УДК 621.34 DOI: 10.18664/ikszt.v0i4.53952

БАЛЕНКО О.І., к.т.н., доцент,
СЕМЕНОВ С.Г., д.т.н., професор,
МОЖАЄВ О.О., д.т.н., професор, (Національний технічний університет "Харківський політехнічний інститут")

Дослідження можливостей графічних процесорів при реалізації алгоритмів симетричного шифрування

Проведено аналіз інформаційних технологій графічних процесорів. Визначено можливість їх використання як додаткових обчислювальних ресурсів для рішення деяких завдань, що потребують значних обчислювальних ресурсів (в тому числі криптографічних завдань). Реалізовано алгоритм блочного шифрування AES з використанням OpenCL. Досліджено залежності швидкості шифрування від обсягу даних.
Ключові слова: *графічний процесор, криптографія, симетричне шифрування, технологія GPU.*

Постановка проблеми у загальному вигляді та її зв'язок із важливими науковими та практичними завданнями

Практично будь-яка сучасна платформа виконання програмного коду, будь то повноцінна операційна система або віртуальна машина (наприклад, JAVA машина або .NET framework), підтримуюча мультизадачність, містить набір API, призначений для управління потоками і створення паралельних програм. Таким чином, є можливість організувати паралельні обчислення практично на будь-якій мові від Assembler до скриптових мов типу Perl. В той же час, проектувати паралельні програми не завжди виправдане рішення з точки зору часових витрат і якості коду, так як на розробника часто лягає безліч специфічних рутинних завдань по створенню, управлінню, контролю та забезпеченню синхронізації потоків виконання. Мова йде, безумовно, про рішення обчислювально-важких завдань, так як прикладні програми створюються головним чином, використовуючи API платформи.

Для рішення цих завдань з'явилися багатоядерні процесори і багатопроцесорні системи, в яких таких компонентів було декілька. Це дозволило машинам виконувати декілька завдань одночасно, а загальна (теоретична) продуктивність системи піднялася рівно у стільки разів, скільки ядер було встановлено в машині. Однак виявилося, що виробляти і конструювати багатоядерні процесори занадто складно і дорого. У кожному ядрі доводилося розміщувати повноцінний процесор складною і заплутаною x86-архітектури, зі своїм (досить об'ємним) кешем, конвеєром інструкцій, блоками SSE, безліччю блоків, що виконують оптимізації і т.і. Тому процес нарощування кількості ядер суттєво загальмувався, і було знайдено спосіб задіяти для розрахунків інші обчислювальні потужності, яких було в достатку на відеокарті.

Аналіз останніх досліджень і публікацій

Аналіз літератури [1 - 6] показав, що графічні процесори, позбавлені багатьох недоліків центрального процесора. У результаті з'явилася технологія nVidia CUDA, що визначає інтерфейс, за допомогою якого стало можливим перенести обчислення складних алгоритмів на GPU.

Формулювання цілей статті (постановка завдання)

У зв'язку з цим існує актуальне наукове завдання дослідження можливості використання відеокарт для спільних обчислень і оцінити їх ефективність.

Виклад основного матеріалу дослідження

1. Аналіз інформаційних технологій графічних процесорів.

Технологія CUDA від Nvidia. CUDA (скор. від англ. Compute Unified Device Architecture, дослівно - уніфікована обчислювальна архітектура пристроїв) – архітектура (сукупність програмних і апаратних засобів), що дозволяє виробляти на GPU обчислення загального призначення, при цьому GPU фактично виступає в ролі потужного сопроцесора.

Технологія NVIDIA CUDA ™ – це єдина середовище розробки на мові програмування C, яка дозволяє розробникам створювати програмне забезпечення для вирішення складних обчислювальних завдань за менший час, завдяки многоядерной обчислювальної потужності графічних процесорів.

CUDA дає розробнику можливість на свій розсуд організовувати доступ до набору інструкцій графічного прискорювача і управляти його пам'яттю, організовувати на ньому складні паралельні

обчислення. Графічний прискорювач з підтримкою CUDA стає потужною програмованої відкритою архітектурою, подібно сьогоднішнім центральним процесорам. Все це надає в розпорядження розробника низькорівневий, що розподіляється і високошвидкісний доступ до устаткування, роблячи CUDA необхідною основою при побудові серйозних високорівневих інструментів, таких як компілятори, отладчики, математичні бібліотеки, програмні платформи.

В архітектурі CUDA використовується модель пам'яті грід, кластерне моделювання потоків і SIMD-інструкції. Застосовна не тільки для високопродуктивних графічних обчислень, але і для різних наукових обчислень з використанням відеокарт nVidia. Учені і дослідники широко використовують CUDA в різних областях, включаючи астрофізику, обчислювальну біологію та хімію, моделювання динаміки рідин, електромагнітних взаємодій, комп'ютерну томографію, сейсмічний аналіз і багато іншого. У CUDA є можливість підключення до додатків, що використовують OpenGL і Direct3D. CUDA - багатоплатформність для таких операційних систем як Linux, Mac OS X і Windows.

Відкритий стандарт OpenCL. OpenCL (від англ. Open Computing Language - відкритий мову обчислень) - фреймворк для написання комп'ютерних програм, пов'язаних з паралельними обчисленнями на різних графічних (англ. GPU) і центральних процесорах (англ. CPU), а також FPGA. Во фреймворк OpenCL входять мова програмування, який базується на стандарті C99, і прикладний програмний інтерфейс (англ. API). OpenCL є повністю відкритим стандартом, його використання не обкладається ліцензійними відрахуваннями.

Мета OpenCL полягає в тому, щоб доповнити OpenGL і OpenAL, які є відкритими галузевими стандартами для тривимірної комп'ютерної графіки і звуку, користуючись можливостями GPU. OpenCL розробляється і підтримується некомерційним консорціумом Khronos Group, до якого входять багато великих компаній, включаючи Apple, AMD, Intel, Nvidia, ARM, Sun Microsystems, Sony Computer Entertainment та інші.

Стандарт OpenCL забезпечує паралелізм і на рівні інструкцій, і на рівні даних і є розвитком технології GPGPU, яка використовує потокові процесори для неграфічних даних. Додаток OpenCL зі стоїть з хостпрограмми і набору ядер (kernels).

У базову структуру OpenCL входить мова програмування C / C ++ (стандарт ISO C99). Модель платформи OpenCL має високорівневе опис гетерогенної системи. Центральним елементом моделі є хост - первинне пристрій, що управляє OpenCL обчисленнями та здійснює взаємодію з користувачем. Пристрої OpenCL логічно поділені на обчислювальні вузли, що складаються з обробних елементів (робочих одиниць).

Ядра написані на мові C з урахуванням паралелізму та ієрархії пам'яті. Ядро створюється в хост програмою і потім за допомогою спеціальної команди ставиться в чергу на виконання в одному з OpenCL пристроїв. Ядра об'єднані в групи, з кожною з яких зіставляється індивідуальний ідентифікатор. Всі ядра в межах однієї групи виконуються паралельно на обробних елементах одного обчислювального модуля пристрою OpenCL. Під час виконання команди створюється целочисленное простір індексів, кожен елемент якого носить назву глобального ідентифікатора. Простір індексів має розмірність N. Кожне ядро виконується окремо для кожного значення глобального ідентифікатора. Іншим важливим поняттям моделі обчислень є контекст, визначення якого (за допомогою виклику спеціальних функцій OpenCL API) є першим завданням додатки OpenCL. Контекст визначає середовище виконання ядер, в яку входять наступні компоненти: пристрої, самі ядра, програмні об'єкти, вихідний і виконуваний код майбутніх ядер, об'єкти пам'яті. Взаємодія між хостом і пристроєм OpenCL відбувається за допомогою команд, поміщених в командну чергу. Модель пам'яті описує набір рівнів пам'яті і забезпечує маніпулювання ними вчасно проведення обчислень. Стандарт OpenCL охоплює ієрархію рівнів пам'яті. Слід зазначити, що в ПЛІС з OpenCL, на відміну від процесорів і графічних процесорів, паралельні обчислення виконуються на різних ядрах. Причому в ПЛІС ядра можуть бути перетворені в спеціальні конвеєрні апаратні схеми, використовують концепцію многопоточного паралелізму оброблюваних джерел обчислювальної інформації. Кожен з цих джерел може бути відтворений багаторазово для забезпечення ще більшого паралелізму. На рис. 1 наведено графік залежності швидкості шифрування від обсягу даних.

2. Реалізація алгоритму блочного шифрування AES з використанням OpenCL

Advanced Encryption Standard (AES), також відомий як Rijndael) - симетричний алгоритм блочного шифрування (розмір блоку 128 біт, ключ 128/192/256 біт), прийнятий як стандарт шифрування урядом США за результатами конкурсу AES. Цей алгоритм добре проаналізований і зараз широко використовується, як це було з його попередником DES. Національний інститут стандартів і технологій США (англ. National Institute of Standards and Technology, NIST) опублікував специфікацію AES 26 листопада 2001 після п'ятирічного періоду, в ході якого були створені і оцінені 15 кандидатур. 26 травня 2002 AES був оголошений стандартом шифрування. Станом на 2009 рік AES є одним з найпоширеніших алгоритмів симетричного шифрування.

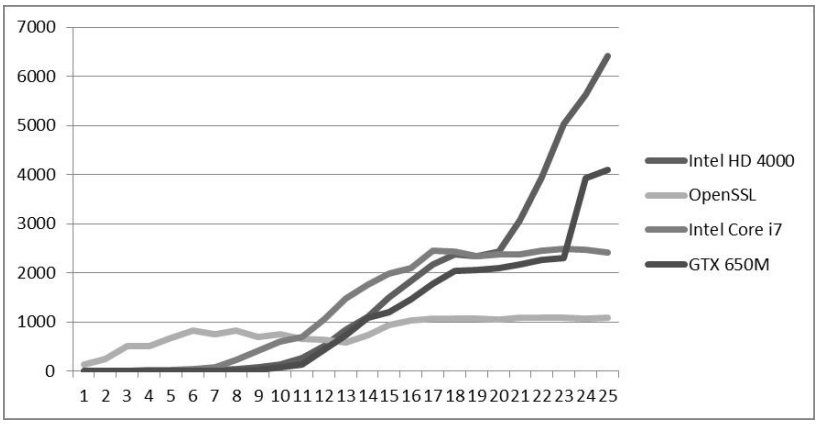

Рис. 1. Графік залежності швидкості шифрування від обсягу даних

Сучасні GPU можуть містити в собі до тисячі ядер. За рахунок розпаралелювання можна домогтися більшого приросту продуктивності. Блокові шифри є чудовим об'єктом для розпаралелювання, при шифруванні великих обсягів даних можна шифрувати по одному блоку в потоці.

В статті представлено розширення для OpenSSL в якому реалізовані функції AES шифрування з використанням OpenCL.

Програма призначена для демонстрації продуктивності алгоритму шифрування AES реалізованого з використанням OpenCL.

Програма складається з 2 основних частин - динамічного модуля OpenSSL Engine і модуля виміру продуктивності Benchmark.

Модуль OpenSSL Engine реалізує шифрування AES за допомогою технології dynamic engine бібліотеки OpenSSL і складається з двох частин - хостової (eng_opencl.h, eng_opencl.c) та коду ядра OpenCL (eng_opencl_aes.cl).

У коді ядра OpenCL функції AES_encrypt і AES_encrypt_local реалізують блоковий шифр AES.

Функції хостової частини:

load_kernel_source - завантажує код ядра OpenCL в програму;

opencl_init - проводить ініціалізацію і настройку OpenCL;

opencl_finish - проводить очищення пам'яті і завершення програми;

opencl_aes_ecb_cipher - реалізує функцію шифрування AES в OpenSSL Engine;

opencl_aes_init_key - реалізує механізм ініціалізації ключів в Engine.

Таблиця 1

Результати вимірів швидкості шифрування

Обсяг даних, Bytes	Intel Core i7	Intel HD 4000	Nvidia GTX 650M	OpenSSL
16	0,004862	128	1,267327	0,00132
32	0,353103	256	3,282051	0,273797
64	1,013861	512	8,126984	0,61244
128	2,221258	512	17,355932	1,354497
256	4,213992	682,666667	22,021505	2,503667
512	10,31738	819,2	46,545455	2,293393
1024	19,051163	744,727273	74,472727	3,785582
2048	37,236364	819,2	227,555556	7,631113
4096	78,580336	697,191489	425,558442	15,678469
8192	145,312639	762,046512	606,814815	76,920188
16384	265,866126	668,734694	689,852632	141,546436
32768	505,094412	647,269136	1044,398406	439,102178
65535	849,737439	580,606866	1481,039548	729,190542
131072	1103,764211	729,697982	1771,243243	1089,995842
262144	1493,698006	934,559715	1984,060549	1198,372571
524288	1833,976388	1030,541523	2098,201101	1455,850052
1048576	2170,403105	1073,261003	2453,526762	1782,913496
2097152	2376,039654	1063,060195	2429,719913	2040,031128
4194304	2341,224672	1064,78063	2337,473494	2060,45023
8388608	2427,434855	1045,521118	2386,092942	2104,517812
16777216	3069,236863	1082,051983	2371,296055	2167,739001

Висновки з дослідження і перспективи подальших розвідок у даному напрямку

Реалізовано алгоритм блочного шифрування AES з використанням OpenCL. На цьому прикладі показано ефективність використання технології GPU і то що незважаючи на безліч найрізноманітніших обмежень і складність розробки софта, GPU – майбутнє високопродуктивних настільних комп'ютерів. Але найголовніше – використовувати можливості цієї технології можна прямо зараз, і це стосується не тільки Windows-машин, але і Linux.

Література

1. Валиев К.А., Кокин А.А. Квантовые компьютеры: надежды и реальность. — Ижевск: РХД, 2004. — 320 с.
2. Суперкомпьютерные технологии в науке, образовании и промышленности / Под редакцией: академика В. А. Садовничего, академика Г. И. Савина, чл.-корр. РАН Вл. В. Воеводина.-М.: Издательство Московского университета, 2009. — 232 с.
3. Тим Джексон Inside Intel. История корпорации, совершившей технологическую революцию XX века = Inside Intel. The unauthorized history of the world's most successful chip company. — М.: Альпина Паблишер, 2013. — 328 с.
4. Aaftab Munshi; Benedict R. Gaster; Timothy G. Mattson; James Fung; Dan Ginsburg OpenCL Programming Guide. — Addison-Wesley Professional, 2011. — 648 p.
5. Gillam, Lee. Cloud Computing: Principles, Systems and Applications / Nick Antonopoulos, Lee Gillam. — L.: Springer, 2010. — 379 p.
6. The open standard for parallel programming of heterogeneous systems // http://www.khronos.org/opencl/

Баленко А.И., Семенов С.Г., Можаев А.А. Исследование возможностей графических процессоров при реализации алгоритмов симметричного шифрования. Проведен анализ информационных технологий графических процессоров. Определена возможность их использования в качестве дополнительных вычислительных ресурсов для решения некоторых задач, требующих значительных вычислительных ресурсов (в том числе криптографических задач). Реализован алгоритм блочного шифрования AES с использованием OpenCL. Исследованы зависимости скорости шифрования от объема данных.

Ключевые слова: графический процессор, криптография, симметричное шифрование, технология GPU.

Balenko O.I., Semenov S.G., Mozhaev O.O. The investigation of graphic engine capabiities while realizing symmetric encryption algorithm. The analysis of GPUs information technology has been conducted. The possibility of using them as additional computational resources to solve some problems that require significant computing resources (including cryptographic tasks) has been determined. Modern graphic processors can contain up to thousands of cores. One can achieve greater performance by paralleling. Block ciphers are an excellent target for parallelization - while encrypting large amounts of data you can encrypt one block in the flow. AES block encryption algorithm using OpenCL has been implemented. OpenCL is a completely open standard; its use is not subjected to royalties. The purpose of OpenCL is to complement OpenGL and OpenAL, which are open branch standards for three-dimensional computer graphics and sound, using GPU capabilities. OpenCL standard provides parallelism at the level of instructions and data level as well and is the development of GPGPU technology, which uses stream processors for non-graphical data. OpenCL appendix consists of a host programme and a set of nuclei (kernels). The dependence of speed encryption upon data volume has been studied.

Key words: graphic processing, cryptography, symmetric encryption, GPU technology.

Рецензент Порошин С.В. д.т.н., професор, завідувач кафедри (НТУ «ХПІ»)

Поступила 02.06.2015г

***Balenko O.I.**, Dr. of philosophy, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine.*
***Semenov S.G.**, Dr. of tech. science, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine.*
***Mozhaev O.O.**, Dr. of tech. science, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine.*

***Баленко О.І.**, к.т.н., доцент кафедри ОТП, Національний технічний університет "Харківський політехнічний інститут", Харків, Україна.*
***Семенов С.Г.**, д.т.н., професор кафедри ОТП, Національний технічний університет "Харківський політехнічний інститут", Харків, Україна.*
***Можаєв О.О.**, д.т.н., професор кафедри МІТС, Національний технічний університет "Харківський політехнічний інститут", Харків, Україна.*