УДК 004.75

д.т.н., доц. **Боряк К.Ф.** (ОДАТРЯ)
д.т.н., проф. **Мясіщев О.А.** (ХмНУ)
Ноянчук В.В. (ХмНУ**)**

РОЗПАРАЛЕЛЮВАННЯ ЗАДАЧ РОЗВ'ЯЗУВАНИХ МЕТОДОМ СКІНЧЕННИХ ЕЛЕМЕНТІВ НА GPU NVIDIA

У статті проведено аналіз швидкості та продуктивності обчислень добутку двох розріджених матриць в послідовному варіанті та за допомогою технологій CUDA і OpenMP. До даного типу матриць зводяться задачі, що розраховуються методом скінченних елементів. Описано методи зберігання даного типу матриць та наведено алгоритм за яким виконуються розрахунки. Отримані результати дають можливість провести узагальнену оцінку покращення швидкості обчислень при використанні графічних процесорів.

Програмно-апаратна архітектура CUDA від компанії NVIDIA для паралельних розрахунків на графічних процесорах, добре підходить для вирішення завдань, що потребують високопродуктивних паралельних обчислень для чисел з одинарною точністю, оскільки результати дослідження показали, що використовуючи, навіть, не досить потужний графічний процесор, можна отримати систему більш продуктивну, ніж при використанні декількох процесорів.

Ключеві слова: матриці, обчислення, МСЕ, CUDA, продуктивність, OpenMP.

Вступ. На сьогоднішній день існує багато задач, що потребують високої продуктивності обчислень: аналіз передачі тепла між матеріалами, деформація об'єктів при будівництві літаків та автомобілів, рух рідин по трубах. Послідовні методи не можуть забезпечити необхідної швидкості розрахунків, саме тому створюється безліч паралельних алгоритмів і систем таких як: кластери, багатопроцесорні системи та інші, що дозволяють виконати розрахунки певних задач на порядок швидше.

Задачі пружності, теплопередачі часто зводяться до вирішення систем диференційних рівнянь, використовується метод скінченних елементів. Завдяки цьому для вирішення задачі необхідно перемножити матриці, обрахувати системи лінійних рівнянь з багатьма невідомими.

Операції між матрицями добре розпаралелюються, що дає змогу скоротити час обрахунку у декілька разів.

Постановка проблеми. Матриці досить широко використовуються при математичному моделюванні різноманітних процесів та явищ. Але такі обчислення досить трудомісткі і потребують використання потужних обчислювальних систем. Проте вартість багатьох з таких систем обчислення та їх модернізація досить висока, саме тому потрібно використовувати варіанти, що потребують затрати меншої кількості коштів при установці і модернізації та гарантують максимальну завантаженість процесорів та процесорних ядер.

Виклад основного матеріалу досліджень. В ході дослідження було проведено порівняння часу виконання програми перемноження двох прямокутних розріджених матриць з використанням технологій OpenMP для розпаралелення задач по процесорах та технології CUDA від фірми NVIDIA, що дозволяє виконувати паралельні розрахунки на графічному процесорі. Також виконання програм виконувалось без використання розпаралелювання, в один потік. В якості обчислювальної системи використовувався комп'ютер з такими характеристиками:

- Процесор: Intel Core i3-2310m 2.10Ghz з 2 фізичними ядрами
- Оперативна пам'ять: 6Гб
- Графічний процесор: NVIDIA GeForce GT 630M
- Кількість ядер CUDA: 96
- Частота графічної підсистеми: 660МГц, тактова частота процесора 1320МГц.
- Розділяєма системна пам'ять: 2047МБ

- Інтерфейс пам’яті: 128біт

Програмне забезпечення, що використовується:

- MS Visual Studio 2012
- Бібліотека OpenMP.

Також використовувалися числа з одинарною точністю.

В ході обрахунків використовуються розріджені матриці, де ненулеві елементи знаходяться біля головної діагоналі. Зберігання в пам’яті такої матриці досить затратно, тому для зберігання таких матриць та пришвидшення обчислень використано два з багатьох форматів, що відкидають усі нулеві елементи, зберігаються тільки дані про значущі елементи та їх розміщення у матриці, а саме CSR та CSC.

Матриці в форматі CSR (Compressed Sparse Row), являють собою три масиви даних де:

- Масив Val – масив усіх ненулевих елементів матриці.
- Масив RovPtr - масив з номерами що вказують на новий рядок з ненулевими елементами.
- Масив ColIndex – масив з індексами ненулевих значень в рядках.

Приклад матриці в форматі CSR відображено на рисунку 1.

$$\begin{bmatrix} 1.0 & 1.3 & 0.0 & 0.0 \\ 2.2 & 0.5 & 0.0 & 0.0 \\ 0.0 & 5.1 & 2.1 & 0.0 \\ 0.0 & 0.0 & 0.1 & 3.0 \end{bmatrix}$$

Val	=	1.0	1.3	2.2	0.5	5.1	2.1	0.1	3.0
RowPtr	=	0	2	4	6	8			
ColIndex	=	0	1	0	1	1	2	2	3

Рис. 1. Формат матриці CSR

Ще одним варіантом зберігання являється CSC (Compressed Sparse Column). Він аналогічний попередньому, але є певні відмінності:

- Значення ненулевих елементів записуються у порядку розміщення їх у стовпцях матриці.
- Масив ColPtr – масив, що зберігає вказівник на новий стовпець матриці.
- Масив RowIndex – масив індексів ненулевих значень у стовпцях.

Приклад матриці у даному форматі відображено на рисунку 2.

$$\begin{bmatrix} 1.0 & 1.3 & 0.0 & 0.0 \\ 2.2 & 0.5 & 0.0 & 0.0 \\ 0.0 & 5.1 & 2.1 & 0.0 \\ 0.0 & 0.0 & 0.1 & 3.0 \end{bmatrix}$$

Val	=	1.0	2.2	1.3	0.5	5.1	2.1	0.1	3.0
ColPtr	=	0	2	5	7	8			
RowIndex	=	0	1	0	1	2	2	3	3

Рис. 2. Формат матриці CSC

Обчислюючи добуток двох матриць A * B = C з використанням даних форматів, матриці зберігаються у наступних форматах: матриці A та B у форматі CSR, матриця C у форматі CSC.

Для множення матриць використовується алгоритм відображений на рисунку 3, де змінні RowPtrArr – масив покажчиків на рядки з ненулевими елементами матриці A,

ColPtrArr – масив покажчиків на стовпці матриці B, CSRCollIndArr/CSCRowIndArr – масиви з індексами ненулевих елементів в матриці B/A, CSRValArr, CSCValArr – масиви зі значеннями матриць.

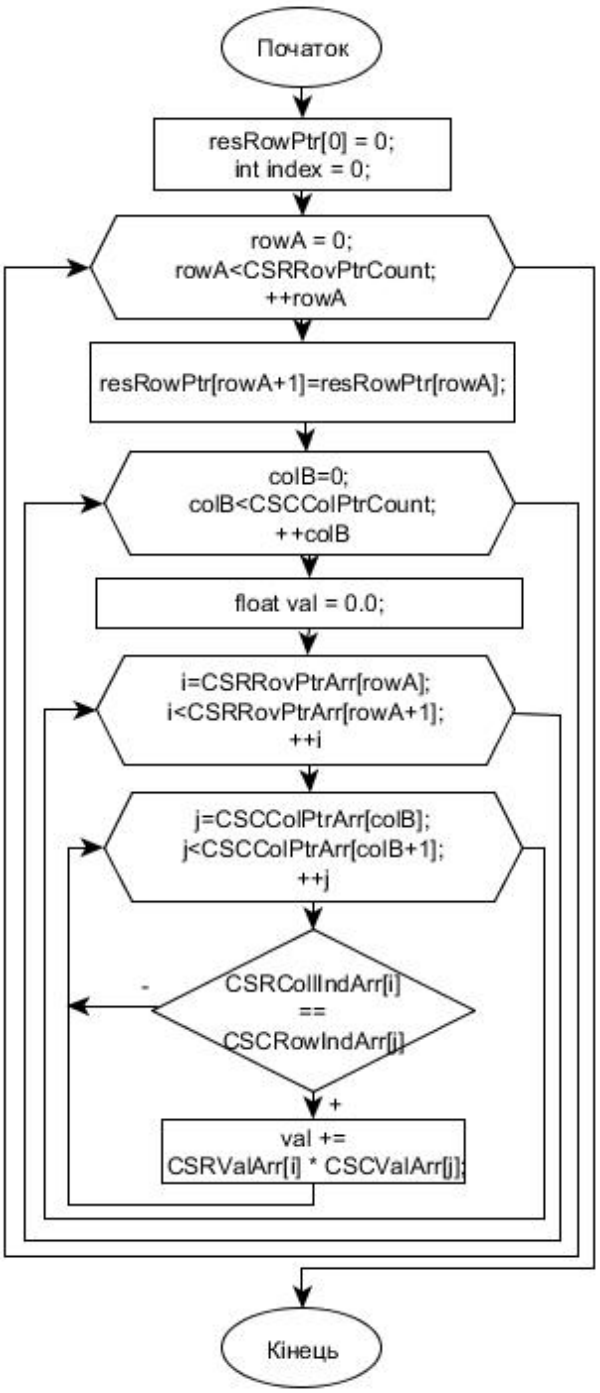

Рис. 3. Алгоритм обчислення

При використанні бібліотеки OpenMP необхідно вказати директиви, які будуть розпаралелювати код та змінні, що будуть використовуватись спільно або індивідуально кожним потоком.

Для ініціалізації паралельного блоку було використану директиву:

#pragma omp parallel private(rowA, j, i, colB) shared(index)

Тут вказується, що усі потоки будуть спільно використовувати змінну index, змінні rowA, j, i, colB будуть у кожного з них свої. Розділено на потоки було перший цикл за допомогою директиви:

#pragma omp for

Дана директива вказує, що даний цикл необхідно виконувати паралельно у декілька потоків.

Також був розроблений варіант програми для розпаралелення обрахунку на графічному процесорі.

Результати дослідження відображено в таблиці 1.

Таблиця 1

Результати обчислення добутку двох матриць

Розміри матриць	К-сть не нулевих елементі в матриці	Intel Core i3-2310m 2.10Ghz 1 потік	Intel Core i3-2310m 2.10Ghz OpenMP 2 потоки	GeForce GT 630M
		сек/ MFlops	сек/MFlops	сек/ MFlops
300x300	49800	3,004/18,126	1,376/39,572	0,433/124,503
500x500	89600	9,388/26,7631	4,193/59,922	0,449/556,236
1000x1000	189100	41,649/48,140	18,046/111,105	0,47/4255,19
3000x3000	587100	387.462/139,485	174,778/309,221	1,528/35334,4

Висновки. Як видно з результатів(таблиця 1), час що було витрачено на обрахунок з використання паралельного обчислення значно менший від того, що виконується послідовно. Затрати на виконання завдання з перемноження двох матриць розміром 3000x3000 в послідовному (однопоточному) варіанті уже займає доволі тривалий час. Використовуючи технологію OpenMP ситуація покращується в 2 рази в порівнянні з однопоточним методом. При розрахунку на графічному процесорі час обчислення триває значно менше, швидкість обчислень в 114 раз швидше ніж при обчисленні з використанням OpenMP.

Програмно-апаратна архітектура CUDA від компанії NVIDIA для паралельних розрахунків на графічних процесорах, добре підходить для вирішення завдань, що потребують високопродуктивних паралельних обчислень для чисел з одинарною точністю, оскільки результати дослідження показали, що використовуючи, навіть, не досить потужний графічний процесор, можна отримати систему більш продуктивну, ніж при використанні декількох процесорів.

Це досягається за рахунок великої кількості процесорів у графічній карті та обміну даними між її блоками, та розбиттям складних задач на простіші.

ЛІТЕРАТУРА:

1. OpenMP Application Program Interface, 2013, http://www.openmp.org/mp-documents/OpenMP4.0.0.pdf.

2. Abhishek Deshpande./ Sparse Matrix Multiplication using CUDA and Mex Interface/ Florida State University College of Arts and Science, https://www.cs.fsu.edu/research/projects/aseshpande.pdf.

3. NVIDIA CUDA — неграфические вычисления на графических процессорах, http://www.ixbt.com/video3/cuda-1.shtml.

4. Sparse Matrix, http://en.wikipedia.org/wiki/Sparse_matrix.

Рецензент: д.т.н., проф. Лєнков С.В., начальник науково-дослідного центру Військового інституту Київського національного університету імені Тараса Шевченка

д.т.н., доц. Боряк К.Ф., д.т.н., проф. Мясищев А.А., Ноянчук В.В.

РАСПАРАЛЛЕЛИВАНИЕ ЗАДАЧ РЕШАЕМЫХ МЕТОДОМ КОНЕЧНЫХ ЭЛЕМЕНТОВ НА GPU NVIDIA

В статье проведен анализ скорости и производительности вычислений произведения двух разреженных матриц в последовательном варианте, с помощью технологий CUDA и OpenMP. К этому типу матриц сводятся задачи, которые рассчитываются методом конечных элементов. Описаны методы хранения данного типа матриц и приведен алгоритм по которому выполняются расчеты. Полученные результаты дают возможность провести обобщенную оценку улучшения скорости вычислений при использовании графических процессоров.

Программно-аппаратная архитектура CUDA от компании NVIDIA для параллельных расчетов на графических процессорах, хорошо подходит для решения задач, требующих высокопроизводительных параллельных вычислений для чисел с одинарной точностью, поскольку результаты исследования показали, что используя даже недостаточно мощный графический процессор, можно получить систему более производительную, чем при использовании нескольких процессоров.

Ключевые слова: матрицы, вычисление, МКЭ, CUDA, производительность, OpenMP.

Prof. Borjak K.F., Prof. Myasischev A.A., Ph.D. Noyanchuk V.V.

PARALLELIZATION PROBLEMS SOLVED BY FINITE ELEMENT METHOD ON GPU NVIDIA

The paper analyzes the speed and efficiency of computing the product of two sparse matrices in a sequential version and using technology CUDA and OpenMP. Such matrix types are used to solve problems by finite element method. Introduced methods for storing of such matrix types and calculation algorithm. Received results allow estimation of calculating speed with usage of GPU.

Software and hardware architecture NVIDIA CUDA from the company for parallel calculations on the graphics processor is well suited for solving problems that require high-performance parallel computing for numbers with single precision, since the results showed that using even not powerful enough graphics processor available system more productive than using multiple processors.

Keywords: matrix calculation, ITU, CUDA, performance, OpenMP.