УДК 519.6

Розв'язування задач з початковими умовами для систем звичайних диференціальних рівнянь на високопродуктивних обчислювальних системах з використанням графічних прискорювачів

Герасимова Т.О. м.н.с.

Інститут кібернетики імені В.М. Глушкова НАН України, просп. Глушкова, 40, Київ, Україна

dept150@insyg.kiev.ua

Анотація. *В роботі розглядаються деякі підходи створення алгоритмів та програм розв'язування задач з початковими умовами для систем звичайних диференціальних рівнянь для комп'ютерів гібридної архітектури.*

Ключові слова

Комп'ютери гібридної архітектури, технологія програмування CUDA

При моделюванні на комп'ютері будь-яких реальних процесів і явищ за допомогою систем звичайних диференціальних рівнянь (СЗДР) часто виникає необхідність розв'язувати задачі з початковими умовами (задачі Коші). 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. 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.

Вимоги до високопродуктивної обчислювальної техніки випереджають можливості традиційних паралельних комп'ютерів. В даний час такі комп'ютер 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, 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. Розглянемо питання організації обчислень при розв'язуванні задач з початковими умовами для СЗДР високого порядку на комп'ютерах гібридної архітектури (CPU+GPU). 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].

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 правих частин системи рівнянь. По цій причині прискорення розв'язування задач з початковими умовами для СЗДР тісно пов'язане з розпаралелюванням обчислень правих частин системи на вибраній кількості CPU багатопроцесорного комп'ютера з графічними прискорювачами [4]. Кількість компонент m, що обчислюються на одному процесорі, визначається за формулою:

$$m = \begin{cases} \left[\frac{n}{p}\right] & \text{при} \quad n\text{-}p\left[\frac{n}{p}\right] = 0 \\ \left[\frac{n}{p}\right] + 1, & \text{при} \quad n\text{-}p\left[\frac{n}{p}\right] \neq 0 \end{cases},$$

тут p – кількість процесорів.

Нехай дана система n звичайних диференціальних рівнянь. Задачу з початковими умовами (задачу Коші) для СЗДР n-го порядку на інтервалі $[t_0, T]$ розглядатимемо у вигляді:

$$\begin{aligned} &\frac{dy}{dt} = f(t, y), \\ &y(t_0) = y^{(0)}, \end{aligned} \tag{1}$$

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

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

$$dv / dt = f(t, v), \tag{2}$$

$$v(t_0) = v_0, \tag{3}$$

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

Такі задачі і розв'язуються на комп'ютерах за допомогою чисельних методів. Одним із методів, що застосовується для чисельного інтегрування таких задач 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:

$$y^{(i+1)} = y^{(i)} + (k_1 + 2\,k_2 + 2\,k_3 + k_4)/6, \tag{4}$$

де

$$\begin{aligned} k_1 &= h_i\, f(t_i, y^{(i)}), \\ k_2 &= h_i\, f(t_i + h_i/2, y^{(i)} + 0{,}5\,k_1), \\ k_3 &= h_i\, f(t_i + h_i/2, y^{(i)} + 0{,}5\,k_2), \\ k_4 &= h_i\, f(t_i + h_i, y^{(i)} + k_3) \quad (i = 0, 1, 2, \ldots). \end{aligned}$$

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: порядок системи, кількість точок виводу результатів, координати точок виводу результатів, повний вектор початкових умов (оскільки в загальному випадку всі компоненти вектор-функції правих частин системи залежать від усього вектора розв'язку), похибка в початкових даних, необхідна точність отримання розв'язку та ін.

   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 $t_i = t_0 + \sum_{j=0}^{i-1} h_j$ $(i = 1, 2, \ldots)$ 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 $y^{(i)} + 0{,}5\,k_1$; 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 $y^{(i)} + 0{,}5\,k_2$, 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 GPU 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), обчислені компоненти вектора k_4 посилаються в CPU;
– в CPU обчислюється m компонент вектора $y^{(i+1)} = y^{(i)} + v/6$, де $v = k_1 + 2k_2 + 2k_3 + k_4$; 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 формулами чисельного інтегрування $\psi^{(i+1)} = \max\limits_{1 \le j \le n} \left| \bar{y}_j^{(i+1)} - \bar{\bar{y}}_j^{(i+1)} \right|$; похибка апроксимації посилається в CPU;

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

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

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. Зауважимо, що у кожному з np процесів матрично-векторні операції виконуються з використанням функцій CUBLAS. Тут описано алгоритм методу Рунге-Кутта для випадку, коли вектор-функція правої частини може бути легко обчислена на графічних прискорювачах. Якщо ж права частина дуже складна, то в описаному алгоритмі треба ігнорувати обчислення вектор-функції в GPU і залишити її обчислення в CPU.

Нижченаведена тестова здача розв'язувалась на експериментальному зразку паралельного комп'ютера на графічних процесорах, який складається з двох обчислювальних вузлів, з'єднаних комунікаційною мережею. У обчислювальному вузлі використовується чотириядерний процесор, на одне ядро приходиться одна карточка GPU. На кожному вузлі, як обчислювальні процесори, використовуються дві ігрові відеокарти.

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 $u_i(0) = 1$ $(i = 0, 1, 2, \ldots, n-1)$.

Нижче наведено таблицю, що містить часи розв'язування цієї задачі при n=20000 на експериментальному зразку паралельного комп'ютера гібридної архітектури (np CPU + np GPU). Для порівняння наведено час розв'язування задачі при використанні паралельних алгоритмів для середовища MPI.

Таблиця

	np=2	***np=4***	***np=8***
CPU+GPU	9,79	4,91	3,4
MPI	40,73	24,48	15,59

Зауважимо, що час розв'язування задачі на CPU становить 78 с. 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.

Література

[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.