УДК 681.322

Є.О. Шквар

Національний авіаційний університет, Київ

ЕФЕКТИВНІСТЬ ПАРАЛЕЛЬНОГО РОЗВ'ЯЗАННЯ РІВНЯННЯ ПУАССОНА НА КЛАСТЕРНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМАХ З ГРАФІЧНИМИ ПРИСКОРЮВАЧАМИ

Розглянуто шляхи масштабування ефективності обчислень при чисельному розв'язанні типового елементу постановок багатьох задач математичної фізики, який одночасно є одним з найвимогливіших до ресурсів обчислювальної техніки – рівняння Пуассона. Особливу увагу приділено дослідженню ефективності спільного використання CUDA та MPI технологій паралелізації та розгалуження алгоритмів. Показано, що ефективність використання MPI при розв'язанні рівняння Пуассона на обчислювальних кластерах з графічними прискорювачами компанії NVIDIA суттєво покращується при збільшенні розмірності задачі і для просторового розрахункового випадку збільшення кількості задіяних графічних прискорювачів дозволяє забезпечити близьке до лінійного масштабування прискорення обчислень.

Ключові слова: *паралельні розподілені обчислення, CUDA, MPI, рівняння Пуассона*

Вступ

Сучасні тенденції розвитку методології аерогідродинамічного моделювання характеризуються стрімким зростанням обсягів впровадження та неухильним здешевленням застосування числових методів у порівнянні з експериментальним та наближено-аналітичним визначенням характеристик обтікання. Типовою рисою розвитку сучасних обчислювальних технологій, орієнтованих на вимогливі до комп'ютерних ресурсів задачі, є інтенсивні розвиток і впровадження ефективних методів паралелізації обчислень та використання відповідного комп'ютерного устаткування, яке дозволяє ці методи втілити. Одним із перспективних напрямків, що бурхливо розвивається у теперішній час, є застосування для ресурсовитратних обчислень прискорювачів з масивно-паралельним принципом побудови процесорів та, відповідно, SIMD (Single Instruction – Multiple Data, одна інструкція – багато даних) та SIMT (Single Instruction – Multiple Threads, одна інструкція – багато потоків) інструкціями управління обробкою даних. Різні реалізації прискорювачів обчислень характеризуються суттєвими відмінностями в архітектурі, що обумовлює необхідність кропіткої адаптації та оптимізації передбачених до використання паралельних алгоритмів під особливості наявних обчислювальних ресурсів. Саме ця обставина забезпечує значні переваги новій розробці компанії Intel – прискорювачу Xeon Phi, який на новій технологічній базі наслідує стандартну архітектуру процесорів Intel x86, розвиваючи та вдосконалюючи її завдяки масштабуванню кількості обчислювальних ядер (Many Integrated Cores – MIC – багато інтегрованих ядер). Переваги такого підходу безсумнівні, оскільки прискорювач Xeon Phi забезпечує досить швидке портування під нього паралельного програмного коду, написаного для центральних процесорів (CPU – Central Processing Unit) з використанням вже добре відлагоджених стандартів багатопоточного програмування, зокрема, OpenMP. З іншого боку, незважаючи на легкість адаптації існуючих програм під архітектуру Xeon Phi, для досягнення максимальної ефективності ці алгоритми все ж потребують відповідної оптимізації, зокрема і у першу чергу щодо узгодження роботи обчислювальних ядер з оперативною пам'яттю (RAM – Random Access Memory) прискорювача. До того ж, на сьогодні пікова продуктивність Xeon Phi суттєво (більш ніж вдвічі) поступається прискорювачам на базі графічних процесорів (GPU – Graphic Processing Unit) з підтримкою архітектури CUDA (Compute Unified Device Architecture), запропонованого компанією NVIDIA в 2006 р., чи стандарту паралельного програмування OpenCL, розробленого консорціумом Khronos Group в 2008 р., які інтенсивно розвиваються по теперішній час провідними компаніями-розробниками GPU – NVIDIA та AMD. Так, при виконанні операцій з одинарною точністю найпродуктивніше рішення від Intel (Xeon Phi 7120P/X) теоретично спроможне забезпечити до 2,416 Teraflops [1] проти 5,04 Teraflops у випадку застосування найсучаснішого та, відповідно, найпотужнішого графічного прискорювача GTX 780 Ti від NVIDIA [2]. Лідер швидкодії від компанії AMD – відеокарта Radeon HD 7970 забезпечує 3,79 Teraflops [3]. Безумовно, у порівнянні з найпродуктивнішим сучасним 6-ти ядерним (12-ти поточним) CPU Intel I7-3960x, який на максимальній частоті ядер 3,3 GHz при їх одночасному використанні забезпечує 185 Gflops, усі вищезазначені результати виглядають вкрай обнадійливо (прискорення досягає 13,06 разів – для Intel Xeon Phi 7120P/X; 27,24 разів – для NVIDIA GTX 780 Ti та 20,49 разів – для Radeon HD 7970). Керуючись ба-

жанням забезпечення в прикладних розрахунках якомога більшої обчислювальної продуктивності, що краще дозволяє забезпечити CUDA у порівнянні з OpenCL [4] при суттєво меншій вартості необхідного устаткування (а навіть найдешевші продукти NVIDIA серії GeForce переважно ігрового спрямування забезпечують можливість ефективного використання обчислювальної технології CUDA на рівні, який у цілому не поступається розробкам тієї ж компанії серії Tesla для професійного використання) дане дослідження присвячено вивченню можливостей масштабування обчислювальної швидкодії саме графічних CUDA акселераторів. Особливу увагу у даному дослідженні приділимо дослідженню масштабування продуктивності GPU шляхом їх використання в складі вузлів розподіленої обчислювальної системи – кластера. Типовим для обчислювальних систем кластерної архітектури є необхідність розгалуження задачі на кілька процесів, що виконуються на окремих вузлах кластера, отже у даному випадку виникає необхідність разом з використанням CUDA застосовувати також засіб обміну проміжними даними між вузлами впродовж обчислень, для чого застосовуватиметься інтерфейс передачі повідомлень (Message Passing Interface - MPI). Останній через високу латентність та порівняльно низьку пропускну здатність обладнання, що забезпечує міжвузловий обмін, неухильно зменшує завантаження GPU, особливо при зростанні кількості задіяних вузлів, тому є актуальним дослідити вплив сумісного застосування CUDA та MPI на ефективність гетерогенних обчислень, орієнтованих переважно на GPU.

Загальна постановка та обґрунтування актуальності проблеми. При існуючому різноманітті формулювань аерогідродинамічних задач, обумовленому як різними геометричними конфігураціями, так і широким діапазоном варіювання режимних параметрів, одним із базових рівнянь значної кількості сучасних розрахункових методів, зокрема для визначення полів тиску та потенціалу швидкості, є рівняння Пуассона. Це – еліптичне диференціальне рівняння у частинних похідних другого порядку, що має наступний вигляд:

$$\Delta\Phi = \nabla^2\Phi = \frac{\partial^2\Phi}{\partial x^2} + \frac{\partial^2\Phi}{\partial y^2} + \frac{\partial^2\Phi}{\partial z^2} = S_\Phi , \qquad (1)$$

де $\Phi = \Phi(x, y, z)$ – шукана функція; x, y, z – декартові координати; S_Φ – джерельний член, який є відомою функцією координат в розрахунковій області W і задається, як правило, у вигляді залежності від інших розрахункових змінних задачі.

Варто зазначити, що крім аерогідродинаміки це рівняння традиційно використовується і при розв'язанні задач теплопровідності, дифузії, магніто-електростатики, при обробці зображень тощо. Обумовлена нелінійністю вихідних рівнянь аерогідродинаміки ітераційна структура алгоритмів чисельних методів потребує багаторазового розв'язання рівняння Пуассона з метою поступового уточнення розрахункових полів, а неухильно зростаючі вимоги до роздільної здатності проведення обчислень вимагають також дискретизації розрахункової області дрібними сітками, що разом з еліптичним типом самого цього рівняння різко збільшує ресурсовитратність процедури відшукання розв'язку. До того ж, сучасні та перспективні у майбутньому методи моделювання зсувних течій на рівні відтворення особливостей структури і динаміки турбулентних вихроутворень, такі як пряме чисельне моделювання (DNS – Direct Numerical Simulation) та моделювання великих вихорів (LES – Large Eddy Simulation) передбачають інтегрування вихідних рівнянь на просторовій сітці з великою роздільною здатністю та з дуже дрібним кроком по часовій змінній, а на кожному з цих кроків треба розв'язувати рівняння Пуассона для тиску, що потребує окрім значних витрат обчислювальних ресурсів також і суттєвих витрат часу і є найсуттєвішим обмеженням впровадженню цих методів в дослідницьку та інженерну діяльність. Як показали дослідження [5], саме розв'язання рівняння Пуассона є найвимогливішим елементом алгоритму LES щодо утилізації обчислювального часу, оскільки складало близько 45% загального часу проведення розрахунків як при використанні послідовного алгоритму, так і при багатопоточних обчисленнях на SMP (Symmetric Multiprocessing – симетрична багатопроцесорна обробка) комп'ютері з двома чотириядерними процесорами. У останньому випадку прискорення обчислень на 8-ми ядрах при розв'язанні рівняння Пуассона склало лише 2,83, що й обумовило величину остаточного прискорення розрахунків за багатопоточним алгоритмом застосованої реалізації метода LES у 2,86 разів. Зазначені чинники є вичерпним обґрунтуванням актуальності подальшого пошуку методів більш ефективної паралелізації процесу чисельного розв'язання рівняння Пуассона, ніж SMP технологія, заснована на масштабуванні обчислювальних центральних процесорів чи кількості їх ядер. Одним з перспективних шляхів є перекладання обчислень найбільш ресурсовимогливих елементів алгоритму і, зокрема, рівняння Пуассона на GPU, тобто застосування технології неграфічних обчислень CUDA з подальшим масштабуванням досягнутого приросту шляхом одночасного використання при проведенні розрахунків обчислювальних потужностей кількох GPU, встановлених в одному комп'ютері. Узгодження паралельної роботи кількох GPU на даному етапі досліджень буде здійснюватися на основі технології програмування MPI обміну інформацією між вузлами. Отже даний гетерогенний підхід об'єднує два різні типи моделей паралельних обчислень: SIMT завдяки застосуван-

ню CUDA та MIMD (Multiple Instructions – Multiple Data, багато інструкцій – багато даних).

Метою даного дослідження є визначення ефективності масштабування процесу паралельного розв'язання рівняння (1) на масиві розподілених GPU при використанні кластерних обчислювальних систем з графічними акселераторами шляхом поєднаного застосування CUDA та MPI технологій.

Основний розділ

Формалізація задачі: межові умови, різницева апроксимація, метод розрахунку та особливості його реалізації для одного GPU

Розглянемо задачу відшукання розв'язку рівняння (1) в просторовій кубічній області W. Зв'яжемо осі координат (x, y, z) з ребрами області і покриємо просторову розрахункову область W рівномірною сіткою. Вважатимемо (як розповсюджений типовий випадок), що на усіх межах розрахункової області задано умови Діріхле (або Неймана), тобто межові значення змінної Ф (або її похідної по нормалі до межі) передбачаються відомими і заданими відповідними розподілами. Джерельний член задаватимемо виразом

$$S_\Phi = \exp\left[-\left(\frac{i}{i_{max}} - 0.2\right)^2 - \right.$$
$$\left. -\left(\frac{j}{j_{max}} - 0.5\right)^2 - \left(\frac{k}{k_{max}} - 0.7\right)^2\right],$$

де i, j, k – номери вузлів скінченно-різницевої сітки, якою покривається розрахункова область W вздовж осей x, y та z відповідно, $1 \le i \le i_{max}$, $1 \le j \le j_{max}$, $1 \le k \le k_{max}$.

Для даної модельної задачі з метою спрощення подальших перетворень використаємо рівномірну сітку з кроками вздовж осей Δx, Δy, Δz відповідно. Кількість вузлів в усіх напрямках з метою спрощення на даному етапі приймемо однаковою, тобто $i_{max} = j_{max} = k_{max} = M$. Різницева апроксимація рівняння (1) на даній сітці зводить його до вигляду

$$\Phi^{n+1}_{i,j,k} = \frac{1}{2}\Big((\Phi^n_{i+1,j,k} + \Phi^n_{i-1,j,k})\Big)\Delta x^{-2} + \\ + (\Phi^n_{i,j+1,k} + \Phi^n_{i,j-1,k})\Delta y^{-2} + \\ + (\Phi^n_{i,j,k+1} + \Phi^n_{i,j,k-1})\Delta z^{-2} - S^n_\Phi\Big)\times \\ \times(\Delta x^{-2} + \Delta y^{-2} + \Delta z^{-2})^{-1}, \quad (2)$$

де n – номер ітерації. Отримане рівняння (2) дозволяє обчислювати значення змінної $\Phi_{i,j,k}$ за значеннями Ф в шести сусідніх з нею точках різницевого шаблону.

Для знаходження поля $\Phi(x, y, z)$ у вигляді масиву вузлових значень $\Phi_{i,j,k} = \Phi(x_i, y_j, z_k)$ використовуватимемо так званий Red/Black метод [6], суть якого полягає в тому, що обрахунки здійснюються у два послідовні кроки: спочатку лише для усіх парних вузлів, а потім на основі вже знайденого в усіх парних вузлах проміжного розв'язку – лише для усіх непарних (порядок виконання цих кроків допускає зміну на протилежний). Цей метод, по суті, є найефективнішою з можливих модифікацією метода Зейделя ітераційного розв'язання систем лінійних алгебраїчних рівнянь на випадок паралельних алгоритмів для обчислювальних систем, що реалізують масово-паралельну модель обчислень і до яких, зокрема, відносяться GPU. Відповідний ітераційний процес мусить тривати до тих пір, доки не буде досягнута необхідна точність, яка контролюється виконанням умови $\left|(\Phi^{n+1}_{i,j,k} - \Phi^n_{i,j,k})/(\Phi^{n+1}_{i,j,k})_{max}\right| \le \varepsilon$, де ε – параметр, що задає точність розрахунків. Ураховуючи, що метою дослідження є не сам розв'язок (2) і аналіз його властивостей, а прискорення обчислювального процесу його отримання, приймемо ще одне спрощення, яке полягатиме в тому, що буде розглядатися процес чисельного розв'язання (2) протягом наперед заданої кількості ітерацій $N_{it} = 2000$. Є добре відомим факт, що для побудови ефективної програми, що завантажує обчисленнями GPU, замало лише вибору якісного розрахункового методу, а крім того потрібно ще ретельно адаптувати його до наявних ресурсів GPU і, зокрема, забезпечити ефективну взаємодію обчислень масивом ядер з пам'яттю GPU. Заради цього в CUDA передбачено реалізацію доступу до різних видів RAM GPU. На даному етапі розробки алгоритму було використано як найповільнішу за швидкістю доступу, але й найбільшу за об'ємом глобальну пам'ять (global memory), у якій зберігалися та послідовно уточнювалися масиви шуканої змінної $\Phi_{i,j,k}$, так і один з найшвидших за часом доступу вид пам'яті спільного використання (shared memory) потоками в межах кожного з блоків, на які відповідно зі специфікацією CUDA розділяється увесь обсяг даних, що підлягають обробці на конкретному GPU. Крім того, при реалізації Red/Black методу було вжито заходів щодо максимального завантаження усіх потокових процесорів GPU шляхом мінімізації розгалужень алгоритму, а також забезпечено наскільки можливо спільне розташування елементів масивів, що вибираються та опрацьовуються кожним з CUDA потоків. Було проаналізовано розміри блоків і з'ясовано, що найкраща продуктивність обчислень досягається при виборі блоків з розмірами (у кількості потоків вздовж напрямків x, y, z): $N_x \times N_y \times N_z = 4 \times 4 \times 32$. Для збільшення швидкості збіжності до сталого розв'язку застосовувався традиційний метод SOR

(Successive Over-Relaxation – послідовної верхньої релаксації) зі зростаючим по мірі збільшення кількості ітерацій коефіцієнтом релаксації від $r = 0,85$ для перших 50-ти ітерацій послідовно до $r = 1,6$ після досягнення $N_{it} = 1500$. Остаточний розв'язок кожного з двох кроків Red/Black методу відшукується на основі як розв'язку (2) для поточній ітерації $\Phi^{n+1}_{i,j,k}$, так і відповідного значення з попередньої ітерації $\Phi^{n}_{i,j,k}$, а саме $\Phi_{i,j,k} = r\Phi^{n+1}_{i,j,k} + (1-r)\Phi^{n}_{i,j,k}$. Відповідну програму для обчислень на GPU було написано мовою C, яка є базовою мовою CUDA. Особливості реалізації розрахункового метода для кластерної обчислювальної системи, вузли якої обладнані GPU, висвітлює наступний підрозділ.

Відмінності декомпозиції розрахункової області та реалізацій розрахункового метода для обчислювальних систем з багатоядерними CPU та GPU кластерів

Для здійснення обчислень на кластерній системі з кількома GPU розрахункова область W ділилася в напрямку координати x на підобласті по загальній кількості задіяних при проведенні обчислень GPU. Це обумовлено тим, що згідно зі стандартом мови C дані в багатовимірних масивах зберігаються рядками вздовж напрямку, який ототожнюється з останнім індексом (z координата та індекс k). Розрахунки проводилися на кластері СКІТ-4 Інституту кібернетики НАНУ, кожен гібридний вузол якого обладнаний трьома GPU NVidia Tesla M2075, двома восьмиядерними процесорами Intel Xeon E5-2600 (2,6 GHz) та 64 GB RAM. Міжвузловий інтерфейс кластера реалізовано на основі Infiniband FDR 56 Gbit/s. У даному дослідженні використовувалися до 4 вузлів кластера, тобто до 12 GPU. Кожному з GPU програмно (на основі технології MPI) ставився у відповідність процес CPU, який керував обчисленнями для відповідної підобласті на кожній ітерації.

З метою порівняння обчислювальної продуктивності при виконанні розрахунків на кількох GPU та CPU було також створено ще одну програму, яка реалізує на основі OpenMP той самий Red/Black метод з SOR процедурою, але для проведення відповідних обчислень на CPU. Декомпозиція розрахункової області W у цьому випадку здійснювалася відповідно до кількості задіяних для проведення обчислень CPU ядер.

Найсуттєвіша відмінність декомпозиції W для GPU та CPU обчислень полягає в тому, що в першому випадку кожен GPU працює з власною відеопам'яттю, отже кожна підобласть області W, що копіюється з RAM комп'ютера в RAM GPU, мусить на межах мати додаткові зони, в яких міститься інформація з суміжних підобластей. Тобто декомпозиція області W при паралельних обчисленнях на кількох GPU передбачає обов'язкове перекриття підобластей, що опрацьовуються кожним з GPU, а також синхронізацію даних впродовж обчислень в зонах перекриття шляхом організації відповідних пересилань, які в даній реалізації алгоритму здійснювалися через MPI інтерфейс незалежно від того, чи належать задіяні GPU одному вузлу кластера або кільком вузлам. Безумовно, це не тільки ускладнює алгоритм та програмну реалізацію, а й потребує додаткових комп'ютерних ресурсів, тому може бути однією з найсуттєвіших причин уповільнення обчислень. З іншого боку, безсумнівною перевагою реалізації MPI обміну даними між GPU є можливість масштабування ефективності обчислень шляхом їх виконання на розподіленій обчислювальній системі кластерної архітектури з GPU, розташованими на різних вузлах.

При виконанні обчислень на комп'ютері з одним чи кількома багатоядерними CPU кожному потоку, який опрацьовується відповідним ядром, уся RAM комп'ютера є доступною, отже необхідність в перекритті підобластей з необхідністю реалізації відповідного обміну даними для них є відсутньою. Це значно спрощує алгоритм, дозволяючи реалізувати його шляхом застосування технології багатопоточного програмування, наприклад, OpenMP, яка й була використана в даному дослідженні. Так, на вищезазначеному кластері Інституту кібернетики НАНУ кожен вузол забезпечує можливість програмування паралельної роботи до 16 ядер.

Порівняльному аналізу прискорення обчислень на кластері з GPU шляхом збільшення кількості останніх з результатами багатопоточних обчислень на CPU при масштабуванні кількості задіяних ядер присвячено наступний підрозділ.

Порівняльний аналіз результатів масштабування ефективності обчислень на комп'ютері з двома багатоядерними CPU та кластерній системі з кількома GPU

Проведено кілька серій розрахунків на CPU та системі з кількома GPU. Значення розміру M тривимірного масива для шуканої змінної $\Phi_{i,j,k}$ в кожному з напрямків для кожної з серій розрахунків були наступними: $M = 100; 200; 300; 400; 500; 600$. Порівняння часу виконання обчислень на одному ядрі CPU I7-3960x та на одному GPU GTX 680 показало, що зростання M є сприятливим з точки зору підвищення прискорення обчислень на GPU відносно CPU (табл. 1).

Коментуючи результати варто зауважити, що вони демонструють лише загальну тенденцію для конкретних CPU та GPU орієнтованих програмних реалізацій Red/Black методу, оскільки на даному етапі усі наявні можливості оптимізації GPU версії ще далеко не вичерпані.

Таблиця 1

Залежність прискорення обчислень від розміру масиву даних M для одного ядра CPU та одного GPU

M	100	200	400
Час обрахунків на CPU (одне ядро), $N_{it} = 2000$	19,8	165,6	1345,2
Час обрахунків на GPU GTX 680, $N_{it} = 2000$	3,34	22,3	184
Прискорення	5,94	7,43	7,3

З іншого боку, не бажаючи обмежувати розгляд конкретними розмірами масиву $\Phi_{i,j,k}$ з метою досягнення максимально можливої оптимізації роботи окремого GPU з RAM, першочерговою метою даного дослідження було визначено саме встановлення загальних тенденцій масштабування ефективності обчислень одного GPU при застосуванні GPU кластера. Тому подальші результати стосовно масштабування продуктивності обчислень шляхом задіювання кількох ядер CPU чи кількох GPU наводитимуться в збезрозміреному вигляді прискорення, яке визначатиметься часткою часу обрахунку 2000 ітерацій на одному ядрі CPU (чи одному GPU) до часу, потрібного для того ж обрахунку кількома ядрами CPU (чи кількома GPU відповідно).

Результати обрахунків масштабування прискорення обчислень по мірі збільшення кількості задіяних GPU кластерної системи (рис. 1) демонструють покращення ефективності обчислень по мірі збільшення розмірності задачі.

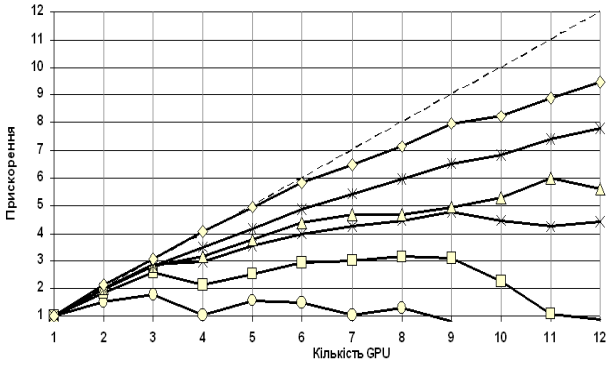

Рис. 1. Залежність прискорення обчислень від кількості задіяних GPU: ○ – M=100; □ – M=200; × – M=300; ◊ – M=400; * – M=500; Δ – M=600; штрихова лінія – теоретична межа лінійного зростання прискорення

Вже при $M \geq 300$ залежність прискорення від кількості задіяних GPU має близький до лінійного вигляд, хоч і з різним градієнтом зростання, який монотонно збільшується по мірі збільшення M. Так, при використанні для обчислень 12 GPU, розташованих на 4 вузлах кластера, максимально досягнуте прискорення дорівнювало 9,44 рази з 12-ти максимально можливих теоретично. Отже GPU кластер забезпечує збільшення ефективності масштабування швидкодії обчислень при зростанні розмірів масивів, що обробляються.

Ефект масштабування прискорення обчислень на CPU у залежності від кількості задіяних ядер (рис. 2) має протилежну динаміку.

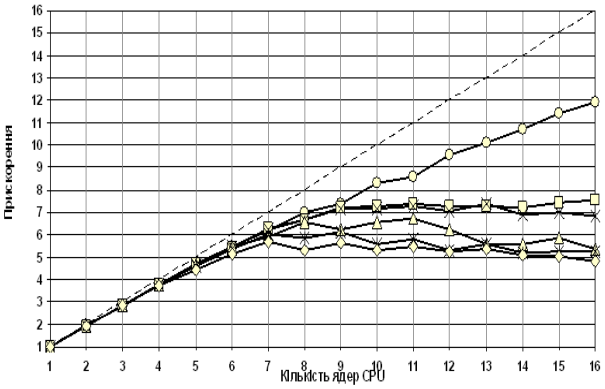

Рис. 2. Залежність прискорення обчислень від кількості задіяних ядер CPU; (позначення ідентичні використаним на рис. 1)

Так, по мірі зростання розміру M масиву $\Phi_{i,j,k}$ залежність прискорення від кількості застосованих ядер CPU (N_{CPU_cor}) втрачає лінійність вже при $M \geq 200$ та $N_{CPU\_cor} \geq 8$, швидко виходячи на стале значення, яке також має чітко виражену тенденцію до зменшення при зростанні M. Пояснення цієї обставини в неспроможності RAM одного вузла ефективно опрацьовувати паралельні запити різних ядер по мірі їх зростання понад $N_{CPU\_cor} \geq 8$. Попередні дослідження автора також довели, що при розв'язанні рівняння Пуассона на кількох багатоядерних CPU кластерної системи масштабування швидкодії дуже чутливе до латентності та пропускної здатності міжвузлового інтерфейсу, тому мінімізація обміну між процесами, а також максимальне зменшення їх кількості за рахунок гібридного використання технологій OpenMP в межах вузла та MPI для організації міжвузлового обміну забезпечують підвищення ефективності обчислень [7, 8]. До того ж, в [9] показано, що для просторової постановки задачі є доцільним використання технології віртуальної багатопоточності (Hyper-threading) сучасних CPU розробки Intel. Але в усіх цих випадках вже при кількості CPU ядер $N_{CPU\_cor} \geq 8$ починає зростати гальмівний вплив MPI інтерфейсу як на ефективність завантаження CPU вузла та ефективність технології Hyper-threading, так і загалом на сукупне масштабування швидкодії.

Таким чином, на підставі отриманих у даному дослідженні результатів та їх порівняння з [7 – 9] можна дійти висновку, що на відміну від обчислень

на CPU кластерних систем, об'єднання продуктивності кількох GPU шляхом організації обміну даними між ними на основі MPI забезпечує зростання ефективності масштабування по мірі зростання розміру масивів задачі.

Висновки

1. Проведене дослідження порівняльної ефективності паралельного розв'язання диференціального рівняння Пуассона (1) в просторовій постановці за Red/Black SOR методом на багатоядерному CPU, одному GPU та обчислювальному кластері системі з кількома GPU переконливо довело переваги останньої конфігурації при збільшенні розміру масивів задачі.

2. Технологія неграфічних обчислень на кластерних системах з GPU потребує значно більших зусиль при оптимізації алгоритму та його програмної реалізації у порівнянні з написанням відповідної багатопоточної програми для багатоядерного CPU, але вона й забезпечує суттєво ефективніше масштабування потужності обчислювальної системи шляхом збільшення кількості графічних прискорювачів.

3. Подальші зусилля автор планує докладати у напрямку підвищення обчислювальної ефективності описаної вище GPU-орієнтованої програмної реалізації Red/Black SOR метода, а також адаптації багатосіткового алгоритму відшукання розв'язку рівняння Пуассона на GPU кластерах.

Список літератури

1. Intel® Xeon Phi™ Product Family Performance. Rev 1.4 12/30/13 Електрон. ресурс. – Режим доступу: http://www.intel.com/content/dam/www/public/us/en/documents/ performance-briefs/xeon-phi-product-family-performance-brief.pdf.

2. NVIDIA GeForce GTX 780 TI 3GB. Електрон. ресурс. – Режим доступу: http://www.techpowerup.com/reviews/NVIDIA/GeForce_GTX_780_Ti/.

3. AMD Radeon™ HD 7970 Graphics Card. Електрон. ресурс. – Режим доступу: http://www.amd.com/uk/products/desktop/graphics/7000/7970/Pages/radeon-7970.aspx.

4. Karimi K. A Performance Comparison of CUDA and OpenCL / K. Karimi, N. G. Dickson, and F. Hamze // Cornell Univ. Library, May 2011. Електрон/ ресурс. – Режим доступу: http://arxiv.org/ftp/arxiv/papers/1005/1005.2581.pdf.

5. Шквар Є.О. Оцінювання масштабування обчислювальної продуктивності паралельної реалізації методу великих вихорів / Є.О. Шквар // Вісник НАУ. – 2012. – № 1. – С. 157-166.

6. Konstantinidis E. Accelerating the Red/Black SOR method using GPUs with CUDA / E. Konstabtinidis, Y. Cotronis // Proc. of 9th International Conference, Parallel Processing and Applied Mathematics 2011, Torun, Poland, September 11-14, 2011. Revised Selected Papers, Part I. – 2012. – P. 589-598.

7. Шквар Є.О. Інтегрована гібридна технологія паралельних обчислень / Є.О. Шквар // Інтегровані технології та енергозбереження: щокв. наук.-пр. ж. НТУ (ХПІ). – Х.: НТУ (ХПІ). – 2010. – № 1. – С. 86-99.

8. Шквар Є.О. Гібридний метод паралельних обчислень / Є.О. Шквар // Вісник Черкаського університету: серія Прикладна математика. Інформатика. – Вип. 172. – 2010. – С. 123–136.

9. Шквар Є.О. Ефективність використання технології віртуальної багатопоточності при паралельному розв'язанні рівняння Пуассона / Є.О. Шквар // Системи обробки інформації. – 2012. – Вип. 3(101). – С. 39-45.

Надійшла до редколегії 19.12.2013

Рецензент: д-р техн. наук, проф. С.А. Калкаманов, Харківський університет Повітряних Сил імені І. Кожедуба, Харків.

ЭФФЕКТИВНОСТЬ ПАРАЛЛЕЛЬНО РЕШЕНИЯ УРАВНЕНИЯ ПУАССОНА НА КЛАСТЕРНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ С ГРАФИЧЕСКИМИ УСКОРИТЕЛЯМИ

Е.А. Шквар

Рассмотрены пути масштабирования эффективности вычислений при численном решении типового элемента постановок многих задач математической физики, который одновременно является одним из самых требовательных к ресурсам вычислительной техники – уравнения Пуассона. Особое внимание уделено исследованию эффективности использования CUDA и MPI технологий распараллеливания алгоритмов. Показано, что эффективность совместного использования MPI при решении уравнения Пуассона на вычислительных кластерах с графическими ускорителями компании NVIDIA существенно улучшается при увеличении размерности задачи и для пространственного расчетного случае увеличение количества задействованных графических ускорителей позволяет обеспечить близкое к линейному масштабирование ускорения вычислений.

Ключевые слова: *параллельные распределенные вычисления, CUDA, MPI, уравнение Пуассона.*

EFFICIENCY OF PARALLEL SOLVING THE POISSON EQUATION ON CLUSTER COMPUTER SYSTEMS WITH GRAPHIC ACCELERATORS

Ye.O. Shkvar

The directions of computational efficiency scaling in numerical solution of typical element of wide range of different mathematical physics problems statements and also one of the most demanding to computer resources consumption – the Poisson equation have been considered. Special attention is given to studying the common efficiency of CUDA and MPI technologies of algorithms parallelization. It has been shown that the efficiency of MPI for solving the Poisson equation on computing clusters with NVIDIA graphic accelerators can be significantly improved as a result of increasing the dimension of the problem and for 3D case the nearly linear acceleration scaling of computations can be reached as a result of increasing the number of applied graphic accelerators.

Keywords: *parallel multithreaded and distributed computations, CUDA, MPI, Poisson equation.*