INTERFACES CIENTÍFICAS

EXATAS E TECNOLÓGICAS

ISSN IMPRESSO - 2359-4934

ISSN ELETRÔNICO - 2359-4942

DOI - 10.17564/2359-4934.2015v1n3p31-42

UM OLHAR PARA O FUTURO DA ARQUITETURA DE GPU

Gabriel Galvan[1]

RESUMO

Este artigo apresenta a definição, planejamento, operação, análise e resultados de um experimento controlado in-vitro, localizado no contexto de segurança da informação, nomeadamente, sobre o desempenho do algoritmo de criptografia simétrica chamado AES-128-ECB. O experimento é para quantificar o tempo de execução, com a premissa de ser avaliado com dois paradigmas diferentes de execução, isto é, o algoritmo é avaliado com o paradigma sequencial clássico (CPU) e contraste com o paradigma paralelo (GPU).

PALAVRAS-CHAVE

GPU. CPU. criptografia. AES-128-ECB.

ABSTRACT

This paper presents the definition, planning, operation, analysis and results of a controlled in-vitro experiment, located in the context of information security, particularly assess the performance of symmetric encryption algorithm called AES-128-ECB. The experiment is to quantify the execution time with the premise of being evaluated with two different execution paradigms, ie, the algorithm is evaluated with sequential classic paradigm (CPU) and contrast with the parallel paradigm (GPU).

KEYWORDS

GPU. CPU. Encryption. AES-128-ECB

RESUMEN

En este artículo se presenta la definición, planificación, operación, análisis y resultados de un experimento controlado in vitro, que se encuentra en el contexto de seguridad de información, en particular, sobre el desempeño del algoritmo de criptografía simétrica llamado AES-128-ECB. El experimento consiste en medir el tiempo de ejecución, con la condición de ser evaluado con dos paradigmas diferentes de aplicación, es decir, el algoritmo es evaluado con el paradigma clásico secuencial (CPU) y el contraste con el paralelo paradigma (GPU).

PALABRAS CLAVE

GPU. CPU. criptografía. AES-128-ECB.

1 INTRODUÇÃO

A computação paralela (HENNESSY ET AL, 1996 e SANDERS ET AL, 2010) é um paradigma que está ganhando terreno; as múltiplas operações são realizadas simultaneamente, segmentando um problema em subproblemas e em seguida, resolvendo cada um destes concorrentemente, sendo que dentro do processamento existem vários métodos para fazer paralelismo, sendo eles: Nível bit (BLP, Bit Level Paralelism), nível de instrução (ILP, Instruction Level Paralelism), nível de dados (DLP, Data Level Paralelism) e nível de tarefa (Task Level Paralelism) (RODRÍGUEZ ET AL, 2010).

O crescimento sustentado vem da tecnologia gráfica chamada Graphics Processing Unit (GPU) ou processador vetorial, na taxonomia de Flynn (1972) é conhecido pela sigla SIMD (Single Instruction Multiple Data), em que a mesma instrução é aplicada a um volume de dados. Este paralelismo é obtido a partir de várias unidades de processamento e da unidade de controle comum, para que todos recebam e executem a mesma instrução (de forma síncrona), mas operem em dados diferentes, com este vemos a explotação de paralelismo de dados.

O conceito **General Purpose Computing on Graphics Processing Units** – GPGPU (LUEBKE ET AL, 2004), no domínio da computação sobre estudar e aproveitar o poder computacional que existe na GPU atual, em que a evolução veio de placas gráficas atuais para entrar no espaço da computação científica. O processamento gráfico está alcançando resultados extraordinários, inicialmente, implementações GPGPU foram feitas em linguagem de baixo nível, especificamente o desenvolvimento está diretamente ligado à sua arquitetura de hardware.

Atualmente, várias empresas têm desenvolvido ferramentas de alto nível, a fim de obter um grau de abstração e facilitar a criação de aplicações. Entre as características mais salientes destacam-se grandes conjuntos de dados, o paralelismo de grão fino SIMD, o gerenciamento de memória explícita e plataformas de desenvolvimento variadas, como **Open Computing Languaje** (OpenCL) (OPENCL, 2015), **Computing Unified Device Architecture** (CUDA), Brook+.

O conceito de criptologia segundo Aguirre (1999) é uma disciplina que utiliza técnicas de mascaramento de informações intimamente ligadas ao contexto da computação, assim, este estudo tem como objetivo avaliar a algoritmo de criptografia simétrica AES (U.S, 2001). Este sistema de cifragem sobre bloco está na categoria de sistemas de cifragem simétrico. Os últimos capítulos explicam sua estrutura operacional.

Conclui-se que as técnicas atuais de processamento de informação estão migrando para formas mais eficientes em dois eixos, ou seja, o primeiro mostra soluções mistas de processamento de informações, tais como a execução sequencial clássica por meio de uma unidade de processamento central (CPU) e um paradigma que funciona em paralelo, uma unidade de processamento gráfico de hardware específico chamado (GPU), que funciona como o co-processador CPU. O segundo eixo é o consumo de energia (LIMA ET AL, 2013) o processamento desses dispositivos.

A eletrônica está envolvida no processo de otimização alcançado por meio da miniaturização dos dispositivos básicos de processamento, a fim de reduzir o espaço, e o consumo de energia e aumentar a velocidade de processamento.

Concordamos que a tecnologia de processamento paralelo por meio de hardware e software de propósito específico é incorporada no contexto do computador para uso e explotação. O desafio é a utilização de processos de reengenharia, a fim de otimizar os recursos computacionais necessários executando um programa de computador.

A motivação deste trabalho é analisar a eficiência algorítmica do algoritmo de criptografia simétrica AES-128-ECB, executado em uma arquitetura sequencial frente à execução em uma arquitetura paralela.

A principal contribuição deste trabalho é dada pela proposta de fazer reengenharia de algoritmos de criptografia simétricos, utilizar especialmente o algoritmo AES-128-ECB, que tem a característica de trabalhar com blocos de dados e operações modulares. Todas estas características tornam possível a uti-

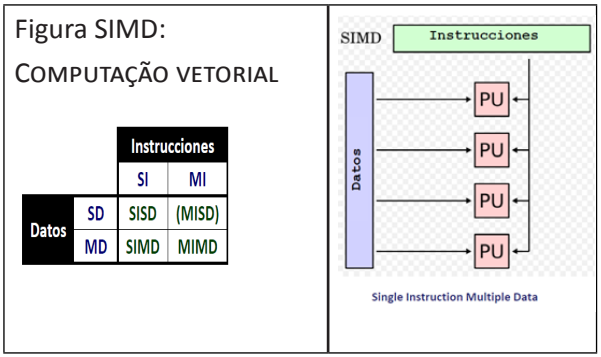
lização de técnicas de programação paralelas, a fim de obter uma otimização dos recursos do computador por meio do uso da plataforma de **hardware** GPU e **software** CUDA.

2 CONCEITOS DE GPU, CUDA E OPENSSL

2.1 GPU

A arquitetura de computação paralela aproveita o poder de um chip gráfico chamado GPU, sendo este um tipo de processador projetado para cálculos relacionados com a geração de gráficos 3D interativos. A relação entre preço e poder de computação torna-o um **commodity** tecnológico, até agora insuperável. De acordo com a classificação de FLYNN (1972), GPU é caracterizado na Figura 1.

Figura 1 – Taxonomia de Flynn



2.2 CUDA

A arquitetura de software para a gestão da GPU é denominado CUDA, esta plataforma fornece toda a lógica de programação, com base em uma extensão da linguagem C tradicional, também extensível a outras linguagens atualmente utilizadas. Entre as diferenças mais notáveis contra processamento da CPU com GPU ele consiste em compreender como o processamento de trabalho ocorre. A CPU consiste em núcleos otimizados para tarefas de processamento em série, com certo grau de paralelismo, enquanto a GPU é um grupo de núcleos menores, otimizados para desempenhar tarefas múltiplas a Figura 2 mostra as diferentes arquiteturas.

Figura 2 – Arquitetura CUDA

2.3 OPENSSL

OpenSSL (OPENSSL, 2015) é uma biblioteca de criptografia **open source** de uso geral, com base no **Secure Socket Layer** (SSL v2/v3) e do protocolo **Transport Layer Security** (TLS), este último para aplicações dentro da camada de transporte (TANENBAUM, 2003).

Entre os usos principais, estão: Transações seguras em sites, criptografia de e-mail. Uma biblioteca de segurança deve cumprir uma série de requisitos não-funcionais (PRESSMAN, 2005), confidencialidade para garantir a acessibilidade apenas para pessoas autorizadas, salvaguardar a integridade dos dados e conteúdo da fonte original, garantindo a disponibilidade de entidades autorizadas a ter acesso a informações e recursos relacionados em todos os momentos. As características mais marcantes da biblioteca OpenSSL são: Criação, gestão de chaves públicas e privadas, administração de operações de criptografia avançados, criando o tipo de certificados X.509, CSRs e CRLs.

3 ALGORITMO AES

O padrão especificado pelo algoritmo Rijndael é um cifragem simétrica sobre o bloco para ambos os processos de cifragem e decifração. O seu desenho permite a

utilização da chave de comprimento variável, sempre que um comprimento de quatro bytes. O comprimento das chaves utilizadas por padrão é AES-128, AES-192 e AES-256 Bits. Da mesma forma o algoritmo permite a utilização de blocos de informação com um tamanho variável, desde que seja múltiplo de quatro bytes, sendo o tamanho mínimo recomendado de 128 bits (16 bytes). Este algoritmo opera em nível de byte, interpretando-as como elementos de um campo de Galois GF (2^8), e ao nível de registros de 32 bits, considerando-os como polinômios de grau inferior a quatro com coeficientes que são polinômios GF (2^8).

3.1 ESTRUTURA DO ALGORITMO

Estrutura do algoritmo Rijndael é formada por um conjunto de rondadas, entendido como um conjunto de repetições de quatro funções matemáticas diferentes e invertidas. A informação gerada por cada função é um resultado intermediário, comumente chamado de estado ou estado intermediário. O algoritmo representa o estado de uma matriz retangular de bytes, com quatro linhas e N_b colunas, sendo o número de colunas N_b dependente do tamanho do bloco.

N_b = Tamanho do bloco utilizado em Bits / 32

Por exemplo, a representação de uma matriz de estado para um tamanho de bloco de 160 bits ($N_b = 5$) é mostrado na Figura 3.

Figura 3 – Matriz de estado (Nb = 5)

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$

A chave do sistema é representada como uma estrutura análoga ao estado, ou seja, é representada por um conjunto retangular de bytes com quatro linhas e N_k colunas, o número de colunas N_k varia de acordo com o tamanho da chave:

N_k = tamanho da chave em bits / 32

Por exemplo, a representação de uma chave de 128 bits ($N_k = 4$), em forma de matriz retangular é representada na Figura 4.

Figura 4 – Matriz retangular (Nk = 4)

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$

A partir deste momento a matriz de estado passa por quatro operações de transformações por rodadas, utilizando o processo de chaves e sub-chaves para todas as voltas que se geram do sistema da chave selecionada. Dessa forma as quatro operações por rodada são:

Função ByteSub: substituição com propriedade ótima de não linearidade.

Função ShiftRow y MixColumn: permite um alto nível de difusão da informação nas diferentes rodadas.

Função AddRoundKey: permite aplicar na matriz de estado uma operação OR exclusiva com a subchave correspondente a cada rodada.

3.2 DESCRIÇÃO DO PROCESSO DE CRIPTOGRAFIA

O processo de criptografia envolve quatro funções matemáticas inversíveis dos dados que se querem criptografar.

Figura 5 – Pseudocódigo do algoritmo Rijndael

Algoritmo 1

```
1: Cipher(Byte in [4*Nb], byte out [4*Nb], word w[Nb*(Nr+1)])
2: Begin
3: byte state [4, Nb]
4: state = in
5: AddRoundKey(state, w[0, Nb -1])
6: for round = 1 step 1 to Nr -1
7: SubBytes (state)
8: ShiftRows (state)
9: MixColumns (state)
10: AddRoundKey (state, w[round * Nb, (round + 1)* Nb -1 ])
11: end for
12: SubBytes(state)
13: ShiftRows(state)
14: AddRoundKey (state, w [Nr * Nb, (Nr +1)* Nb -1])
15: out = state
16: end
```

4 PROJETO EXPERIMENTAL

A metodologia selecionada para a projeto experimental é o proposto por Wohlin (2000), esta seção se concentrará na definição do objetivo e planejamento do experimento.

4.1 OBJETIVOS

O objetivo foi formalizado com o modelo GQM proposto por Basili e outros autores (1988): analisar o tempo de execução do algoritmo criptografia simétrica AES-128-ECB, a fim de avaliar o tempo médio de cifrado em relação às plataformas sequenciais (CPU) e paralelas (GPU), desde o ponto de vista dos administradores de segurança do sistema no contexto da segurança da informação.

4.2 PLANEJAMENTO

4.2.1 FORMULAÇÃO DA HIPÓTESE

As questões de pesquisa para este experimento são: A plataforma sequencial e paralela, afeta o desempenho de algoritmo de criptografia simétrica AES-128-ECB?

Qual dessas duas plataformas tem melhor eficiência algorítmica na implantação do algoritmo de criptografia simétrica AES-128-ECB?

Para avaliar estas questões, é usada a seguinte métrica: 1) O tempo médio de execução: Tanto para processamento sequencial e paralelo.

HIPÓTESE 1: Tempo de execução.

H_{0tempo}: O tempo de execução do paradigma sequencial (CPU) é igual ao tempo de execução do paradigma paralelo (GPU).

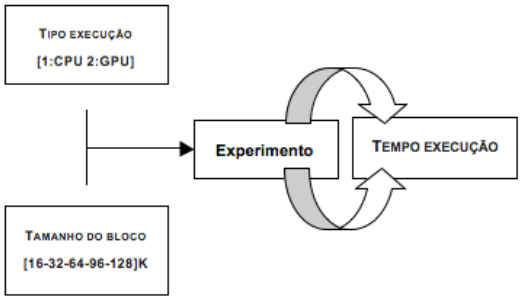
$$H_0: \mu_{cpu} = \mu_{gpu}$$

H_{1tempo}: O tempo de execução do paradigma sequencial (CPU) é diferente do tempo de execução do paradigma paralelo (GPU).

$$H_1: \mu_{cpu} \neq \mu_{gpu}$$

Para a presente hipótese, a H_0 é a que se deseja rejeitar. Para averiguar qual delas será rejeitada, serão consideradas as seguintes variáveis dependentes e independentes que se encontram na Figura 6.

Figura 6 – Variáveis do experimento.



4.2.2 SELEÇÃO DE VARIÁVEIS

Nesta seção do nosso trabalho são selecionados os tipos de variáveis (SOLINGEN, 1999). Em nosso experimento identificamos dois tipos de variáveis que são parte do processo de análise e os resultados no contexto de execução do experimento. Cada variável é única e tem uma unidade de medida mensurável.

As variáveis identificadas estão contidas no seguinte grupo.

4.2.3 VARIÁVEIS INDEPENDENTES

Estas variáveis são aquelas que podem ser controladas e alteradas dentro do experimento e são críticas para a abordagem de cenários experimentais. Dentro deste grupo definimos as seguintes variáveis.

Tamanho do bloco: representa os diferentes tamanhos de blocos que suportam o algoritmo de criptografia.

Tipo execução: representa os modos de execução do algoritmo, ou seja, nos permite escolher entre a plataforma de execução, sequencial e plataforma de execução paralela.

4.2.4 VARIÁVEIS DEPENDENTES

Esta variável representa a saída de um processo ou sistema, e são críticas para a tomada de decisão. Para este experimento propõe-se a seguinte variável.

Tempo de execução: tempo incorrido pelo algoritmo em função de cifragem.

4.2.5 ESCOLHA DO NÍVEL DE SIGNIFICÂNCIA

O nível de significância escolhido é de 99%. = 0.01.

4.2.6 SELEÇÃO DA AMOSTRA

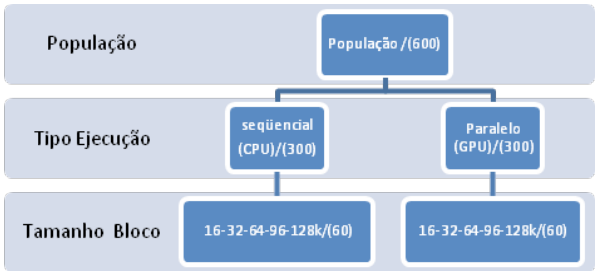
O tamanho da amostra é feita como se segue. A população é representada por N = 600 valores, esta simula o tempo de execução do algoritmo de criptografia simétrica AES-128-ECB, tendo em conta a seguinte distribuição das amostras.

Cada amostra (n) é independente tanto tipo de execução (CPU) e (GPU), como no tamanho do bloco escolhido. Em seguida, descrevem-se as amostras em cada caso, na Figura 7 está concebida a configuração.

A variável tipo de execução representa o modo de execução (CPU) e (GPU), seguido com um valor numérico correspondente ao valor do tamanho da amostra (n).

A variável tamanho de bloco representa a execução do algoritmo com respeito ao tamanho de bloco designado (16-32-64-96-128K), o valor da amostra (n) está identificada ao lado entre parênteses.

Figura 7 – Distribuição das amostras



4.2.7 PROJETO DO EXPERIMENTO

A fim de provar os resultados obtidos no experimento proposto, são utilizados métodos de análise estatísticos.

4.2.8 INSTRUMENTAÇÃO

O processo de implantação foi inicialmente feito com a configuração do ambiente de trabalho e como lidar com a coleta de dados. Este experimento será conduzido no laboratório de informática da Universidade Federal de Sergipe (UFS). O sistema é configurado para implantação do experimento. Em seguida, a configuração geral do ambiente de trabalho é detalhada.

4.2.9 INSTRUMENTAÇÃO DE OBJETOS

Instrumentação em nível de sistema operacional e ambiente de trabalho.

Uma breve descrição da configuração do sistema, onde o experimento será executado.

• Instalação do sistema operacional base: distribuição Linux Ubuntu 14-04 LTS 64Bit.

• Instalação do **drivers** da placa gráfica **NVIDIA**.

• Instalação do **ToolKit** de desenvolvimento de CUDA 7.0.

• Instalação da biblioteca OpenSSL versão 1.0.2.

• Atualização de compiladores e bibliotecas dinâmicas.

• Configuração das variáveis de trabalho.

• Compilação do programa fonte.

4.2.10 GUIAS INSTRUMENTAÇÃO DO SCRIPT DE EXECUÇÃO

A seguir, os passos na execução do script de execução do algoritmo AES-128-ECB.

4.2.11 ETAPAS DE DESENVOLVIMENTO DO SCRIPT:

Passo 1: Criando um arquivo de texto simples com um tamanho de 700 MB, usado para o cifrado.

Passo 2: O algoritmo é avaliado com os seguintes tamanhos de bloco [16, 32, 64, 96 e 128] KB.

Passo 3: O tempo de execução é medido em segundos, cada corrida tem um tempo de resposta para o processamento sequencial e paralelo.

Passo 4: Os tamanhos dos blocos são executados nos dois paradigmas, sequencial e paralelo.

Passo 5: O tempo de cifragem do algoritmo é medido com o comando **Time** do sistema operacional Linux.

Passo 6: Prossegue-se a armazenamento os dados de saída do algoritmo para sua análise em um arquivo de texto simples.

Passo 7: A implantação do teste do algoritmo é de forma exclusiva, isto é, o sistema é iniciado no modo de consola, a fim de assegurar que todos os recursos de computação estejam dedicados ao experimento.

4.2.12 NOMENCLATURA

A seguir, encontra-se detalhada a sintaxe de execução do sistema operacional que interpreta o comando de execução do algoritmo de criptografia AES-128-ECB. O comando contido na biblioteca OpenSSL, na Tabela 1 os parâmetros do comando.

Sintaxe: openssl comando [opções] [argumentos]

"openssl [**list-standard-commands** | **list-message-digest-commands** | **list-cipher-commands** | **list-cipher-algorithms** | **list-message-digest-algorithms** | **list-public-key-algorithms**]"

#openssl aes-128-ecb -bufsize 29491200 -engine gpu -in input.dat -out data.gpu -k abcd

Tabela 1 – Detalhes de configuração

Descrição: OpenSSL é uma biblioteca de criptografia implementada SSL (v2/v3) e TLS (v1).	
AES-128-ECB	Função de criptografia usada com tamanho de chave de 128 bits no modo ECB.
BUFSIZE	Tamanho do Buffer [16-32-64-96-128] K.
ENGINE GPU	Execução pela placa gráfica.
IN	Arquivo de texto simples para criptografar.
OUT	Arquivo de texto simples cifrado.
K	Chave de criptografia 128 Bits.

4.2.13 AVALIAÇÃO DA VALIDADE

A fim de garantir a validade dos dados no experimento e seguindo as recomendações de Cook (1979), consideramos que o âmbito de nossos dados do estudo correspondem à validade interna.

Para garantir a validade da execução do experimento, foram realizados testes de hardware, software e o algoritmo, antes da execução do experimento, com o intuito de descobrir quaisquer erros que possam ter influenciado o resultado dos testes.

Testes hardware:

• Funcionamento do sistema operacional.

• Funcionamento do hardware GPU.

• Testes de software:

• Funcionamento da biblioteca criptográfica OpenSSL.

• Funcionamento de bibliotecas e dependências de compiladores na plataforma CUDA.

• Teste do algoritmo.

5 OPERAÇÃO

Nessa fase ocorre a execução do experimento, isto é, a fase em que ê realizada a coleta de dados gerados pela execução do algoritmo.

A operação consiste em três etapas:

Preparação: este processo abrange os objetos envolvidos no desenvolvimento do experimento. Em primeiro lugar, a pessoa que efetua o processo de execução do algoritmo, e em segundo, os outros objetos participantes, tais como o sistema operacional, e biblioteca OpenSSL e a plataforma CUDA.

Execução: a execução do algoritmo por meio do script foi efetuado corretamente.

Validação: foi encontrada por meio de funções internas a integridade dos dados coletados.

6 ANÁLISE E INTERPRETAÇÃO

6.1 TESTE DE NORMALIDADE

Em seguida, apresentados na Tabela 3, o teste de normalidade das seguintes variáveis, a fim de corroborar com o comportamento dos dados, isto é, verificar se esses se ajustam a uma distribuição normal.

Intervalo de confiança para 99%. = 0.01.

A hipótese estabelece que os dados são normais, de modo que Sig ou P-valor devem ser maiores que = 0,01.

Tabela 3 – Teste de normalidade.

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Est[b]	gl	Sig.	Est[b]	gl	Sig.
Tempo Execução	,243	609	,00	,830	609	,000
Bloco Cifrado	,204	609	,00	,867	609	,000
Tipo Processador	,342	609	,00	,637	609	,000

a. Correção de significância de Lilliefors.

b. Estadístico.

Conclusão: observar o P-value = 0, portanto, pode-se inferir que as variáveis acima não são normais.

6.2 TESTE PARA DADOS NÃO-NORMAL

A Tabela 4 mostra os resultados do teste de Mann-Whiteney para duas amostras independentes, sendo o objetivo de este ensaio tentar encontrar diferenças entre as amostras. Como mostrado Sig ou P-value = 0, portanto, temos de rejeitar a hipótese nula de amostras iguais à Ho.

Tabela 4 – U de Mann-Whiteney

	Tempo Execução (Seg)
U de Mann-Whitney	,000
W de Wilcoxon	46360,000
Z	-21,354
Sig.asintótica bilateral)	,000

a. Variável de agrupação: tipo Processador.

Aceitamos a seguinte hipótese:

$H_{1tiempo}$: o tempo de execução do paradigma sequencial (CPU) é diferente do tempo de execução do paradigma paralelo (GPU).

7 AMEAÇAS DO EXPERIMENTO

Nesta seção as possíveis ameaças são tipificadas no processo de experimentação, bem como, as soluções possíveis são mencionadas, a fim de mitigar o impacto destas.

Ameaças: disponibilidade de recursos computacionais em tempo de execução do algoritmo.

Memória RAM.

Recursos do processador CPU.

Recursos do processador gráfico GPU.

Processos execução em segundo plano.

Funcionamento normal da plataforma CUDA.

Dependências das bibliotecas em tempo de execução.

Soluções possíveis:

A fim de mitigar as potenciais ameaças descritas, prossegue estabelecimento das possíveis soluções, a fim de aumentar a percentagem de sucesso da implantação do algoritmo.

O sistema básico para que possamos realizar o experimento, corresponde ao sistema operacional Linux Ubuntu 14 LTS. Este foi utilizado para obter o controle total do sistema em baixo nível, ou seja, a gestão e administração de memória e processos. Com essa diretriz principal, minimizamos as ameaças potenciais na implementação do algoritmo de criptografia.

A execução do experimento realizará em modo consola, sem a intervenção do ambiente gráfico, a fim de minimizar os recursos computacionais usados pelo sistema operacional.

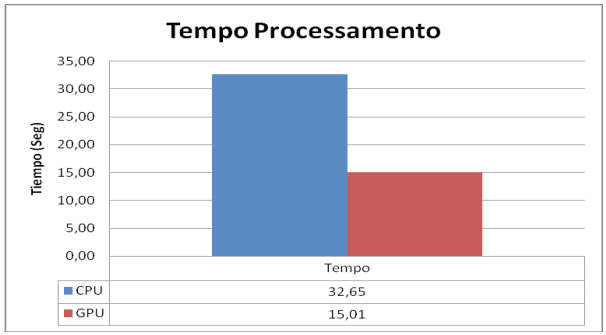
8 RESULTADOS

Analisando os resultados acima chegamos à seguinte conclusão: A implementação do algoritmo de criptografia AES-128-ECB na arquitetura (GPU) apresenta melhores resultados que a arquitetura sequencial (CPU).

8.1 TEMPO DE EXECUÇÃO MÉDIO DO CPU VS GPU

Esta seção, por meio da Figura 8, mostra o resultado do tempo médio de processamento do algoritmo AES-128-ECB executado em modo sequencial e paralelo, onde é avaliado no tamanho de bloco seguinte [16-32-64- 96-128]k. Podemos concluir que o processamento paralelo superou o processamento sequencial de uma análise do desempenho global.

Figura 8 – Tempo médio de processamento



9 CONCLUSÃO

Neste trabalho, foi apresentada uma descrição de como abordar o desenho e desenvolvimento de um experimento controlado, definindo como ponto de partida um objetivo claro e tangível, a partir de um contexto de análise de eficiência algorítmica do algoritmo de criptografia simétrica AES-128-ECB, executados na plataforma sequencial (CPU) e na plataforma paralela (GPU).

Com o processamento de dados por meio de ferramentas estatísticas, podemos dizer que a amostra apresenta prova suficiente, no qual o tempo de execução do paradigma paralelo (GPU) é significativamente mais eficiente que o paradigma de execução seqüencial (CPU).

8.1 TRABALHO FUTURO

Fazer um estudo da eficiência energética do algoritmo AES-128-ECB, a fim de comparar a relação entre o tempo de execução do algoritmo e o consumo de energia que este demanda.

Fazer um estudo de emissões eletromagnéticas geradas a partir do processo de cifragem e decifragem de algoritmo de criptografia AES-128-ECB.

REFERÊNCIAS

AGUIRRE, J R. **Aplicaciones Criptográficas**. 2.ed. junio de 1999, España: Universidad Politécnica de Madrid, 1999.

BASILI, V., ROMBACH, H. **The tame Project**: Towards Improvement – Oriented. 1988. Disponível em: <https://www.cs.umd.edu/~basili/publications/technical/T89.pdf>. Acesso em: 1 Jul. 2015.

COOK, T.; CAMPBELL, D. **Quasi-experimentation design and analysis issues for fields settings**. Chicago: Rand McNally, 1979.

FLYNN, M. Some computer organizations and their effectiveness, department of computer science, the johns hopkins university, baltimore, md. 21218**, 1972.**

HENNESSY, JOHN.; PATTERSON, D. **Computer architecture a quantitative approach**. Morgan Kaufmann, 1996.

LIMA, F.; MORENO E.; MAGALHÃES, F.; DIAS, W. Estudo do Consumo de Energia do Algoritmo Criptográfico AES no simulador da arquitetura ARM Sim-Panalyzer. Conference, **XIII Simpósio em Sistemas Computacionais de Alto Desempenho** (WSCAD-WIC), Petrópolis-RJ-Brazil, 2013.

LUEBKE, D.; HARRIS, M.; KRÜGER, J.; PURCELL, T.; GOVINDARAJU, N.; BUCK, I.; WOOLLEY, C.; LEFOHN, A. **GPGPU**: General Purpose Computation on raphics Hardware, Siggraph 04 ACM. Course Notes Article, n.33, ACM New York, NY, USA 2004.

OPENCL, KHRONOS GROUP. Disponível em: <https://www.khronos.org/opencl/.,2015>. Acesso em: 1 abr. 2015.

OPENSSL PROYECT. Disponível em: <https://www.openssl.org/about/>. Acesso em: 20 jun. 2015.

PRESSMAN, R. **Ingeniería del software**: un enfoque práctico. 6.ed. México: MacGraw-Hill, 2005.

RODRÍGUEZ, R.; MENÉNDEZ I.; GONZÁLEZ, F. **Procesadores vectoriales comerciales arquitectura y tecnología de computadores**. 4º curso de Ingeniería en Informática. EPSIG, Universidad de Oviedo, Campus de Viesques, E-33271. Gijón, Asturias, 2010.

SANDERS, J.; KANDROT E. **CUDA by example**: An introduction to general purpose GPU programming. Addison-Wesley. Pearson educación, 2010.

SOLINGEN, R.; BERGHOUT, E. **The Goal/Question/Metric Method**: A practical guide for quality improvement of software development. London: McGraw-Hill 1999.

TANENBAUM, A. **Redes de computadoras**. 5.ed. México: Pearson educación, 2003.

U.S. FEDERAL INFORMATION PROCESSING STANDARDS, Publication 197: Advanced Encryption Standard AES, National Institute of Standards and technology, november 2001. Disponível em: <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>. Acesso em: 10 maio 2015.

WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M.; REGNELL, B.; WESSLÉN. **A experimentation in Software Engineering**. Springer-Verlag Berlin Heidelberg, 2000.

Recebido em: 12 de Abril 2015
Avaliado em: 18 de Abril 2015
Aceito em: 4 de Maio de 2015

1. Mestrando em Ciências da Computação – Universidade Federal de Sergipe, Campus São Cristóvão (UFS). E-mail: galvan.gabriel@gmail.com