ЯКОВЛЕВ М.Ф., ГЕРАСИМОВА Т.О., БРУСНІКІН В.М.

РОЗВ'ЯЗУВАННЯ ЗАДАЧ З ПОЧАТКОВИМИ УМОВАМИ ДЛЯ СИСТЕМ ЗВИЧАЙНИХ ДИФЕРЕНЦІАЛЬНИХ РІВНЯНЬ НА БАГАТОЯДЕРНОМУ КОМП'ЮТЕРІ З ГРАФІЧНИМИ ПРИСКОРЮВАЧАМИ ІНПАРКОМ

При моделюванні на комп'ютері будь-яких реальних процесів і явищ за допомогою систем звичайних диференціальних рівнянь часто виникає необхідність розв'язувати задачі з початковими умовами (задачі Коші). СЗДР можуть існувати як самостійний клас задач, або можуть виникати на проміжних стадіях розв'язування більш складних математичних задач. Такі задачі виникають, наприклад, при описуванні за допомогою СЗДР рухів, процесів або явищ, що змінюються в часі. Це стосується, зокрема, процесів хімічної кінетики, процесів, що відбуваються в ядерних реакторах, а також динамічних задач аналізу міцності конструкцій при застосуванні методу скінченних елементів по просторових змінних та ін. При розв'язуванні деяких задач, наприклад, пов'язаних із рухом керованих об'єктів, виникає необхідність розв'язувати СЗДР швидше, ніж відбувається процес в реальному часі; більш того, їх розв'язування потребує багатоваріантних розрахунків та значних обчислювальних ресурсів. Дуже часто виникає необхідність розв'язувати задачі з початковими умовами у випадку, коли вихідні дані (тобто формули для обчислення правих частин СЗДР та початкові умови) задані наближено. Такі задачі можна ефективно розв'язувати на високопродуктивних комп'ютерах з паралельною організацією обчислень, зокрема, на комп'ютерах MIMD-архітектури [5].

Вимоги до високопродуктивної обчислювальної техніки випереджають можливості традиційних паралельних комп'ютерів. В даний час такі комп'ютери можуть налічувати сотні і навіть тисячі процесорів (ядер). Але збільшення числа процесорів у паралельних комп'ютерах часто приводить до значного збільшення комунікаційних втрат і зниження їх ефективності. Одним із основних напрямків підвищення продуктивності комп'ютерів є використання графічних процесорів (GPU). Цей напрямок є особливо ефективним у випадку, коли необхідно виконувати великі обсяги однорідних арифметичних операцій. Використання графічних процесорів дало можливість розробити комп'ютери гібридної архітектури, яка об'єднує багатоядерні процесори MIMD–архітектури та графічні прискорювачі SIMD–архітектури.

В статті розглядається питання організації обчислень при розв'язуванні задач з початковими умовами для СЗДР високого порядку на комп'ютерах гібридної архітектури на прикладі застосування гібридного алгоритму методу Рунге-Кутта 4-го порядку, який використовує обчислювальні можливості CPU та GPU.

2. Постановка задачі. Нехай дана система *n* звичайних диференціальних рівнянь. Задачу з початковими умовами для СЗДР *n*-го порядку на інтервалі $[t_0, T]$ розглядатимемо у вигляді:

$$\frac{dy}{dt} = f(t, y), \tag{1}$$

$$y(t_0) = y^{(0)}, \tag{2}$$

де $y = (y_1, y_2, \ldots, y_n)^T$ – шуканий вектор, а права частина системи – *n*-вимірна неперервна вектор-функція $f(t, y) = (f_1(t, y), f_2(t, y), \ldots, f_n(t, y))^T$.

При моделюванні реальних процесів на комп'ютері за допомогою СЗДР виникає ряд труднощів, зокрема, потреба мати справу із задачами з наближеними вихідними даними. Наближений характер вихідних даних може бути обумовлений такими причинами:

- похибками в вихідних даних;

- похибками в значеннях правої частини;
- застосуванням чисельного (дискретного) методу інтегрування і округленням чисел при обчисленнях.
- дискретизацією динамічних задач по просторових змінних.

Тому на практиці, як правило, замість задачі (1), (2) маємо задачу з наближеними вихідними даними:

$$\frac{dv}{dt} = f(t, v), \tag{3}$$

$$v(t_0) = v_{0,} \tag{4}$$

де $\|y_0 - v_0\| \le \delta$, $\varphi(t, w) = f(t, w) + \Delta(t, w)$, $\|\Delta(t, w)\| \le \Delta$ для довільних значень $w(t)$.

Такі задачі і розв'язуються на комп'ютерах за допомогою чисельних методів.

3. Паралельний алгоритм методу Рунге-Кутта 4-го порядку для комп'ютерів гібридної архітектури. Зупинимося на питанні організації обчислень при розв'язуванні задач з початковими умовами для СЗДР високого порядку на комп'ютерах гібридної архітектури (CPU+GPU) методом Рунге-Кутта 4-го порядку.

3.1. Декомпозиція даних. Для розв'язування СЗДР чисельними методами необхідно задавати наступну вихідну інформацію: порядок системи звичайних диференціальних рівнянь; початкову та кінцеву точки інтервалу інтегрування; визначальну точність отриманого розв'язку; похибку завдання початкових умов; похибку завдання вектор-функції; повний вектор розв'язку (оскільки в загальному випадку всі компоненти вектор-функції правих частин системи залежать від усього вектора розв'язку), на початку інтегрування цей вектор містить початкові значення розв'язку; функцію обчислення значень вектор-функції.

Реалізація методів розв'язування задач даного класу на комп'ютерах гібридної архітектури вимагає автоматичного розподілу вихідних даних та компонент векторів правих частин в локальній пам'яті ядер кожного процесора. Кожний процес одночасно і незалежно реалізує власну програму обчислень з використанням функцій CUBLAS на GPU [1]. Комунікаційні операції та синхронізація роботи процесів здійснюється засобами MPI [2].

Щоб розв'язати СЗДР n-го порядку на MIMD-комп'ютері її заздалегідь необхідно розбити на p блоків (p – кількість ядер). В паралельних алгоритмах розв'язування СЗДР на комп'ютерах гібридної архітектури використовується одновимірний блочний розподіл даних та обчислень. Блоки вибираються приблизно одного розміру в залежності від порядку СЗДР n та кількості процесів (CPU) p, що використовуються для розв'язування задачі. Тоді розмір блоку обчислюється за формулою

$$s_q = \begin{cases} \lceil n/p \rceil + 1, & q < n - p\lceil n/p \rceil \\ \lceil n/p \rceil, & q \ge n - p\lceil n/p \rceil \end{cases},$$

де $q = 0, 1, \ldots, p-1$ – логічний номер CPU, значення $\lceil a \rceil$ дорівнює цілій частині числа a. За цією схемою розподіляються компоненти векторів y та $f(t, y)$. Такий розподіл дозволяє виконувати автоматичний розподіл обчислень компонент вектор-функції СЗДР на p процесів.

3.2. Розпаралелювання обчислень вектор-функції правих частин СЗДР. При розв'язуванні задач з початковими умовами на MIMD-комп'ютерах СЗДР необхідно задавати спеціальним чином, щоб забезпечити максимально можливе розпаралелювання операцій на кожному кроці інтегрування, зокрема, при обчисленні вектор-функції правих частин, та якщо необхідно, наближення до матриці Якобі. При автоматичному розпаралелюванні обчислень на GPU слід враховувати, що лише в рідкісних випадках правила обчислення значень компонент вектор-функції СЗДР істотно відрізняються одна від одної. В цих випадках системи рівнянь, як правило не мають великого порядку і тому розв'язувати їх на комп'ютерах гібридної архітектури недоцільно. В більшості ж випадків побудова вектор-функції правих частин під-

порядковується деякому спеціальному правилу і такі системи можуть мати достатньо великий порядок.

Як правило, при розв'язуванні задач з початковими умовами для СЗДР значна частина арифметичних операцій припадає на обчислення вектор-функції правих частин системи. Тому у більшості чисельних методів в першу чергу розпаралелюється обчислення компонент вектор-функції правих частин системи рівнянь на вибраній кількості CPU багатопроцесорного комп'ютера з графічними прискорювачами [4]. Останні $s = p(q+1) - n$ процесорів оброблятимуть блоки по q рівнянь, а перші $p - s$ процесорів - блоки по (q+1)-му рівнянню; тут p – кількість процесорів, $q = 0, 1, \ldots, p-1$ – логічний номер CPU. Опишемо один із можливих способів завдання вектор-функції правих частин на мові програмування Сі. В кожному процесорі праві частини системи обчислюються за програмою:

```
diffun (n, l, m, t, y, f)
{
. . . . .
for ( i = l;  i < m;  i++)
        {
…f =...;
        }
}
```

В кожному процесорі обчислюється вектор-функція від l-ої до (m-1)-ої компоненти. При цьому $l = kq$, $m = (k+1)q$-1, де k – логічний номер процесора, $k = 0, 1, ..., p$-1. Величини l і m для кожного процесора повинні бути обчислені перед зверненням до програми обчислення вектор-функції правих частин. Цей шаблон використовується для всіх випадків завдання правих частин. Тут введені такі позначення: n – порядок системи звичайних диференціальних рівнянь; t – точка інтегрування; y – значення вектор-функції в точці t. Такий спосіб завдання вектор-функції має такі переваги:

- по-перше, полегшення при програмуванні СЗДР великого порядку, оскільки такі системи, як правило, мають регулярну структуру, і тому фактично необхідно написати заголовок циклу і рівняння, залежне від параметра циклу;
- по-друге, при прихованому паралелізмі обчислення вектор-функції правих частин автоматично розподіляється між вибраною кількістю процесорів.

Якщо ж система звичайних диференціальних рівнянь має нерегулярну структуру, то кожне рівняння доведеться позначати власною міткою типу «*case k*:» (*k*: – ціле число) і забезпечувати звернення до функції за описаним вище правилом, використовуючи в програмі оператор типу «switch (i)».

Слід зазначити, що кількість арифметичних операцій, необхідних для обчислення компонент вектор-функції правих частин СЗДР істотно впливає на ефективність і прискорення обчислень при розпаралелюванні на комп'ютері з графічними прискорювачами. Слід також зазначити, що задача може бути ефективно розв'язана, коли порядок системи n є достатньо великим числом, а також достатньо великим числом є і кількість операцій, необхідна для обчислення компонент вектор-функції $f(u)$.

3.3. Гібридний алгоритм методу Рунге-Кутта 4-го порядку. Одним із методів, що застосовується для чисельного інтегрування задач з початковими умовами для СЗДР є метод Рунге-Кутта 4-го порядку. Цей метод є найбільш розповсюдженим методом чисельного розв'язування задач з початковими умовами для СЗДР.

Класичний метод Рунге–Кутта 4-го порядку реалізується за формулами:

$$y^{(i+1)} = y^{(i)} + (k_1 + 2k_2 + 2k_3 + k_4)/6, \quad (5)$$

де

$$k_1 = h_i f(t_i, y^{(i)}),$$
$$k_2 = h_i f(t_i + h_i/2, y^{(i)} + 0{,}5k_1),$$
$$k_3 = h_i f(t_i + h_i/2, y^{(i)} + 0{,}5k_2),$$
$$k_4 = h_i f(t_i + h_i, y^{(i)} + 0{,}5k_3), (i = 0{,}1{,}2{,}...)$$

На кожному кроці інтегрування цей метод потребує чотириразового обчислення вектор-функції $f(t, y)$. Але для оцінки головного члена похибки, який визначає вибір кроку інтегрування, доводиться тричі застосовувати метод Рунге-Кутта.

Наведемо обчислювальну схему паралельного алгоритму методу Рунге-Кутта 4-го порядку для знаходження розв'язку систем звичайних диференціальних рівнянь на комп'ютерах гібридної архітектури (CPU+GPU) [3]:

1. В пам'ять кожного з *p* CPU посилається вихідна інформація.

Для подальшого викладу позначимо через *m* – кількість компонент вектор-функції, що будуть обчислюватись одним CPU.

2. На інтервалі від початкового значення незалежної змінної до координати точки виводу при кожному значенні $t_i = t_0 + \sum_{j=0}^{i-1} h_j$ (*i*= 1, 2,...) проводяться обчислення з використанням CPU та графічних процесорів в наступному порядку:

a) в CPU обчислюється прогнозована довжина кроку інтегрування h_i та посилається в GPU;

b) далі виконується інтегрування системи, кожний *i*-ий крок якого обчислюється за схемою

- в GPU обчислюються *m* компонент вектора k_1 обчислені компоненти вектора k_1 посилаються в CPU,
- в кожному CPU обчислюються *m* компонент вектора $y^{(i)} + 0{,}5k_1$; проводиться мультизбирання цього вектора та пересилка його в GPU;
- в GPU обчислюються *m* компонент вектора k_2; обчислені компоненти вектора k_2 посилаються в CPU,
- в кожному CPU обчислюються *m* компонент вектора $y^{(i)} + 0{,}5k_2$, проводиться мультизбирання цього вектора та пересилка його в GPU;
- в GPU обчислюються *m* компонент вектора k_3; обчислені компоненти вектора k_3 посилається в CPU;
- в CPU обчислюються *m* компонент вектора $y^{(i)} + k_3$, проводиться мультизбирання цього вектора та пересилка його в GPU;
- в GPU обчислюються *m* компонент вектора k_4, а потім *m* компонент вектора $y^{(i+1)} = y^{(i)} + v/6$, де $v = k_1 + 2k_2 + 2k_3 + k_4$; обчислені компоненти вектора $y^{(i+1)}$ посилаються в CPU;
- кожен CPU проводить мультизбирання всього вектора $y^{(i+1)}$ та посилає його в GPU.

Цим завершується *i*-ий крок інтегрування системи.

c) За схемою, описаною в п. «b», двократним інтегруванням з довжиною кроку $h_i/2$ обчислюється вектор $\bar{\bar{y}}^{(i+1)}$ та однократним інтегруванням з довжиною кроку h_i обчислюється вектор $\bar{y}^{(i+1)}$; ці вектори посилаються в GPU;

d) в GPU обчислюється похибка апроксимації системи $\psi^{(i+1)} = \max\limits_{1\le j\le n} \left| \bar{y}_j^{(i+1)} - \bar{\bar{y}}_j^{(i+1)} \right|$; ця похибка апроксимації посилається в CPU;

e) при виконанні умов досягнення заданої точності в CPU обчислюється уточнена довжина кроку інтегрування.

f) в CPU перевіряється умова $t_i + h_i > t_p$; якщо умова виконується, то як довжина кроку інтегрування вибирається $h_i = t_p - t_i$ де t_p – координата точки виводу розв'язку, в іншому випадку довжина кроку інтегрування не коригується, вибрана довжина кроку посилається для продовження обчислень в GPU;

g) для отримання розв'язку $y^{(i+1)}$ в точці $t_{i+1} = t_i + h_i$ повторюються пункти a) – e) з обчисленою довжиною кроку інтегрування.

Процес розв'язування задачі продовжується, доки не буде досягнуто кінцевої точки інтервалу інтегрування *T*. Після обчислення розв'язку в точці виводу запам'ятовуються вектор розв'язку, константа Ліпшиця та оцінка похибки розв'язку.

4. Програмна реалізація та чисельний експеримент. **Експериментальне дослідження ефективності гібридного алгоритму.** Розрахунки проводились на вузлі кластеру Інпарком-G [7] з наступними характеристиками:

Процесори: 2 Xeon 5606 (4 ядра з частотою 2.13 ГГц).

Графічні прискорювачі: 2 Tesla M2090 (6 Гб пам'яті).

Об'єм оперативної пам'яті: 24 Гб.

Комунікаційне середовище : InfiniBand 40 Гбіт/сек (з підтримкою GPUDirect), Gigabit Ethernet.

Також на вузлах встановлена бібліотека MKL10.2.6 та CUDA, починаючи з версії 3.2.

Розв'язувалась наступна тестова задача:

На інтервалі [0.0; 0.4] розв'язати СЗДР *n*-го порядку:

$$\frac{du_i}{dt} = -\sum_{j=0}^{n-1} u_j - u_i + n(1+t) + 2 + t$$

з початковими умовами $u_i(0) = 1$ $(i = 0,1,2,...,n-1)$.

У обчислювальному вузлі використовується чотириядерний процесор, на одне ядро приходиться одна карточка GPU. На кожному вузлі, як обчислювальні процесори, використовуються дві ігрові відеокарти.

Зупинимося на програмній реалізації алгоритму при роботі з GPU. Основними блоками операцій, що виконуються з GPU є: виділення пам'яті для змінних, копіювання даних на GPU, запуск обчислень, копіювання результатів в оперативну пам'ять, звільнення пам'яті GPU. Зауважимо, що кожний процес одночасно і незалежно реалізує власну програму обчислень з використанням функцій CUBLAS на GPU для матрично-векторних операцій [1]. Комунікаційні операції та синхронізація роботи процесів здійснюється засобами MPI [2].

Для даної тестової задачі вектор-функція правої частини може бути легко обчислена на графічних прискорювачах, оскільки система має регулярну структуру і порядок цієї системи *n* є достатньо великим числом. Якщо ж права частина дуже складна, то в описаному алгоритмі треба ігнорувати обчислення вектор-функції в GPU і залишити її обчислення в CPU.

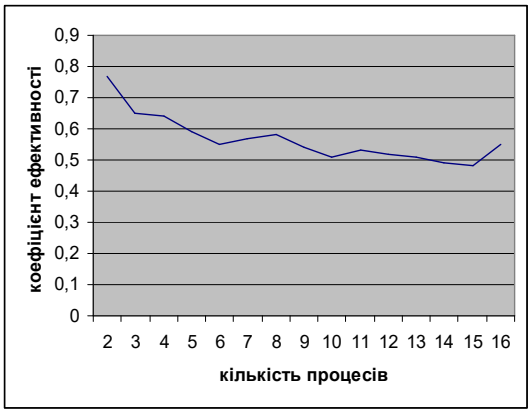
Нижче наведено таблицю, що містить часи розв'язування цієї тестової задачі при n=20000. на експериментальному зразку паралельного комп'ютера гібридної архітектури Інпарком. Результати експериментів наведені для гібридної архітектури 1 CPU, 1 GPU. У таблиці порівнюються часи розв'язування задачі при обчисленні компонент вектор-функції правої частини на GPU та на CPU. Для порівняння наведено час розв'язування задачі при використанні паралель-

ного алгоритму для середовища MPI. Зауважимо, що час розв'язування задачі на CPU становить 220 секунд.

Таблиця

	***np*=2**	***np*=4**	***np*=8**	***np*=16**
1 CPU + 1GPU Права частина обчислюється на CPU	114,96	69,48	42,46	22,44
1 CPU + 1GPU Права частина обчислюється на GPU	11,73	6,84	3,9	2,04
MPI	122,21	62,34	38,18	18,48

Нижче наведено графіки, з яких видно залежність коефіцієнта прискорення та коефіцієнта ефективності від кількості процесорів.



З графіка видно, що коефіцієнт прискорення практично лінійно збільшується зі збільшенням кількості процесів, що використовується.

Висновки. Дослідження паралельних алгоритмів розв'язування систем звичайних диференціальних рівнянь показали, що багатоядерні комп'ютери з графічними процесорами дають можливість суттєво прискорити час розв'язування задач, особливо у випадку, коли праві частини СЗДР обчислюються на GPU. У випадку, коли права частина дуже складна, і має нерегулярну структуру, то обчислення вектор-функції доведеться залишити в CPU, що суттєво погіршить час розв'язування задачі.

Запропоновано алгоритм, який забезпечує високу ефективність розпаралелювання на GPU, оптимізує використання пам'яті CPU. Показано ефективність гібридного алгоритму методу Рунге-Кутта 4-го порядку розв'язування задач початковими умовами для систем звичайних диференціальних рівнянь (СЗДР). В подальшому науковий інтерес представляють алгоритми для архітектури з n CPU, m GPU зі спільною або розподіленою пам'яттю.

Перелік використаних джерел:

[1] http://developer.download.nvidia.com/compute/cuda/1_0/CUBLAS_Library.

[2] http://www.mpi.org/.

[3] Химич А.Н., Молчанов И.Н., Попов А.В., Чистякова Т.В., Яковлев М.Ф. Параллельные алгоритмы решения задач вычислительной математики. – Киев, Наукова думка – 2008 – 248 с.

[4] Яковлев М.Ф., Герасимова Т.О., Нестеренко А.Н. Особливості розв'язування систем нелінійних та диференціальних рівнянь на паралельних комп'ютерах//Питання оптимізації обчислень (ПОО – XXXV). Праці міжнародного симпозіуму. – Київ: Інститут кібернетики ім. В.М. Глушкова НАН України, 2009. – Т.2.– С. 435-439.

[5] Молчанов И.Н. Интеллектуальные параллельные компьютеры на графических процессорах для решения научно-технических задач// Праці Міжнародної молодіжної математичної школи «Питання оптимізації обчислень ()ПОО-XXXVII) ». – К.: Інститут кібернетики імені В.М. Глушкова НАН України, 2011. – С. 121-122.