advantage of parallelism in order to provide basic functionality of a cryptographic system: encrypt, decrypt and store data. This goal was achieved by implementing symmetric key cryptography algorithm Advanced Encryption Standard (AES) using Open Computing Language (OpenCL) and exposed this functionality through a REST web service. The performance results were obtained by deploying this solution on Windows Azure platform in order to take advantage of 20x CPU computing power in Amazon Web Service platform equipped with 2x Nvidia Tesla K20 GPU and regular home user hardware. This study represents a first step in a broader project which final goal is to provide full support for all encryption algorithms.

Key-Words: OpenCL, AES, Cloud Computing, C++11, REST Services, Benchmarking.

1. Introduction

At the time of this writing CPUs are great things because compilers evolved in a way so we can easily program without knowing the complexity behind them. But in terms of parallelism the evolution of CPUs is almost insignificant even the number of CPU cores increased because most of the instructions executed by CPU are focused on local caches, instruction decoders and branch prediction. Another option for data-parallel hardware are discrete graphics cards which basically is totally separated from CPU, has its own memory space and enough computing power. As a result during the last few years a lot of research focused on execution of basic algorithms on the Graphics Processor Units (GPU) because they had an architecture that can take advantage of data parallelism and these computationally intensive parts are also the prime target of parallel programming because where there is more work to do, there is more opportunity to divide work among cooperating parallel workers. OpenCL is the first open standard for parallel cross-platform programming for both GPUs and many-core CPUs. At this moment is managed by the Khronos Group, which is a non-profit consortium created by main hardware and software companies. As a parallel programming framework one of the main purposes of OpenCL is to not be limited to a specific vendor hardware or operating system.

2. Problem Formulation

In the cloud computing era we are seeing day by day launching new services that take advantage of so called by vendors “unlimited” computing power. In this category of “everything as a service” we can include encryption which can be implemented in any application not only as feature but as necessity and requirement of our informational world.

2.1 Reason for implementation

Developing a software solution on top of a cloud platform has a lot of challenges in terms of development because today requirements even for a simple solution can be a time consuming action. That is why belong hardware, cloud computing vendors has in their portfolio a lot of services which can easily be integrated in every application .There are services which simply integrates a login API from
different websites like Facebook or Google but also some services need computing power like SendGrid which is an service available in Windows Azure used to send email messages. Encryption of data in storage and in transit can be used to align with best practices for ensuring confidentiality and integrity of data. Recognized encryption algorithms like AES that have years of real-world exposure and testing, avoiding the classic mistake of attempting to “roll your own crypto” for applications. But the whole encryption/decryption process is a time and resource consuming but having optimized algorithms for specific devices or being able to outsource this computing process to an external service will result in a better use of available resources without affecting user experience.

2.2 Prior research
At this moment there are a few implementations of AES algorithm on CUDA platform which is quite similar with OpenCL but there are optimized to work only with NVIDIA devices and also optimal throughput and performance is achieved only on a few high-end products. Regarding encryption services there is Amazon which has in his portfolio CloudHSM which is a service that provides dedicated hardware to protect and store cryptographic keys, store, encrypt data and simply integrate with Amazon EC2 applications. Also Amazon Web Services cloud platform has GPU computing in their portfolio product which includes High End Nvidia Tesla GPU. Regarding performance of encryption or decryption on multi-core platforms previous research results show that GPU implementations are much more efficient than the serial version. Notable results are shown in the picture below where we can notice that a regular GPU device is even 15 times faster.

2.3 AES
Advanced Encryption Standard has an important role in current encryption communications and security fields because it is a valid instrument to encrypt data in applications ranging from personal to high confidential domains. The algorithm is developed by Joan Daemen and Vincent Rijmen that submitted it to AES selection process with “Rinjdael” codename. Due to its characteristics it can greatly benefit from parallel implementation because it operates on data split in 4x4 matrix of bytes, called the state, and uses a symmetric key of size 128, 192 or 256 bits. The algorithm consists in several execution cycles that convert the original data into encrypted / decrypted data. The main steps for encryption algorithm are:

- Key expansion using Rijndael’s key schedule
- Add the first round key to the state before starting the rounds with AddRoundKey when every byte of the state is combined with round key.
- First Round to n-1 Rounds
1) SubBytes – on linear substitution when every byte is replaced by another one
2) ShiftRows – transposition when every row of the state is cyclically shifted in a number of steps
3) MixColumns – mixes the columns of the state
4) AddRoundKey – every byte of the state is combined with the round key
   • Final round consist in the above operations except MixColumns
AES uses the following structures:
• Forward S-box: is generated by determining the multiplicative inverse for a given number which is transformed using an affine transformation;
• Inverse S-box: is simply the S-box run in reverse. First is calculated the inverse affine transformation of the input value and then multiplicative inverse;
• RCON: The round constant word array. It is used by KeyExpansion and it is the exponentiation of 2 to a user-specified value \( t \) performed in Rijndael’s finite fields [3].

2.4 OpenCL
OpenCL is an open standard aimed at providing a programming environment suitable to access heterogeneous architecture. In particular OpenCL allow executing computational programs on multi many-core processors. Considering the increasing availability of such types of processors, OpenCL is playing a crucial role to enable the wide access of portable applications to innovate computational resources. To achieve this aim, various levels of abstraction have been introduces in the OpenCL model: Platform layer performs an abstraction of the number and type of computing devices in the minimize transfers to hardware platform. At this level are made available to the developer the routines to query and to manage the computing device, to create the contexts and work-queues to submit instructions on it (it is called kernel).
Execution layer is based on the concept of kernel. The kernel is a collection of instructions that are executed on the computing device, GPU or CPU multicore, called OpenCL device. The execution of OpenCL application can be divided in two parts: host program and kernel program. The host program is executed on CPU, defines the context (platform layer) for the kernels and manages their execution. Language later specification describes the syntax and programming interface for writing compute kernels (set of instructions to execute on computing device). The language is based on a subset of ISO standard C99 [5].
The figure 3 shows the OpenCL device architecture with memory regions and how they are related to the platform model.

As you can see, there are four memory levels:
• Global memory: permits read/write to all work items in all work-groups. It is advisable to minimize transfer to and from this type of memory
• Constant memory : global memory’s section that is immutable during kernel execution
• Local memory: permits read/write to all work-items owned by a work-group. Can be used to allocate variables shared in a work-group.
• Private memory: permits read/write to a work-item. Variables into private memory are not visible to other work-items.

The application running on the host uses the OpenCL API to create memory objects in global memory, and to enqueue memory commands that operate on these memory objects. Besides you can synchronize enqueue command by
command-enqueue barrier or using context’s events. [6]

3 Problem Solution

3.1 General architecture

Encryption as a service has as core component a library which exposes AES algorithm implemented with OpenCL. This core component needed a wrapper which basically in a distributed system has to decide on the fly how to call encrypt / decrypt functionality and what parameters should be used. This decision is the result of an analysis involves checking available resources and a message queue built on web service calls.

There are also some static analysis from my personal experience which is the result of applying APOD (Analyze Parallelize Optimize Deploy) systematic optimization.

1. **Analyze** involves profiling the whole application in order to discover where can be optimized and how much will be the benefit. Discovering hotspots of an application basically it’s done on a regular code review but we don’t have to rely on human intuition as long as we have profiling tools (like gProf, VTune, VerySleepy, nSight) which can measure the impact of every line of code, benchmark and output the result.

2. **Parallelize** is the action of picking an approach, splitting set of data in sub-sets and picking the right algorithms

3. **Optimize** involves measuring the impact and the performance. We need to keep an eye to make sure memory bandwidth is always occupied and to optimize global memory accesses. For CPU computing we need to take advantage of L2 and L3 cache memory available. For GPU computing by working with global memory on the GPU side and for GPU-CPU computing by working with global memory

4. **Deploy** is the final step and also the beginning of a new analysis.

As I already stated there is a wrapper over the AES OpenCL implementation which based on how much resources is available at some point it has to set a priority for the content that has to be encrypted or has to set which device should be used or how many threads should be used.

```c
struct Content
{size_t size;
 char* data;
 int instruction;
 int priority;
 int numberofthreads;
 int devicetype;
 char* key; };
```

After the above structure is filled with all necessary parameters then the corresponding instruction is execute.

- First step is to transfer memory objects to device global memory.
- After that the key is expanded using Rijndael’s key schedule algorithm.
- OpenCL device performs kernel operations. Some sub-operations are included in a unique kernel that uses the shared memory for optimal execution to minimize the read / write latency from / to global memory. The shared memory is used like a memory buffer in order to store intermediates states. As soon as the last state is computed, this buffer is copied into global memory.

At this point kernel uses data-parallel paradigm executing a lot of operations in multiple data domains.
These operations are calls to the following serial functions:

- **SubBytes**: each byte is updated using S-BOX;
- **ShiftRows**: each row is shifted by a certain offset;
- **MixColumns**: each column is multiplied with a fixed polynomial;
- **AddRoundkey**: the subkey is combined with the state using XOR.

Also the input file is split into multiple parts depending on the local work size, priority and available resources. After the kernel finishes its computation memory objects are transferred from device into global memory and then the result is sent back to the wrapper which can send a response to the web service with the encrypted content in order to store the object on the web server storage file or under a different data structure like BLOBs.

### 3.3 Deploying encryption service

In order to access compute power of a server to several devices we have to deploy encryption library on a regular web server and to expose its functionality through a REST web service. First instance we are going to deploy it under Windows Azure platform where we want to take advantage for unlimited CPU computing power. Windows Azure applications consist of roles (essentially, components of the app). Roles come in two types: web and worker. Web roles are designed for hosting web applications under IIS. Worker roles are for everything else, with a simple “run forever” interface. Secondly we are going to deploy the service on UNIX Machine which has a high-end GPU for benchmarking purposes. On both platforms I deployed an instance of “Mongoose” which provides simple, functional, embeddable web server. This web server is responsible to instantiate HTTP Wrapper and create an endpoint responsible to manage incoming messages.

### 3.4 Consuming encryption service

Encryption service can be consumed by any client because REST services are platform independent. For each specific platform there are a lot of libraries designs to consume REST services and handle JSON. In this paper for Java platform I used Restlet framework which is a programming API which support concepts of REST and facilitates the handling of calls for both client-side and server-side applications. Last version of .Net framework includes "HttpClient" class which besides regular support for REST services has also support for asynchronous operations.

Basically encryption web service exposes two methods ( encrypt and decrypt ) which can be accessed by calling ( %SERVER%:%PORT%/encrypt or %SERVER%:%PORT%/decrypt. Also query string is also supported as following. If we want to encrypt message “m” with specific k key we can use %SERVER%:%PORT%/encrypt?m=“Text to encrypt”?k=“key” and similar syntax for decrypt action.

### 4. Conclusion

Encryption as a service provides full functionality for developers who need to develop applications that manage sensitive data and that can take the benefit of tremendous computational power of the modern GPU and modern cloud platforms.

First of all this solution demonstrates full capability of the OpenCL standard to exploit the many core property of CPUs and GPUs. This scalability is also increased by using an APOD systematic optimization approach which results is to keep hardware busy.

Also this solution is portable on different operating system and various heterogeneous devices.

Future work of this solution is to keep optimizing and add new capabilities to support encryption algorithms and parallel programming languages or directives.

### Acknowledgment

Parts of this research have been published in the Proceedings of the 6th International Conference on Security for Information Technology and Communications, SECITC 2013.
References


