UDC: 004.72.057.4 Info M str. 4-10

EKSPERIMENTI SA UPOTREBOM GPU UREĐAJA KAO KRIPTOGRAFSKIH AKCELERATORA
EXPERIMENTS WITH THE USE OF GPU DEVICES AS CRYPTOGRAPHIC ACCELERATORS

Boris Damjanović, BLC

REZIME: Moderna kriptografija se u posljednje vrijeme u velikoj mjeri oslanja na GPU uređaje. Blockchain tehnologija i rudarenje velikog broja kriptovaluta se više ne može zamisliti bez ogromne procesne moći ovih uređaja. Ipak, GPU jedinice koje se koriste za rudarenje su mnogo moćnije od onih koje se ugrađuju u uređaje koji pripadaju segmentu potrošačke elektronike, poput laptop i desktop računara srednjeg ili nižeg ranga cijena. U ovom tekstu predstavljena je ideja korišćenja GPU jedinica nižeg cjenovnog ranga kao kriptografskih akceleratora koji su u stanju da u velikoj mjeri ubrzaju paralelno izvršavanje AES algoritma.

KLJUČNE REČI: paralelizacija, GPU, bezbjednost i zaštita, enkripcija podataka

ABSTRACT: Modern cryptography today relies largely on processing power of GPU devices. Blockchain technology and methods of mining for a large number of crypto valutes can no longer be imagined without the enormous process power of these devices. However, GPU devices that are used for mining are much more powerful than those that are built into devices that belong to the consumer electronics segment, such as laptops and desktop computers of the middle or lower price range. This paper presents the idea of using low-cost GPU units as cryptographic accelerators that are able to greatly accelerate the parallel execution of the AES algorithm.

KEY WORDS: parallelisation, GPU, security and protection, data encryption

I. UVOD

Moderne GPU jedinice posjeduju višestruko veći broj jezgara i niti od klasičnih CPU jedinica, dok istovremeno poprimaju sve više karakteristika procesora opšte namjene. Zbog toga su mnogi istraživači i razvojni timovi počeli da istražuju mogućnsti da se snaga grafičkih procesora iskoristi za aplikacije opšte namjene. Istovremeno sa razvojem GPU (Graphic Processing Unit) jedinica počeo je razvoj razvojnih okruženja poput OpenCL (Open Computing Language) ili CUDA (Compute Unified Device Architecture). CUDA je platforma za paralelnu obradu podataka i programiranje kompanije NVIDIA koja koristi NVIDIA grafičke procesore za rješavanje mnogih kompleksnih problema opšte namjene [1]. Open Computing Language (OpenCL) je prvi otvoren i besplatan standard za paralelno više-platformno programiranje modernih procesora poput GPU i višejezgrenih CPU procesora [2].

Fokus ovog rada je dobijanje odgovora na pitanje u kojoj mjeri jeftiniji računarski uređaji koji pripadaju srednjem i nižem cjenovnom rangu mogu da profitiraju na osnovu povećanja procesne moći GPU uređaja kada je u pitanju kriptografija. Ovaj tekst prikazuje rezultate nastavka istraživanja koje je počelo u [4].

II. PRIKAZ RADOVA KOJI SU VEZANI ZA PARALELIZACIJU ALGORITMA AES I IMPLEMENTACIJU NA GPU UREĐAJIMA

U [3] i [4] istražuje se mogućnost modifikacije kriptografskih načina rada radi paralelizacije izvršenja ovog algoritma. U nastavku poglavlja će biti prikazani neki radovi u kojima je istraživana mogućnost paralelizacije algoritma AES, te mogućnost da se takve ideje primjene na različitim GPU uređajima.

U svom istraživanju predstavljenom u [5] autor razmatra mogućnosti implementacije algoritma AES pomoću, u to doba tradicionalnog, pristupa koji je zasnovan na OpenGL biblioteci kao i na implementaciji pomoću NVIDIA CUDA platforme, te utvrđuje benefite koji se postižu korišćenjem nove arhitekture.

Postojanje različitih kripotgrafskih načina rada je već poslužilo kao ideja za paralelizaciju izvršavanja pojedinih algoritama. Prema [6] blokove C1, C2,... moguće je šifrovati u isto vrijeme pa je zbog toga CTR način rada moguće paralelizovati.

Brzi razvoj GPU (Graphic Processing Unit) jedinica kao i korisničkih okruženja poput CUDA kompanije NVidia ili OpenCL doveo je do različitih pokušaja paralelizacije AES algoritma pomoću CTR načina rada. Tako prema [7] autori najprije povećavaju veličinu bloka u odnosu na standardom definisani AES algoritam, a zatim koriste grubu (coarse grained) granularnost za paralelizovanje algoritma.

S druge strane autori u rješenju prikazanom u [8] koriste finu (fine grained) granularnost i interni paralelizam svake runde tako što nezavisno manipulišu sa 4 32-bitne riječi (T-riječi) koje se javljaju u svakoj AES-ovoj rundi. U ovom istraživanju implementacije AES-CTR algoritma na CUDA platformi autori tvrde da postižu ubrzanje od oko 14 puta u odnosu na OpenSSL implementaciju koja se oslanjala na upotrebu običnog centralnog procesora (CPU) računara.

U [9] autori modifikuju već postojeće open source rješenje koje je ponudio Can Berk Güder [10] da bi ubrzali izvršenje pomoću grube (coarse grained) granularnosti i CTR načina rada. U [11] autori koriste CUDA arhitekturu i CTR način rada sa WAES bibliotekom. Prema [11] WAES biblioteka je prva biblioteka koja je koristila predprocesiranje ili prethodnu obradu [6] za povećanje brzine obrade podataka. Kako je kriptografska obrada u šifrovanju poruke M kod CTR načina rada nezavisna od M [6] jer se šifruje brojač (counter), može se koristiti i predprocesiranje ili prethodna obrada za povećanje brzine. Ovu osobinu WAES biblioteke autori nazivaju špekulativna enkripcija [11].

Istraživanje uticaja alocirane GPU memorije i granularnosti programskih niti (threads) na performanse izvršavanja AES algoritma u ECB režimu rada koje su obavili [12] pokazalo je da najbolje performanse pokazuju implementacije koje obrađuju 16 bajta u jednoj niti, dok implementacije koje obrađuju 8 bajta u dvije ili 4 bajta u 4 niti pokazuju lošije performanse zbog po-

trebe sinhronizacije niti, a pored toga im je potrebna i dijeljena memorija za rad.

U toku istraživanja koje je predstavljeno u [13], autori istražuju mogućnost poboljšanja paralelizacije algoritma AES u ECB načinu rada pažljivim aranžiranjem podataka u memoriji grafičke karte. Prema njihovim navodima, oni uspijevaju da postignu određena ubrzanja u odnosu na jednonitnu sekvencijalnu implementaciju AES algoritma, kao i u odnosu na implementaciju koja na CPU paralelno obrađuje podatke pomoću 4 niti.

U [14] autori su implementirali četiri različita blokovska algoritma, a zatim su procjenjivali njihovu efikasnost na dva CUDA GPGPU koji se oslanjaju na Fermi i GT200 arhitekturu.

Autori u [15] koriste OpenMP (Open Multiprocessing) API da bi uporedili mogućnosti paralelizacije izvršenja na različitim testnim platformama sa različitim CPU i GPU jedinicama. Prema prikazanim rezultatima, moguće je postići značajna poboljšanja performansi kada se koristi paralelna obrada na GPU jedinicama.

Autori u [16] koriste činjenicu da mnogi sistemi za upravljanje bazama podataka pristupaju podacima na nivou stranice različite veličine, pa istražuju mogućnost primjene paralelne obrade pojedinačnih stranica na GPU jedinicama.

U [17] autori su implementirali algoritam AES na tri različite GPU arhitekture (Kepler, Maxwell i Pascal) a potom su poredili postignute performanse na pojedinim uređajiima.

III. PREGLED STANDARDOM DEFINISANOG ALGORITMA AES

Algoritam AES nastao je na osnovu algoritma pod imenom Rijndael, koji je bio pobjednik javno organizovanog tamičenja za izbor slijedećeg kriptografskog algoritma vlade SAD. U januaru 1997. godine, NIST (National Institute of Standards and Technology) objavio je početak inicijative za razvoj algoritma koji će postati sljedeći kriptografski standard američke vlade [18]. Ovaj algoritam je, zajedno sa još 15 kriptografskih algoritama koji su učestvovali u takmičenju, podvrgnut intenzivnim višegodišnjim testiranjima tokom kojih su njegovi konkurenti jedan za drugim otpadali. Krajem 2000. godine, NIST je proglasio pobjednika – algoritam zvan Rijndael (po imenima pronalazača Vincent Rijmen i Joan Daemen). Rijndael je blokovski kriptografski algoritam sa varijabilnom dužinom bloka i varijabilnom dužinom ključa. Dužina bloka i dužina ključa mogu biti nezavisno navedene kao umnošci broja 32, s tim da je najmanji dozvoljeni umnožak 128, a najveći dozvoljeni 256 bita [18].

Vlada SAD je dio Rijndaela koji je prihvatila nazvala Advanced Encryption Standard (AES). Algoritam AES, kako je opisano u [19], operiše nad 128-bitnim blokovima podataka (16 bajta) korišćenjem ključeva dužine 128, 192 i 256 bita. Ulaznih 16 bajta su organizovani u 4x4 matricu pod nazivom Stanje (State). Operacija šifrovanja se izvodi cikličnim ponavljanjem transformacija pod nazivima SubBytes(), ShiftRows(), MixColumns() i AddRoundKey() u okviru 10, 12 ili 14 rundi sa različitim dužinama ključeva. U procesu dešifrovanja koriste se inverzne transformacije InvSubBytes(), InvShiftRows() i InvMixColumns(), dok se transformacija AddRoundKey() zadržava u neizmijenjenom obliku. Različite operacije koje se provode tokom jedne Rijndaelove runde mogu se objediniti u svega nekoliko pretraga kroz unaprijed pripremeljene tabele (T-tabele), čime postaju moguće veoma brze implementacije na procesorima koji imaju 32 bita ili više [20].

Za sve navedene implementacije veoma važna rutina koja se mora realizovati je rutina za ekspanziju ključeva. U standardnom algoritmu AES, dužine inicijalnog ključa su 128, 192 i 256 bita. Da bi AES algoritam bio funkcionalan, inicijalni ključ se mora u funkciji ekspanzije ključeva razviti do određenog broja bajta, koji zavisi od tipa algoritma (AES-128, AES-192 ili AES-256). Broj rundi u algoritmu zavisi od dužine ključa i obrnuto. Ako dužinu inicijalnog ključa u bitima predstavimo sa Nbk, a broj rundi sa Nr, imamo [4]:

- Nr = 10 kada je Nbk = 128,
- Nr = 12 kada je Nbk = 192,
- Nr = 14 kada je Nbk = 256.

IV. CUDA – PLATFORMA ZA PARALELNO PROGRAMIRANJE

Kako je u [4] opisano, CUDA je paralelna kompjuterska arhitektura opšte namjene sa paralelnim programskim modelom i skupom instrukcija koji koriste mogućnosti NVIDIA grafičkih procesora za rješavanje kompleksnih problema na efikasniji način nego što ih je moguće riješiti pomoću CPU-a [1].

Prvi DirectX 10 grafički procesor pod nazivom NVIDIA GeForce 8800 GTX koji je predstavljen 2006. godine bio je istovremeno i prvi GPU koji je bio zasnovan na NVIDIA CUDA arhitekturi. Za razliku od prethodnih generacija grafičkih procesora, CUDA arhitektura je dozvoljavala da se programski pristupa svakoj pojedinoj aritmetičko logičkoj jedinici (ALU) i da se na njima izvode izračunavanja opšte namjene [21]. Svega nekoliko mjeseci nakon predstavljanja NVIDIA GeForce 8800 GTX procesora NVIDIA je objavila i poseban CUDA C kompajler namijenjen za programiranje CUDA uređaja.

Do 2013 godine [22] su se pojavile tri arhitekture CUDA mikroprocesora:

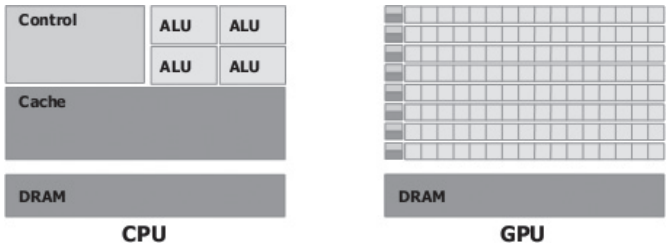
- Tesla, koja je predstavljena 2006. godine u GeForce 8800 GTX (G80) procesoru,
- Fermi, koja je predstavljena 2010. godine u GeForce GTX 480 (GF100) procesoru,
- Kepler, koja je predstavljena 2012. godine u GeForce GTX 680 (GK104) procesoru,

Nakon toga, slijedile su još tri arhitekture [23]:

- Maxwell arhitektura, koja je predstavljena 2014. godine u modelima GeForce700 serije,
- Pascal arhitektura, koja je predstavljena u aprilu 2016. godine u Tesla P100 (GP100) procesoru,
- Volta arhitektura, koja je predstavljena u junu 2017. godine u GV100 procesoru.

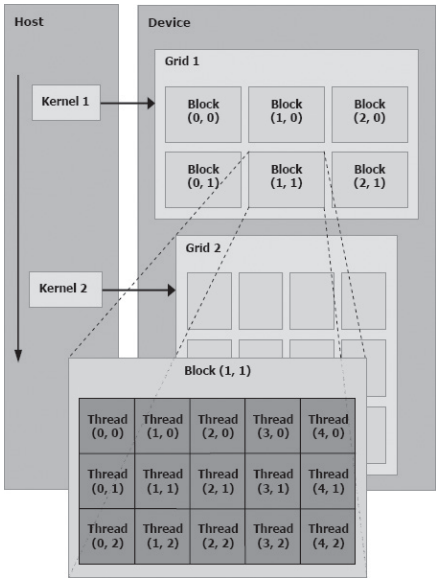
Današnji grafički procesori su evoluirali u visoko paralelizovane, višenitne i višejezgrene procesore sa ogromnom računarskom snagom i sa ogromnim memorijskim propustnim opsegom. Prema [1], osnovna razlika između običnog procesora opšte namjene (CPU) i grafičkih procesora (GPU) je u tome što su grafički specijalizovani za intenzivna, visoko paralelizovana izračunavanja kakva su grafička izračunavanja, te su

zbog toga dizajnirani tako da je više njihovih tranzistora posvećeno obradi podataka nego keširanju i kontroli toka, kako je vidljivo sa sljedeće slike:



Slika 1: *Razlike između CPU i GPU [1]*

CUDA je platforma za paralelnu obradu podataka i razvojno okruženje koje je kreirano u kompaniji NVIDIA [23]. Svaki CUDA grafički procesor ima N multiprocesora (MP) i globalnu memoriju. Svaki multiprocesor ima N skalarnih procesora (SP), dijeljenu memoriju i skup 32-bitnih registara. Svaki multiprocesor je dizajniran pomoću posebne arhitekture koja se naziva SIMT (Single Instruction, Multiple-Thread) koja mu omogućava da izvršava hiljade niti istovremeno.



Slika 2: *CUDA grafički procesor: Grid-Block-Thread prikaz [1]*

Paralelno izvršavanje se u CUDA C implementira posebnim funkcijama koje se nazivaju kernel funkcije (kernels) koje se, kada su pozvane, izvršavaju paralelno N puta od strane N niti. Svaka nit ima vlastiti ID pomoću kojeg se referencira. Radi referenciranja, niti se grupišu u blokove, a blokovi u gridove. Multiprocesor kreira i izvršava niti u grupama od po 32 paralelne niti koje se nazivaju warpovi. Ovakva arhitektura se u [23] naziva SIMT (Single Instruction Multiple Threads) arhitektura. Broj niti po bloku treba odabrati tako da bude umnožak veličine warpa. Za one blokove kod kojih broj niti nije umnožak broja 32, neki warpovi neće biti do kraja popunjeni aktivnim nitima [22].

Karakteristika prikazane arhitekture je da, kada je koristimo za izračunavanja opšte namjene, ona neće pokazati svu svoju snagu sve dok joj ne damo dovoljno podataka.

Sve navedeno dovodi do povećanja interesovanja za GPU procesore u istraživanjima koja se bave poboljšanjem performansi algoritma AES.

V. IMPLEMENTACIJE KORIŠĆENE U ISTRAŽIVANJU

Originalna rješenja kriptografskih modula za jednojezgrene i višejezgrene mikroprocesore koja su nastala u okviru istraživanja [24], [25], [3] i [4] oslanjaju se na standard FIPS 197 koji opisuje AES algoritam kao i na modifikovane ideje autora algoritma te na određene ideje o korišćenju različitih kriptografskih režima rada (modes of operation) radi paralelizacije njegovog izvršenja.

U okviru istraživanja, radi generisanja realnih podataka na različitim platformama, implementiran je odgovarajući aplikacija. Da bi mogli da poredimo uticaj i slučajeve kada treba koristiti paralelnu ili serijsku implementaciju, ovdje su izdvojene jednonitna C++ implementacija i paralelna CUDA implementacija.

Jednonitna C++ implementacija zasnovana je na idejama autora algoritma, korišćenjem T tabela [18] i implementacijama dr Briana Gladmana [26], a za njegovu izradu korišćen je Microsoft Visual C++ kompajler u sastavu Microsoft Visual Studija 2012. Navedena aplikacija koristi 128, 192 i 256-bitnu enkripciju i u potpunosti poštuje standard definisan dokumentom FIPS-197.

Implementacija koja vrši šifrovanje pomoću AES algoritma koji je zasnovan na AES New Instructiuons (AES-NI) skupu instrukcija razvijen je [4] na osnovu kombinacije prethodno pomenutog rada na razvoju originalnih modula, teksta [27] i Intelovog White-Paper uputstva [28], te na osnovu izvornog koda Intelove biblioteke Intel® AES sample library [29] i na osnovu izvornog koda biblioteke Botan [30].

Pored serijske implementacije, implementirano je još jedno originalno paralelno rješenje koje se oslanja na CUDA arhitekturu i kombinaciju CUDA C kompajlera te kompajlera Visual C++ 2012. Navedeni izvorni kod je napisan u skladu sa sintaksom i preveden uz pomoć CUDA C kompajlera tako da se paralelno izvršava na velikom broju jezgara pri čemu svako jezgro obrađuje po jedan blok podataka (Stanje). Svako pojedino jezgro ponovo izvršava kod koji je baziran na idejama autora algoritma, korišćenjem T tabela [18]. Ova implementacija se u radu oslanja na postojanje specifičnog hardverskog sklopa, odnosno CUDA grafičkog procesora.

VI. PARALELNA CUDA IMPLEMENTACIJA ZA ŠIFROVANJE I DEŠIFROVANJE ALGORITMOM AES

Originalna implementacija za šifrovanje i dešifrovanje pomoću algoritma AES koja je predstavljena u [4] a koja je zasnovana na CUDA arhitekturi i na CUDA kompajleru oslanja se na prethodno razvijen jednonitni C++ modul na kojem je primijenjeno ubrzanje u skladu s idejama autora algoritma (T-tabele).

Izvorni kod ovog modula je napisan u skladu sa sintaksom i preveden uz pomoć CUDA C kompajlera tako da se paralelno izvršava na velikom broju jezgara, pri čemu svako jezgro obrađuje po jedan blok Stanje. Navedeni modul je implementiran tako da se paralelno izvršava u ECB režimu rada.

U implementaciji opisane paralelizacije korišćen je Visual Studio 2012 C/C++ kompajler kao i C kompajler koji se isporučuje sa Nvidia Cuda v6.0 toolkitom. Navedena implementacija zahtijeva da se najprije S-Box, četiri T-Boxa, IV, ključ i bafer sa otvorenim (plain) tekstom odnosno šifratom prenesu u globalnu i dijeljenu memoriju uređaja (device), a zatim se pomoću poziva kernel funkcija vrši paralelno šifrovanje pojedinih dijelova otvorenog teksta. Implementirane su dvije kernel funkcije, jedna pod nazivom processBlock za šifrovanje i druga pod nazivom invProcessBlock dešifrovanje algoritmom AES.

U toku izrade neke aplikacije koja vrši šifrovanje ili dešifrovanje, mora biti pripremljen algoritam koji može da obrađuje kako kraće, tako i veoma dugačke datoteke. Veoma dugačke datoteke ne mogu odjednom biti učitane u operativnu memoriju, već se one obično u memoriju učitavaju u fragmentima odgovarajuće dužine, tako da se u memoriju učitava i obrađuje fragment po fragment. U aplikaciji se ovi fragmenti podataka smještaju u nizove (bafere) željenog tipa podatka, pa se ovi baferi jedan za drugim obrađuju i po potrebi ponovo snimaju u neku drugu datoteku. Kako je veličina bloka Stanje algoritma AES 16 bajta, svaki ovakav bafer se u nekoj jednonitnoj implementaciji obrađuje u fragmentima dužine 16 bajta. Sa aspekta performansi, poželjno je da dužina fragmenata bude što veća, jer se na taj način smanjuje vrijeme potrošeno na komunikaciju sa hard diskom i na prenos podataka preko sabirnice.

Kada se kriptografska obrada podataka vrši pomoću GPU jedinice, osim već opisanog slučaja komunikacije sa hard diskom i sabirnicom, postoje još dvije kritične tačke na kojima je moguće izgubiti vrijeme ili ostvariti ogromno poboljšanje performansi.

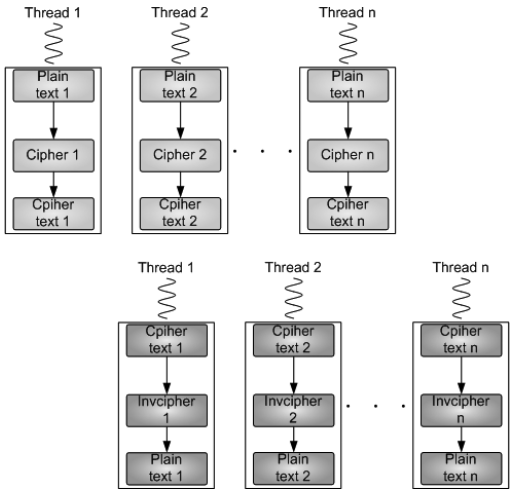
1. Na performanse nekog CUDA programa u velikoj mjeri utiče ukupan broj niti i blokova koji se paralelno izvršavaju kao i raspoloživi broj registara po jednoj niti. Broj blokova i warpova koji su alocirani u svakom multiprocesoru za pojedini poziv kernela zavisi od broja gridova i blokova koji se navode pri pozivu kernela (Execution Configuration), memorijskih resursa multiprocesora i količine resursa koji su kernelu potrebni za rad. Da bi programeri mogli lakše da odaberu optimalnu veličinu bloka, u sastavu CUDA SDK se isporučuje CUDA Occupancy Calculator.
2. CUDA programi moraju da prenesu sve varijable i podatke koje će obrađivati iz memorije opšte namjene računara u memoriju uređaja. Rezultati obrade se nakon toga moraju vraćati nazad u memoriju računara. Svaki prenos podataka u memoriju uređaja degradira performanse, tako da je sa aspekta performansi mnogo bolji jedan veliki transfer nego više malih koji prenose istu količinu podataka.

Kao testna platform4 korišćeni su

1. Dell Inspiron N5110 računar sa procesorom Intel Core i7-2630QM i 6 GB RAM memorije, instaliranim 64-bitnim Windows 7 Ultimate operativnim sistemom, na kojem je kao druga GPU jedinica bila instalirana NVidia GeForce GT 525M i
2. Toshiba Satellite L50-A-1D5 računar sa Intel Core i7-4700MQ i 8 GB RAM memorije, instaliranim 64-bitnim Windows 8.1 operativnim sistemom, na kojem je kao druga GPU jedinica bila instalirana NVidia GeForce GT 740M.

CUDA arhitektura (i upošte arhitektura GPU-a) je slična arhitekturi superkompjutera. Sam SDK je kreiran sa fokusom na paralelizacju. Najvažnija funkcija u CUDA SDK je Kernel, koja može da pokreće hiljade niti paralelno. Svaka kernel funkcija se pokreće sa dodatnim parametrima, od kojih su najvažniji broj blokova i broj niti po jednom bloku. Npr. poziv kernel funkcije cipherBytes<<<16, 256>>> pokreće 16x256 = 4096 niti paralelno, pri čemu će svaka nit u slučaju AES algoritma obraditi isti kod ali sa drugim podacima (npr. sa drugim otvorenim tekstom i ključem). Treba primijetiti da je CUDA blok u stvari blok pokrenutih niti i da ga treba razdvojiti od kriptografskog bloka Stanje.

Implementirano rješenje funkcioniše u skladu s CUDA/GPU arhitekturom, tako da kada se pokrene jedna kernel funkcija, u stvari se paralelno obrađuju podaci pomoću hiljada niti (Slika 56), pri čemu svaka nit obradi jedan AES-ov blok podataka (jedno Stanje/State). Samo šifrovanje je ovdje munjevito brzo, ali se određena količina vremena potroši na kreiranje i inicijalizaciju kernel funkcije te na transfer podataka do memorije CUDA uređaja.



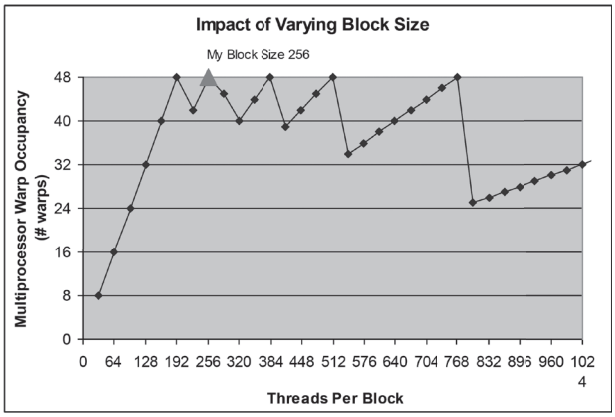
Slika 3: *Blok šema paralelne obrade podataka pomoću n-niti CUDA procesora [4]*

Ovakav coarse grain dizajn može se implementirati na grafičkim procesorima na takav način da se S-Box, četiri T-Boxa, IV, ključ i otvoreni (plain) tekst odnosno šifrat prenesu u globalnu i dijeljenu memoriju uređaja (device), a zatim se pomoću poziva kernel funkcija vrši paralelno šifrovanje pojedinih blokova otvorenog teksta. U [2] je prikazano slično rješenje koje koristi OpenCL okruženje (framework) i koje u memoriju uređaja prenosi samo S-Box.

Još jedna karakteristika CUDA platforme je da neće svaka kombinacija broja blokova i niti dati isti rezultat. Kako broj registara koji se koristi u jednoj kernel funkciji, veličina dijeljene memorije i verzija grafičkog procesora u saradnji sa veličinom bloka utiču na brzinu obrade, kompanija NVidia je objavila CUDA Occupancy Calculator koji se može koristiti za optimizaciju izvršenja.

Da bi pomoću CUDA Occupancy Calculatora mogao biti odabran optimalan broj niti i blokova koji obezbjeđuje najbolje performanse, mora biti utvrđen broj registara koji će biti korišćen od strane jednog izvršenja funkcije kernela u okviru jed-

ne niti. Ovu informaciju je pomoću Nvidia Cuda v6.0 moguće dobiti jedino u toku kompajliranja programa pomoću prekidača „--ptxas-options=-v“. Kako navedena implementacija za svoja jezgra zahtjeva po 18 registara, za NVidia GeForce GT 525M grafičku kartu koja zadovoljava 2.1 Compute Capability sa 49152 bajta dijeljene memorije, kalkulator je ponudio vrijednosti od 192, 256, 384, 512 i 768 niti po bloku, za koje će iskorišćenost svakog multiprocesora biti maksimalna i pri kojima se postiže 48 aktivnih warpova po multiprocesoru.



Slika 4: *CUDA Occupancy Calculator rezultati za GPU koji je kompatibilan sa verzijom 2.1* ***[4]***

CUDA Occupancy Calculator daje još dosta dodatnih podataka o CUDA arhitekturi. Tako se za GPU koji je kompatibilan sa verzijom 2.1 mogućnosti obrade podataka (CUDA compute capability) po jednom multiprocesoru može koristiti maksimalno 8 blokova.

Physical Limits for GPU Compute Capability:	**2.1**
Threads per Warp	32
Warps per Multiprocessor	48
Threads per Multiprocessor	1536
Thread Blocks per Multiprocessor	8
Total # of 32-bit registers per Multiprocessor	32768
Register allocation unit size	64
Register allocation granularity	warp
Registers per Thread	63
Shared Memory per Multiprocessor (bytes)	49152
Shared Memory Allocation unit size	128
Warp allocation granularity	2
Maximum Thread Block Size	1024

Tabela 1: *Isječak podataka CUDA Occupancy Calculatora za GPU koji je kompatibilan sa verzijom 2.1* ***[4]***

VII. IMPLEMENTACIONA ANALIZA

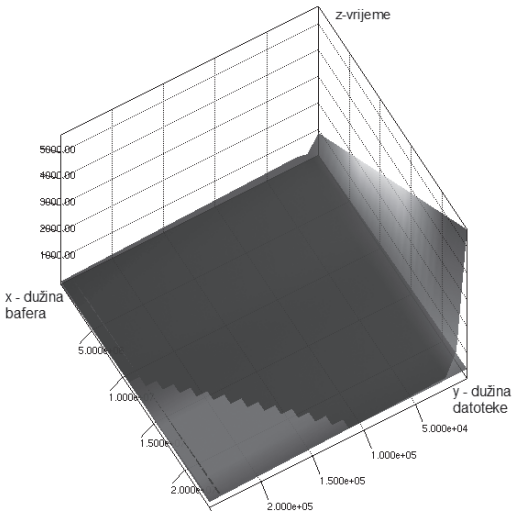
Trenutno su i jeftinije Nvidia CUDA GPU jedinice sa npr. svega dva multiprocesora i 384 jezgra u stanju da izvršavaju do 4096 niti paralelno. Zbog toga se sa svakim novim uposlenim warpom povećava iskorišćenje ovakvih uređaja. Kada je u pitanju kriptografija, ova činjenica ima dvostruko značenje koje se manifestuje kroz slijedeća dva pitanja:

1. Kolika je ukupna dužina toka ili datoteke koju obrađujemo pomoću ovakvog GPU uređaja, odnosno da li je ona dovoljna da se zaista uposle sve raspoložive niti?
2. Kolika je veličina fragmenata (bafera) koje koristimo za pribavljanje podataka odnosno za njihovo prebacivanje iz operativne u memoriju GPU uređaja? Ovo pitanje se dijeli na dva potpitanja:
   a. Ponavlja se pitanje količine podataka - da li je dužina bafera dovoljna da se uposle sve raspoložive niti?
   b. Da li je dužina bafera dovoljna da se izbjegnu gubici vremena koji nastaju prilikom prenosa podataka između operativne memorije računara i memogije GPU uređaja i na kreiranje i inicijalizaciju kernel funkcije?

Kako su u testovima korišćene NVidia GeForce GT 525M i NVidia GeForce GT 525M i GPU jedinice koje imaju po 2 multiprocesora, u implementaciji je korišćeno 16 blokova veličine po 256 niti svaki, odnosno po 4096 niti. Za predstavljanje rezultata korišćena je metoda mašinskog učenja pod nazivom M5’. Ova metoda, kako je to prikazano u [31] postiže dobre rezultate i na relativno malim skupovima testnih podataka. Uz pomoć ove metode, na osnovu malih skupova podataka moguće je dobiti veoma precizne prediktivne modele [4] koji precizno predviđaju vremena izvršenja pojedinih implementacija u određenim uslovima.

Na slici broj 1 prikazani su prediktivni modeli koji opisuju jednonitnu C++ implementaciju i paralelnu GPU implementaciju na Toshiba Satellite notebook računaru sa NVidia GeForce GT 740M GPU jedinicom (druga testna platforma). Modeli su za obje implementacije nastali obradom testnih podataka pomoću M5’ metode, dok su sami testni podaci nastali mjerenjem brzine šifrovanja, odnosno odgovarajućih vremena potrebnih da se datoteke različitih dužina šifruju u ECB modu 128 bitnom enkripcijom.

Na slici 1 je plavom bojom prikazan model koji opisuje paralelnu GPU implementaciju, a crvenom bojom model koji opisuje jednonitnu C++ implementaciju. Na navedenoj slici, na x osi je prikazana dužina bafera, na y osi dužina datoteke a na z osi vrijeme potrebno da se izvrši šifrovanje.



Slika 5: *Poređenje jednonitne C++ i GPU implementacije*

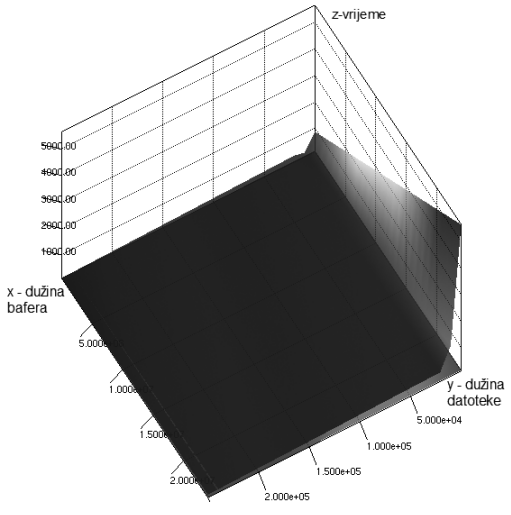
Rezultati testiranja GPU implementacije i odgovarajući prediktivni model pokazuju da postoji ogromna razlika u brzini izvršenja u zavisnosti od količine podataka koja se koristi u procesu šifrovanja i dešifrovanja, kako je prikazano na slici 1. U skladu sa ranijom diskusijom i dva postavljena pitanja, za male količine podataka paralelna GPU implementacija je veoma neefikasna.

Kako se povećava količina podataka koja se daje ovoj implementaciji, tako ona postaje sve brža i brža, sve dok konačno ne prestigne jednonitnu implementaciju u brzini izvršavanja. Treba reći da je ova situacija identična i na drugoj testnoj platformi na kojoj smo vršili mjerenja na osnovu kojih su nastajali prediktivni modeli. Takođe, treba reći i da se gotovo identični modeli dobijaju i u slučaju šifrovanja i dešifrovanja 192 bitnom i 256 bitnom enkripcijom.

Na slici 1 je vidljiva još jedna karakteristika GPU implementacije koja zaslužuje da je prokomentarišemo. Ako posmatramo ovu implementaciju, vrijeme potrebno da se obavi šifrovanje uvećava se za male bafere sa uvećanjem datoteke, kako je to prikazano na modelu duž y ose. Ovo uvećanje vremena je posljedica gubitaka koji nastaju prilikom prenosa podataka između operativne memorije računara i memogije GPU uređaja, kao i zbog gubitaka vremena prilkom kreiranja i inicijalizacije kernel funkcije.

Ako je odabran previše mali bafer za prenos podataka, sa povećanjem dužine datoteke uvećavaju se negativni efekti vezani za prenos podataka na relaciji operativna memorija – memorija GPU uređaja i broj kreiranja kernel funkcije.

Na slici broj 2 prikazani su prediktivni modeli koji opisuju jednonitnu AES-NI implementaciju i paralelnu GPU implementaciju na Toshiba Satellite notebook računaru sa NVidia GeForce GT 740M GPU jedinicom.



Slika 6: *Poređenje jednonitne AES-NI i GPU implementacije*

Na slici 2 je plavom bojom prikazan model koji opisuje paralelnu GPU implementaciju, a crvenom bojom model koji opisuje jednonitnu AES-NI implementaciju. Na navedenoj slici, ponovo je na x osi prikazana dužina bafera, na y osi dužina datoteke a na z osi vrijeme potrebno da se izvrši šifrovanje. Vidljivo je da AES-NI implementacija očekivano pokazuje bolje rezultate.

Na obje testne platformame procesori su imali ugrađen AES-NI skup instrukcija, tako da je na njima izvršeno poređenje paralelne GPU implementacije i implementacije zasnovane na AES-NI skupu instrukcija. Na obje testne platforme dobijeni su veoma slični rezultati, zbog čega su ovdje prikazani samo modeli kreirani na drugoj testnoj platformi.

Iako u datim testnim uslovima rezultati mjerenja i pripadajući modeli pokazuju da je AES-NI implementacija brža, rezultati CUDA GPU implementacije tek neznatno zaostaju za njim kako na prikazanoj slici, tako i u svim drugim slučajevima šifrovanja i dešifrovanja na korišćenim testnim platformama.

VIII. ZAKLJUČAK

U ovom tekstu fokus istraživanja je dobijanje odgovora na pitanje da li jeftiniji računarski uređaji koji pripadaju segmentu potrošačke elektronike mogu da koriste GPU uređaje kao kriptografske akceleratore. Iako odgovor na ovo pitanje nije jednoznačan, možemo izvući neke zaključke koji nam mogu pomoći pri donošenju odluke o tome kada treba koristiti navedene uređaje kao kriptografske akceleratore. Na prvom mjestu, na izbor akceleratora utiče količina podataka koja treba da bude obrađena. Ukoliko se npr. radi o šifrovanju ili dešifrovanju datoteka velikih dužina ili protočnih podataka, korišćenje GPU kao akceleratora može biti opravdano. Takođe, ako postoji potreba za rasterećenjem CPU jedinice, ponovo korišćenje GPU kao akceleratora može biti opravdano. U navedenimprimjerima korišćeni su relativno jeftini GPU uređaji, koji su zbog svojih karakteristika dali lošije rezultate od šifrovanja pomoću AES-NI skupa instrukcija. Ipak, ovo ne mora biti pravilo, jer današnji proizvođači računarskih sistema često ugrađuju mnogo moćnije i skuplje GPU uređaje u notebook i desktop računare. Očigledno je da CUDA GPU implementacija posjeduje veliki potencijal i da je pitanje da li je bolje koristiti GPU uređaje u stvari pitanje izbora komponenata od strane proizvođača računarskih sistema, kao i pitanje stvarnih potreba sistema na kojem se ovakav akcelerator nalazi.

REFERENCE

[1] NVIDIA Corporation, NVIDIA CUDA C Programming Guide, v9.1.85, (2018) available at: http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf, (2018)

[2] Gervasi O., Russo D., Vella F., The AES implantation based on OpenCL for multi/many core architecture, International Conference of Computational Science and Its Applications, ICCSA 2010, (2010), Fukuoka, Japan, pp. 129-134

[3] Damjanović, B. and Simić, D. Tweakable parallel OFB mode of operation with delayed thread synchronization, Wiley, Security and Communication Networks, Security Comm. Networks (2015), DOI: 10.1002/sec.1404.

[4] Damjanović, B. Adaptibilna primjena AES algoritma kod savremenih operativnih sistema (2016), Doktorska disertacija, Fakultet organizacionih nauka, Univerzitet u Beogradu

[5] Manavski, S.A. CUDA Compatible GPU as an Efficient Hardware Accelerator for AES Cryptography (2007), IEEE International Conference on Signal Processing and Communications,

[6] Lipmaa H., Rogaway P and Wagner D. Comments to NIST Concerning AES-modes of Operations: CTR-mode Encryption (2000), In Symmetric Key Block Cipher Modes of Operation Workshop, Baltimore, Maryland, USA

[7] Tran, N. P., Lee, M., Hong, S., & Lee, S. J. Parallel execution of AES-CTR algorithm using extended block size (2011), In C om putational Science an d Engineering (CSE), 2011 IEEE 14th International Conference on (pp. 191-198). IEEE.
[8] Di Biagio, A., Barenghi, A., Agosta and G. and Pelosi, G. (2009), Design of a Parallel AES for Graphics Hardware using the CUDA framework, IEEE International Symposium onParallel & Distributed Processing
[9] Jacquin L., Roca V., Roch J.L., Al Ali M., (2010), Parallel arithmetic encryption for high-bandwidth communications on multicore/GPGPU platforms, PASCO '10 Proceedings of the 4th International Workshop on Parallel and Symbolic Computation
[10] Berk Guder, C., AES on CUDA, (2009), dostupno na:http://github.com/cbguder/aes-on-cuda (mart 2018).
[11] Zola W.M.N., De Bona L.C.E., (2012), Parallel Speculative Encryption of Multiple AES Contexts on GPUs, Innovative Parallel Computing (InPar).
[12] Iwai K., Kurokawa T and Nisikawa N., (2010), AES Encryption Implementation on CUDA GPU and Its Analysis. In Proceedings of the 2010 First International Conference on Networking and Computing (ICNC '10). IEEE Computer Society, Washington, DC, USA, 209-214. DOI=http://dx.doi.org/10.1109/ICNC.2010.49
[13] Mei, C., Jiang H., Jenness J., (2010), CUDA-based AES Parallelization with Fine-Tuned GPU Memory Utilization, IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2010, Atlanta, USA
[14] Nishikawa, N,. Iwai K. and Kurokawa T., High-Performance Symmetric Block Ciphers on CUDA, 2011 Second International Conference on Networking and Computing (ICNC), Osaka, Japan pp. 221-227
[15] Ortega, J., Trefftz, H. and Trefftz, C., (2011), Parallelizing AES on multicores and GPUs, Mankato, USA, pp. 1-5
[16] Fazackerley S., McAvoy S.M. and Lawrence R., (2012) GPU accelerated AES-CBC for database applications. In Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC '12). ACM, New York, NY, USA, 873-878. DOI=http://dx.doi.org/10.1145/2245276.2245446
[17] Awadallah A. A., Fouad M. M., Dahshan H. and Mousa A. M. (2017) "High performance CUDA AES implementation: A quantitative performance analysis approach," 2017 Computing Conference, London, 2017, pp. 1077-1085. doi: 10.1109/SAI.2017.8252225
[18] Daemen J. and Rijmen V. (2002), The Design Of Rijndael, Springer-Verlag, Inc.
[19] FIPS197, Specification for the Advanced Encryption Standard (AES). (2001), Federal Information Processing Standards Publication 197, Available at: http://csrc.nist.gov/publications/
[20] Daemen J., Rijmen V., (1999), The Rijndael Block Cipher - AES Proposal (from original submission), NIST, Available at: http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf
[21] Sanders J. and Kandrot E. (2011), CUDA by Example, An Introduction to General- Purpose GPU Programming, Addison-Wesley
[22] Wilt N. (2013), The CUDA Handbook (A Comprehensive Guide to GPU Programming), Addison-Wesley
[23] NVIDIA Corporation, NVIDIA CUDA Toolkit Documentation v9.1.85, (2018) available at: http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf, (2018)
[24] Damjanović, B. and Simić, D. (2011), Comparative Implementation Analysis of AES Algorithm, Journal of Information Technology and Applications, Banja Luka
[25] Damjanović, B. and Simić, D. (2013), Performance evaluation of AES algorithm under Linux operating system, Proceedings Of The Romanian Academy, Series A, Mathematics, Physics, Technical Sciences, Information Science, Volume 14, Number 2, pp. 177–183
[26] Gladman, B., (2007), A Specification for Rijndael, the AES Algorithm, Available at: http://gladman.plushost.co.uk/oldsite/cryptography_technology/rijndael/
[27] Gueron S. (2009), Intel's New AES Instructions for Enhanced Performance and Security, Fast Software Encryption, Lecture Notes in Computer Science Volume 5665, pp 51-66, Springer Berlin Heidelberg
[28] Gueron, S. (2012), Intel Corporation, White Paper, Intel Advanced Encryption Standard (AES) New Instructions Set
[29] Rott J. (2011), Intel AESNI Sample Library, Available at: http://software.intel.com/en-us/articles/download-the-intel-aesni-sample-library
[30] Lloyd J. (2013), Botan C++ crypto library, Available at: http://botan.randombit.net/index.html
[31] Damjanović, B. i Simić D., (2014) Modeliranje performansi različitih implementacija algoritma AES pomoću M5′ metode, Infotech 2014, Medunarodna Konferencija i Izložba, Aranđelovac

Boris Damjanović, FSOM – Univerzitet Union – Nikola Tesla, Beograd i Banja Luka College
Kontakt: boris.damjanovic@blc.edu.ba
Oblasti interesovanja: zaštita računarskih sistema, kriptografija, informacioni sistemi, blockchain tehnologija.

infoM

UPUTSTVO ZA PRIPREMU RADA

1. Tekst pripremiti kao Word dokument, A4, u kodnom rasporedu 1250 latinica ili 1251 ćirilica, na srpskom jeziku, bez slika. Preporučen obim – oko 10 strana, single prored, font 11.
2. Naslov, abstakt (100-250 reči) i ključne reči (3-10) dati na srpskom i engleskom jeziku.
3. Jedino formatiranje teksta je normal, bold, italic i bolditalic, VELIKA i mala slova (tekst se naknadno prelama).
4. Mesta gde treba ubaciti slike, naglasiti u tekstu (Slika1...)
5. Slike pripremiti odvojeno, VAN teksta, imenovati ih kao u tekstu, radi identifikacije, u sledećim formatima: rasterske slike: jpg, tif, psd, u rezoluciji 300 dpi 1:1 (fotografije, ekranski prikazi i sl.), vektorske slike – cdr, ai, fh,eps (šeme i grafikoni).
6. Autor(i) treba da obavezno priloži svoju fotografiju (jpg oko 50 Kb), navede instituciju u kojoj radi, kontakt i 2-4 oblasti kojima se bavi.
7. Maksimalni broj autora po jednom radu je 5.

Redakcija časopisa Info M