УДК 004.8

І.М. Шевчук
В.О. Денисюк

ДОСЛІДЖЕННЯ ТА РЕАЛІЗАЦІЯ ПАРАЛЕЛЬНОГО АЛГОРИТМУ РОЗВ'ЯЗАННЯ СИСТЕМИ НЕЛІНІЙНИХ РІВНЯНЬ МЕТОДОМ РУНГЕ-КУТТА 4-ГО ПОРЯДКА ТОЧНОСТІ

Вінницький національний технічний університет

Анотація

Розглянуто розробку паралельного алгоритму розв'язання системи нелінійних диференціальних рівнянь методом Рунге-Кутта 4-го порядку з використанням технології multiprocessing. Розглянуто питання аналізу чисельних методів для досягнення високої точності, обґрунтовано вибір моделі геометричного паралелізму (SPMD), розроблено архітектуру програмного забезпечення та схему декомпозиції даних. У роботі створено програмну реалізацію паралельного алгоритму мовою Python з використанням механізмів спільної пам'яті (Shared Memory) та проведено тестування його продуктивності на системі великої розмірності. Використання результатів дозволить покращити швидкодію моделювання складних динамічних систем, які описуються диференціальними рівняннями.

Ключові слова: *паралельні обчислення, метод Рунге-Кутта, multiprocessing, Python, спільна пам'ять.*

Abstract

The development of a parallel algorithm for solving a system of nonlinear differential equations using the Runge-Kutta 4th order method using multiprocessing technology is considered. The issue of analyzing numerical methods to achieve high accuracy is considered, the choice of the geometric parallelism model (SPMD) is justified, the software architecture and data decomposition scheme are developed. In the work, a software implementation of a parallel algorithm in Python using Shared Memory mechanisms is created and its performance is tested on a large-scale system. Using the results will allow improving the speed of modeling complex dynamic systems described by differential equations.

Keywords: *parallel computing, Runge-Kutta method, multiprocessing, Python, shared memory..*

Вступ

Актуальність реалізації паралельного алгоритму методу Рунге-Кутта полягає у тому, що сучасні задачі фізики, біології та інженерії вимагають моделювання систем великої розмірності ($N \geq 10^6$), що є критично ресурсомістким для послідовних обчислень. Зростання обсягів даних та вимог до точності призводить до того, що час розрахунків на одному процесорному ядрі стає неприпустимо великим, обмежуючи можливості дослідників.

У таких умовах використання паралельних архітектур стає єдиним ефективним шляхом підвищення продуктивності. Розподіл обчислювального навантаження між ядрами багатоядерних процесорів дозволяє суттєво скоротити час отримання результату, зберігаючи при цьому високу точність методу ($O(h^4)$). Це забезпечує ефективність розв'язання задач Коші не лише для масивних наборів даних, але й у системах реального часу, де швидкість реакції є критичною.

Метою роботи є розробка та програмна реалізація ефективного паралельного алгоритму розв'язання систем нелінійних рівнянь методом Рунге-Кутта 4-го порядку для підвищення продуктивності обчислювальних систем.

Постановка задачі дослідження

Задачі дослідження полягають у вирішенні наступних питань:

- аналіз методів чисельного інтегрування та вибір оптимального за критерієм точності;

- обґрунтування моделі розпаралелювання (декомпозиція за даними);
- мінімізація накладних витрат на синхронізацію процесів та обмін даними;
- розробка програми мовою Python з використанням бібліотеки multiprocessing;
- тестування програми та аналіз показників прискорення та ефективності.

Виклад основного матеріалу

Математичне моделювання складних динамічних систем у фізиці, хімії, біології та інженерії часто зводиться до розв'язання задачі Коші для систем звичайних диференціальних рівнянь. Серед існуючих однокрокових чисельних методів, таких як метод Ейлера (точність $O(h)$) та його модифікації, метод Рунге-Кутта 4-го порядку є стандартом де-факто завдяки високій точності ($O(h^4)$) та чисельній стійкості на великих інтервалах інтегрування [1]. Однак його обчислювальна складність є значною: на кожному кроці інтегрування необхідно чотири рази обчислювати праву частину системи рівнянь, що для систем великої розмірності ($N \geq 10^6$) призводить до критичних часових витрат при послідовному виконанні.

Для вирішення проблеми швидкодії було обрано стратегію геометричного паралелізму в рамках архітектури MIMD (Multiple Instruction, Multiple Data) з використанням моделі SPMD (Single Program, Multiple Data). Суть підходу полягає у декомпозиції області даних: загальний вектор стану системи розбивається на рівні безперервні блоки, які розподіляються між доступними обчислювальними вузлами [2]. Кожен процес виконує ідентичний алгоритм над своєю частиною даних, що забезпечує статичне балансування навантаження та рівномірне використання ресурсів процесора [3].

Аналіз інформаційного графа алгоритму показав наявність жорстких інформаційних залежностей між стадіями обчислень. Знаходження коефіцієнтів k_{i+1} (наприклад, k_2) неможливе без знання значень k_i (k_1) для всієї системи рівнянь, оскільки рівняння можуть бути взаємозв'язаними. Це зумовлює необхідність використання точок глобальної синхронізації (бар'єрів) після кожного підкроку методу [4]. Така структура алгоритму накладає підвищені вимоги до швидкості міжпроцесорної комунікації та мінімізації латентності при обміні даними.

Програмну реалізацію виконано мовою Python, яка є стандартом для наукових обчислень завдяки потужним бібліотекам [5]. Враховуючи архітектурні обмеження Global Interpreter Lock (GIL) стандартного інтерпретатора CPython, для організації істинного паралелізму використано бібліотеку multiprocessing, яка дозволяє створювати незалежні процеси операційної системи, кожен з яких має власний простір пам'яті [6].

Критично важливим етапом оптимізації стало використання механізму спільної пам'яті (Shared Memory) через об'єкти multiprocessing.Array. Це дозволило уникнути накладних витрат на серіалізацію (pickling) та пересилання великих масивів даних між процесами через канали (Pipes) або черги (Queues), замінивши їх на прямий доступ до спільних областей оперативної пам'яті [7]. Для додаткового прискорення арифметичних операцій всередині кожного процесу застосовано векторизацію обчислень засобами бібліотеки NumPy, що дозволяє використовувати SIMD-інструкції сучасних процесорів [8].

Аналіз результатів тестування програми виконано на тестовій системі з 1 000 000 рівнянь (табл.1-3).

Таблиця 1 – Час виконання програми на різних конфігураціях

Кількість процесів	Час виконання	Примітка
1	41.6498	Послідовний алгоритм
2	25.4192	Паралельний алгоритм
4	22.3604	Паралельний алгоритм
8	21.4831	Паралельний алгоритм

Таблиця 2 – Коефіцієнти прискорення

Кількість процесів			
Коефіцієнт прискорення	.	1.86	1.94

Таблиця 3 – Коефіцієнти ефективності

Кількість процесів			
Коефіцієнт ефективності	0.82	0.465	0.242

Проаналізувавши обчислені коефіцієнти прискорення та коефіцієнти ефективності при різних кількостях процесів (табл.1-3) можливо зробити висновки:

- використання паралельних обчислень дозволило скоротити час виконання задачі майже вдвічі (з 41.65 с до 21.48 с), що є значним результатом для задач такого класу;
- найбільша ефективність спостерігається при використанні 2 процесів (82%), де накладні витрати на створення процесів та синхронізацію є мінімальними порівняно з часом корисних обчислень;
- при збільшенні кількості процесів до 8 спостерігається ефект насичення прискорення ($S_p \approx 1.94$) та різке падіння ефективності до 24.2%. Це явище пояснюється законом Амдала: зі зростанням ступеня паралелізму частка часу, що витрачається на послідовні операції (бар'єрна синхронізація, управління доступом до спільної пам'яті), починає домінувати над часом безпосередніх обчислень.

Отже, розроблений паралельний алгоритм продемонстрував свою ефективність як інструмент для значного зменшення часу виконання ресурсомістких розрахунків, особливо при роботі з системами диференціальних рівнянь великої розмірності. Однак результати тестування виявили нелінійний характер зростання продуктивності, що обумовлено збільшенням накладних витрат на міжпроцесорну комунікацію та бар'єрну синхронізацію при масштабуванні системи. Це свідчить про необхідність ретельного підбору конфігурації обчислювального середовища та пошуку оптимального балансу між кількістю задіяних обчислювальних вузлів і обсягом вхідних даних для забезпечення максимального коефіцієнта корисної дії використання апаратних ресурсів.

Висновки

Досліджено метод Рунге-Кутта 4-го порядку як один із найбільш точних методів чисельного інтегрування, визначено доцільність його розпаралелювання для систем великої розмірності. Розглянуто різні підходи до організації паралельних обчислень, зокрема архітектуру SPMD та стратегію геометричного паралелізму, описано математичну модель та побудовано граф інформаційних залежностей алгоритму. На основі літературних джерел досліджено принципи роботи з бібліотекою multiprocessing та механізми управління спільною пам'яттю (Shared Memory) для мінімізації накладних витрат при обміні даними.

Програмно реалізовано задану задачу мовою Python та описано всі компоненти коду. Також наведено UML-діаграми. Проведено тестування розробленої програми на системі з 10^6 рівнянь, проаналізовано її результати, досліджено коефіцієнти прискорення та ефективності.

Визначено, що збільшення кількості процесів дозволяє досягти прискорення до 1.94 разів, однак подальше масштабування призводить до зниження ефективності через зростання витрат на бар'єрну синхронізацію.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Фельдман Л. П. Чисельні методи в інформатиці : підручник / Л. П. Фельдман, А. І. Петренко, О. А. Дмитрієва. К. : Видавнича група BHV, 2006. 480 с.
2. Jesper Larsson Träff. Lectures on Parallel Computing. 2024. 266p. URL: https://arxiv.org/pdf/2407.18795

3. Яровий А. А. Методи та засоби організації високопродуктивних паралельно-ієрархічних обчислювальних систем: монографія / А. А. Яровий. Вінниця : ВНТУ, 2016. 363 с.
4. Аксак Н. Г. Паралельні та розподілені обчислення : підручник / Н. Г. Аксак, О. Г. Руденко, А. М. Гуржій. Х. : Компанія СМІТ, 2009. 480 с.
5. Маккінні В. Python для аналізу даних. Обробка даних за допомогою Pandas, NumPy та IPython / В. Маккінні. К. : Наш Формат, 2019. 568 с.
6. Documentation for multiprocessing // Python 3.12.1 documentation. URL: https://docs.python.org/3/library/multiprocessing.html.
7. Дорошенко А. Ю. Паралельні обчислювальні системи : конспект лекцій / А. Ю. Дорошенко. К. : Видавничий дім «КМ Академія», 2013. 46 с.
8. NumPy User Guide // NumPy v1.26 Manual. URL: https://numpy.org/doc/stable/user/.

Шевчук Іванна Миколаївна – студентка кафедри комп`ютерних наук, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: theivanysik@gmail.com;

Денисюк Валерій Олександрович – канд. техн. наук, доцент, доцент кафедри комп'ютерних наук, Вінницький національний технічний університет, м.Вінниця, e-mail: vad64@i.ua.

Shevchuk Ivanna Mykolaivna – student of Computer Science Department, Faculty of Intelligent Information Technologies and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: theivanysik@gmail.com;

Denysiuk Valerii Olexandrovich – Ph.D., Assistant Professor, Assistant Professor of the Chair of Computer Science, Vinnytsia National Technical University, Vinnytsia, e-mail: vad64@i.ua