УДК 004.272.2:519.63

О.А. Дмитрієва, д-р техн. наук, проф.,
В.В. Кулаков, магістр,
В.Г. Гуськова, студентка
Донецький національний технічний університет
dmitrieva.donntu@gmail.com

Оптимізація комунікаційних операцій при керуванні кроком в паралельних екстраполяційних алгоритмах

У статті розглядаються питання керування кроком інтегрування при паралельному моделюванні динамічних об'єктів за допомогою екстраполяційних алгоритмів. Запропоновано підходи, орієнтовані на автоматичний вибір оптимального розміру кроку й порядку в кожній точці сітки. Розроблені алгоритми базуються на явних і неявних екстраполяційних методах і орієнтовані на мінімізацію обчислювальної роботи за одиничний крок. Обчислювальні вузли, що простоюють, використовуються для превентивного обчислення задачі у наступній точці зі зменшеним кроком інтегрування. При реалізації на кластерних системах за допомогою стандарту MPI оцінюється доцільність використання колективних операцій і операцій типу точка-точка. На основі побудованих алгоритмів виконана паралельна реалізація тестових задач.

Ключові слова: паралельне моделювання, стадійний метод, явна й неявна екстраполяція, адаптація кроку, кластерна архітектура.

Вступ

Сучасний етап наукових досліджень характеризується наявністю надпотужних обчислювальних технічних засобів, проте існує широкий клас завдань, вирішення яких за допомогою класичного (послідовного) моделювання відзначається неприйнятними часовими витратами, а також низькими показниками паралелізму [1]. В зв'язку з цим в світі спостерігається стрімке зростання кількості багатопроцесорних обчислювальних систем кластерного типу і їх сумарної продуктивності. У зв'язку з цим актуальними є задачі, що пов'язані, по-перше, з розробкою сучасних паралельних чисельних методів і адаптацією для паралельної реалізації існуючих [2] і, по-друге, з ефективною реалізацією на паралельних і розподілених засобах обчислювальної техніки різних прикладних програмних продуктів, що раніше розроблялися лише для розрахунку на комп'ютерах з класичною неймановською архітектурою [3-4]. Успішне розв'язання цих задач визначає один з найважливіших шляхів підвищення швидкості розв'язання складних і трудомістких задач, до яких, насамперед, треба віднести системи диференціальних рівнянь великої розмірності, які описують поведінку складних динамічних об'єктів із зосередженими параметрами [5-6]. При математичному моделюванні фізичних процесів часто виникає необхідність знаходження високоточного розв'язання задачі Коші для рівнянь і їх систем виду

$$x' = f(t,(x(t)), \quad x(t_0) = x_0, t \in [t_0, t_0 + T], \quad (1)$$

де праві частини $f(t,(x(t)): R^{m+1} \to R^m$ є достатньо гладкими функціями.

Отримання розв'язку необхідного порядку точності послідовними чисельними методами для таких задач може вимагає виконання значної кількості ітерацій, що унеможливлює розв'язання задач великої розмірності, жорстких. Одним із перспективних класів алгоритмів, які можуть застосовуватися для таких задач є алгоритми, що будуються на екстраполяційних методах [7]. Але при значному підвищенні порядку екстраполяційного методу починає позначатися накопичення обчислювальної похибки [8]. Підвищення розрядності операндів при обчисленнях ускладнюється у зв'язку з обмеженістю розрядності обчислювальних систем, а використання методів довгої арифметики тягне за собою значне зростання обчислювального навантаження [9]. Удосконалення екстраполяційних методів знаходження чисельного розв'язку задачі Коші надає можливість не лише зменшити час отримання результату але і значно підвищити розмірність та(або) точність розв'язання.

Метою роботи є модернізація і розпаралелювання існуючих чисельних алгоритмів екстраполяційного типу задля підвищення їх ефективності, точності обчислень, а також скорочення обчислювального навантаження, не вимагаючи підвищення потужності апаратної складової обчислювальної системи.

Паралельна реалізація екстраполяційного алгоритму розв'язання СЗДР

Розглядається клас алгоритмів, що здійснює паралельне керування кроком на основі явних і неявних екстраполяційних методів. Задаючи базову довжину кроку τ_n, обирається монотонно зростаюча послідовність натуральних чисел $k_1 < k_2 < ... < k_m$ з відповідними довжинами кроків $\tau_n^1 > \tau_n^2 > \cdots > \tau_n^m$, де

$$\tau_n^i = \frac{\tau_n}{k_i}, \quad i = 1,2,...m.$$

Уточнення наближень здійснюється за допомогою рекурентних формул

$$u_{i,l+1} = u_{i,l} + \frac{u_{i,l} - u_{i-1,l}}{\tau_n^{i-l} / \tau_n^i - 1}, i = 2,3,...m, \quad l = 1,2,...i-1.$$

Для чисельної реалізації будуються первинні екстраполяційні таблиці, значення в яких можуть бути отримані паралельно без обмінів на стадіях, якщо число процесорів у системі не менше, ніж кількість підходів до екстраполяції *m*. Оптимальний порядок екстраполяційного методу визначається, виходячи з вимог мінімізації обчислювальної роботи на кроці або шляхом рекурсивного оцінювання коефіцієнта головного члена глобальної помилки.

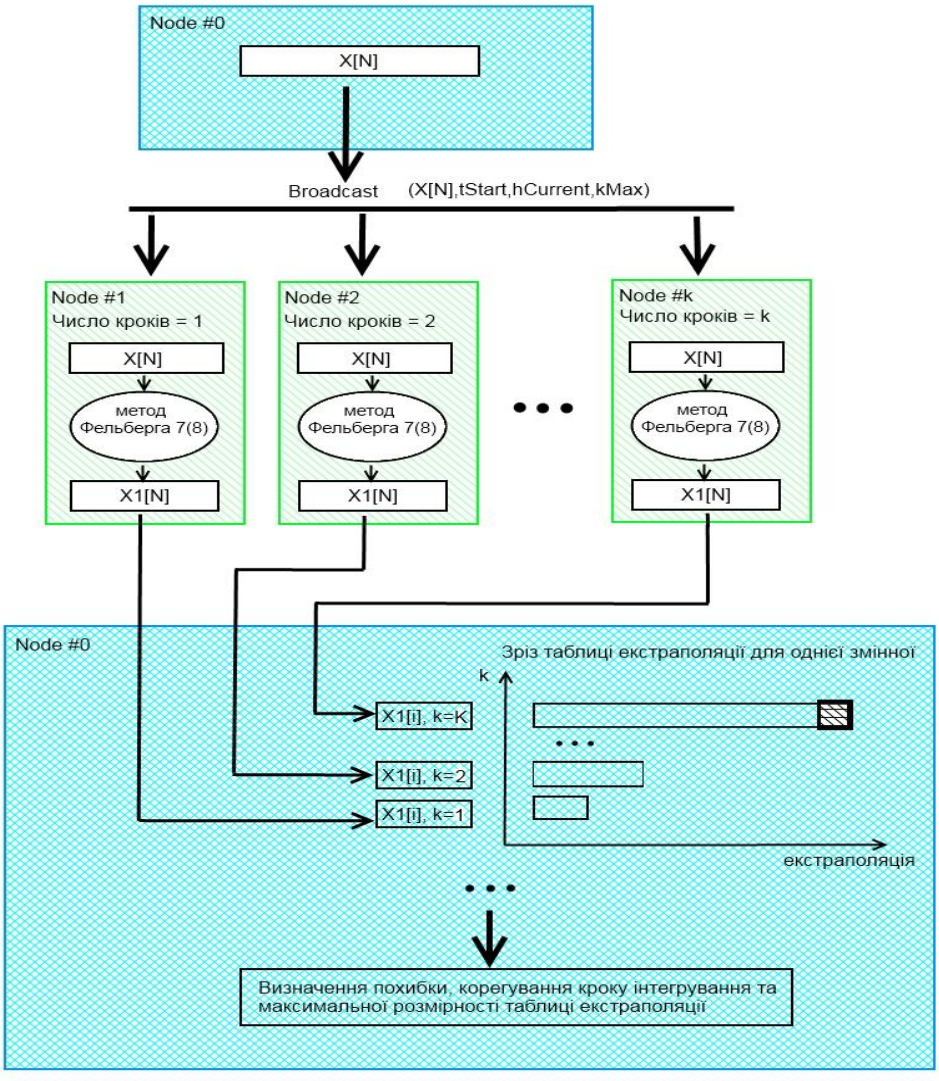

Рисунок 1 – Схема паралельного екстраполяційного алгоритму

Розрахунок значень для нової точки починається з паралельного формування наближення до розв'язання за допомогою *s*-стадійного методу порядку *p*. При цьому початковий крок уже сформований [10]. Відповідно до структури наявної обчислювальної системи (кластера) було обрано такий спосіб розгалуження алгоритму, за яким час виконання обчислювальної роботи домінує над часом виконання комунікаційних операцій. Таким чином, початкові дані розсилаються

від керуючого вузла керованим, які, у свою чергу, обчислюють результат розв'язання СЗДР однокроковим методом. Кожен вузол відповідно до свого рангу обирає кількість кроків для обчислення своєї частки задачі. Після завершення обчислень керовані вузли відсилають отримані результати керуючому, де відбувається екстраполяція. Після цього виконується оцінка апостеріорної похибки та, виходячи з цієї інформації, приймається рішення продовжувати обчислення із обраним кроком, чи крок потрібно зменшити. Також від отриманої похибки залежить, чи буде змінено максимальну розмірність таблиці екстраполяції.

Отримання та аналіз характеристик часу для розробленої системи

Для отримання характеристик розробленої програмної системи було проведено чисельні експерименти, в яких досліджувалися залежності часу реалізації від заданої точності розв'язання, розмірності СЗДР, кількості доступних обчислювальних вузлів. Результати експериментів, отриманих на 10 розрахункових вузлах, відображено на графіках (рис. 2-3).

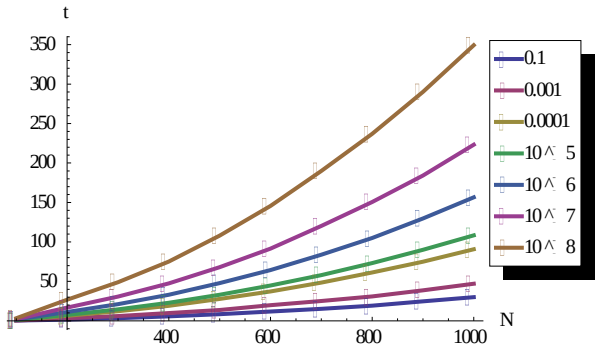

Рисунок 2 – Залежність часу отримання результатів (t) від розмірності системи (N)

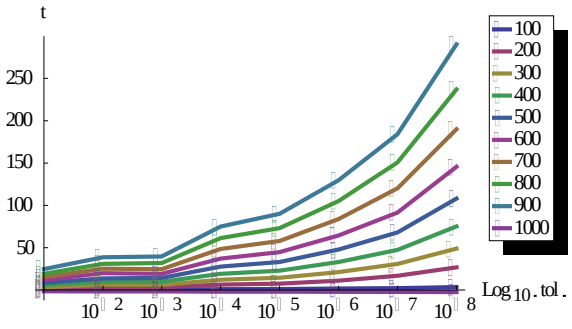

Рисунок 3 – Залежність часу отримання результатів (t) від заданої точності (tol)

Отримані дані вказують на те, що із зростанням кількості обчислювальних вузлів прискорення також зростає, але темпи цього зростання скорочуються. Ефективність у такому разі знижується. Також треба зазначити, що зростання прискорення є обмеженим через неможливість постійного збільшення розмірності таблиці екстраполяції. Якщо максимальна розмірність таблиці екстраполяції менше, ніж число наявних обчислювальних модулів, частина вузлів буде постійно простоювати, прискорення зростати не буде, а ефективність буде знижуватись.

Оптимізація комунікаційних операцій

У базовому варіанті (рис.1) паралельної реалізації екстраполяційного алгоритму для розповсюдження завдання по обчислювальних вузлах використовується операція *broadcast*. За однакових умов така розсилка даних буде виконуватись швидше за циклічне пересилання даних від керуючого вузла керованим.

```
void sendDataBCast(int kMax, double T,double
tStart, double *xArr)
{
        int MPIResult;
        MPI_Request req;
        double sendBuff[N+3];
                sendBuff[0] = (double)kMax;
        sendBuff[1] = T;
        sendBuff[2] = tStart;
                for (i=0;i<N;i++)
        {
                sendBuff[i+3] = xArr[i];
        }
        MPIResult = MPI_Bcast (sendBuff,N+3,
        MPI_DOUBLE_8,0,MPI_COMM_WORLD);
}
```

Рисунок 4 – Розсилка даних із використанням групової операції

Проте в нашому випадку ефект від використання операції broadcast може бути знівельовано. На початку функціонування алгоритму розмірність таблиці екстраполяції є невеликою, тому деяка кількість наявних обчислювальних вузлів буде простоювати. Для використання групової операції пересилки даних її повинні виконати всі наявні вузли. Таким чином, вузли, про які наперед відомо що вони не будуть брати участі у розрахунках на даному етапі, все одно повинні отримати вхідні дані від керуючого вузла. Ця обставина здатна збільшити навантаження на мережну підсистему та підвищити час виконання алгоритму в цілому.

Відмова від колективної операції пересилання даних та застосування циклічної відправки початкових даних лише тим вузлам, результати роботи яких будуть дійсно потрібні, дасть можливість більш гнучко налаштовувати алгоритм, при цьому не завдаючи шкоди швидкості виконання. Якщо у системі існують такі обчислювальні вузли, що не можуть бути використаними,

то у такому випадку за оновленим алгоритмом, вільний вузол (вузли) буде знаходитись у режимі очікування, доки керуючий вузол не відправить йому вхідні дані (рис. 5). У такому стані керований вузол може чекати на надходження даних від керуючого вузла навіть протягом багатьох ітерацій.

```
void slaveMethod(FuncFactory *ff)
{
    int i,j;
    int MPIResult;
    double recvBuff[N+3];
    MPI_Recv(recvBuff,N+3,
MPI_DOUBLE_8,0,ProcRank,
MPI_COMM_WORLD,&status);
```

Рисунок 5 – Очікування та отримання керованим вузлом даних від керуючого

Запобіжне скорочення кроку інтегрування

Як результат автоматичного регулювання кроку інтегрування та розмірності таблиці екстраполяції, під час роботи алгоритму інколи виникає необхідність зменшити крок та відновити обчислення на даному етапі. Запропоноване покращення алгоритму полягає в тому, що наявні в системі, але не використовувані в базовому алгоритмі обчислювальні вузли використовуються для превентивного обчислення задачі на черговому такті зі заздалегідь зменшеним кроком інтегрування (рис. 6).

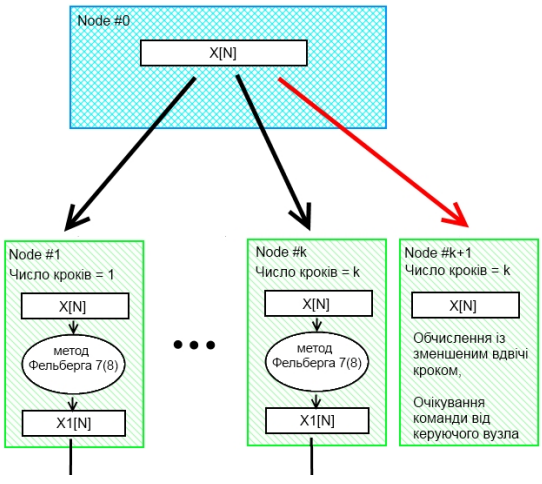

Рисунок 6 - Схема розсилки завдань у режимі передбачення кроку

Такий підхід у разі необхідності дозволяє розпочати обчислення для тієї самої точки але із зменшеним кроком, що забезпечує значне скорочення часу очікування результатів, які повинні надійти з керованих вузлів. Але для функціонування такої схеми необхідно на кожному кроці розсилати дані на вузли, результати роботи яких можуть і не знадобитися. Ті вузли, що виконують допоміжні обчислення, також отримують від керуючого вузла деякі специфічні команди, котрих вони очікують паралельно із виконанням обчислень. Оскільки обчислення за екстраполяційними методами є достатньо тривалими, для забезпечення можливості швидкого реагування вузлів попереднього розрахунку на команди метод модифікується. Модифікація полягає в додатковій можливості перевірки надходження команд та перериванні процесу обчислення. Перед початком обчислення вузол, що працює в режимі передбачення, підготовує асинхронне отримання команди від керуючого вузла (використовуючи команду MPI_Irecv). Протягом проведення обчислень статус цієї операції постійно перевіряється. Отримавши команду із кодом 1, такий вузол переривається виконання обчислень та повертається до початкового стану. Отримання команди із кодом 2 означає необхідність довести обчислення до кінця та передати результат керуючому вузлу. У такому разі подальші перевірки на надходження команд не відбуваються. Функцію розсилання даних також було модифіковано для того, щоб зменшити час очікування передачі інформації на вузли попереднього розрахунку. Відправка даних таким вузлам здійснюється за допомогою асинхронної передачі, що дає можливість розвантажити мережну підсистему. У табл. 1 наведено характеристики часу виконання паралельного алгоритму із використанням передбачення скорочення кроку та без нього на прикладі розв'язання СЗДУ із 1000 рівнянь із припустимою похибкою $1*10^{-8}$.

Таблиця 1. Порівняльний аналіз методів

Кількість обчислювальних вузлів	Метод без передбачення кроку, сек.	Метод із передбаченням кроку, сек.
4	558,74	558,67
5	452,7	452,8
6	396,79	396,79
7	352,04	352,07
8	324,62	324,65
9	304,52	304,5
10	289,76	289,56
11	280,06	274,27
12	264,66	261,19
13	258,79	253,71
14	258,88	252,08

Після додавання 15-го вузла, час виконання алгоритму без передбачення скорочення кроку не зменшився. Це означає, що останній обчислений рядок таблиці екстраполяції має номер

14, можливості по розгалуженню екстраполяції такого алгоритму на цьому вичерпані.

Прискорення від використання такої модифікації можна спостерігати не тільки в тих випадках, коли кількість наявних обчислювальних модулів перевищує максимальну розмірність таблиці екстраполяції для усієї задачі. Ефект буде відчутний, якщо хоча б на декількох ітераціях, доки є вільні обчислювальні вузли, з'явиться необхідність дроблення кроку інтегрування. Покращений алгоритм у певних ситуаціях демонструє прискорення на рівні 2 - 7% у порівнянні із базовим алгоритмом. Такі властивості алгоритму можуть бути корисними, коли є достатня кількість ресурсів, та на передній план виходить задача скорочення часу, що витрачається на розв'язання СЗДР.

Висновки

Дослідження, що представлені в роботі, сприяють підвищенню ефективності моделювання з використанням паралельних комп'ютерів за рахунок скорочення числа обмінів при паралельній чисельній реалізації рішення задачі Коші за допомогою екстраполяційних методів з керуванням кроком інтегрування. Спроектовано та побудовано програмну систему, що реалізує цей алгоритм та дозволяє проводити масштабні випробування завдяки наявності модуля генерації СЗДР. Проведено експеримент, за результатами якого отримані характеристики часу виконання даного алгоритму на ОЕМ із кластерною архітектурою. Розраховані показники прискорення та ефективності для даного алгоритму, найбільше прискорення досягається при використанні значної кількості вузлів, але ефективність при цьому падає. Покращено паралельний екстраполяційний алгоритм шляхом додавання в нього підтримки передбачення скорочення кроку інтегрування. Таке вдосконалення дозволяє розширити можливості розгалуження базового алгоритму та отримати додаткове прискорення.

Аналізуючи результати експерименту можна зауважити, що найбільше прискорення від використання передбачення дроблення кроку припадає на ту область, де ефективність базового алгоритму різко спадає, а використання додаткових обчислювальних вузлів не призводить до скорочення часу виконання програми.

Список літератури

1. Dmitrieva O. Parallel Algorithms of Simulation. Increase of simulation of dynamic objects with the lumped parameters into parallel computer systems / O. Dmitrieva, A. Firsova. – Lambert Academic Publishing, 2012. – 192 p. – ISBN-13: 978-3-659-28540-0, ISBN-10: 3659285404.
2. Дмитриева О.А Параллельное моделирование динамических объектов с автоматическим выбором шага на основе экстраполяционных методов / О. А. Дмитриева // Радіоелектронні і комп'ютерні системи. – 2012. – № 6 (58). – С. 103–108.
3. Дмитрієва О.А. Паралельні різницеві методи розв'язання задачі Коші / О.А Дмитрієва. – Донецьк: ДонНТУ, 2011. – 265 с. – ISBN 978-966-377-104-5.
4. Дмитриева О.А. Параллельные экстраполяционные алгоритмы моделирования динамических систем / О. А. Дмитриева // Тезисы докладов 7-й научно-практической конференции с международным участием «Математическое и имитационное моделирование систем», Чернигов, 25–28 июня 2012. – Чернигов: ЧГТУ, 2012. – С. 403–406.
5. Дмитриева О.А. Параллельное управление шагом на основе экстраполяционных методов / О. А. Дмитриева // Материалы IX Международной конференции по неравновесным процессам в соплах и струях, Алушта, 25–31 мая 2012. – М.: Вузовская книга, 2012. – С. 485–487.
6. Хайрер Э. Решение обыкновенных дифференциальных уравнений. Нежесткие задачи / Э. Хайрер, С. Нерсетт, Г. Ваннер. – М.: Мир, 1990. – 512 с. – ISBN 5-03-001179-X.. - М.: Мир, 1990.- 512с.
7. Хайрер Э. Решение обыкновенных дифференциальных уравнений. Жесткие задачи / Э. Хайрер, Г. Ваннер. – М.: Мир, 1999. – 685 с. – ISBN 5-03-003117.
8. Butcher J. C. ARK methods for stiff problems / J. C. Butcher, N. Rattenbury // Applied Numerical Mathematics. – 2005. – Vol. 53, № 1. – P. 165–181.
9. Butcher J. C. Numerical methods for ordinary differential equations / J. C. Butcher. – Wiley-VCH Verlag, 2003. – 418 p. – ISBN 0-471-96758-0.
10. Дмитриева О.А. Параллельный контроль размера шага на основе коллокационных методов с использованием интерполяционных полиномов Эрмита / О. А. Дмитриева // Искусственный интеллект. – 2013. – № 3 (61). – С. 488 – 494.

Надійшла до редакції 11.03.2014

О.А. ДМИТРИЕВА, В.В. КУЛАКОВ, В.Г. ГУСЬКОВА
Донецкий национальный технический университет

ОПТИМИЗАЦИЯ КОММУНИКАЦИОННЫХ ОПЕРАЦИЙ ПРИ УПРАВЛЕНИИ ШАГОМ В ПАРАЛЛЕЛЬНЫХ ЕКСТРАПОЛЯЦИОННЫХ АЛГОРИТМАХ

В статье рассматриваются вопросы управления шагом интегрирования при параллельном моделировании динамических объектов с помощью экстраполяционных алгоритмов. Предложены подходы, ориентированные на автоматический выбор оптимального размера шага и порядка в каждой точке сетки. Разработанные алгоритмы базируются на явных и неявных экстраполяционных методах и ориентированы на минимизацию вычислительной работы за единичный шаг. Предложено простаивающие вычислительные узлы использовать для превентивного вычисления задачи в очередной точке с уменьшенным шагом интегрирования. При реализации на кластерных системах с помощью стандарта MPI оценивается целесообразность использования коллективных операций и операций типа точка-точка. На основе построенных алгоритмов выполнена параллельная реализация тестовых задач.

Ключевые слова: параллельное моделирование, стадийный метод, явная и неявная экстраполяция, адаптация шага, кластерная архитектура.

O.A. DMITRIEVA, V.V. KULAKOV, V.G. GUSKOVA
Donetsk National Technical University

OPTIMIZATION OF COMMUNICATION OPERATIONS IN STEP CONTROL IN PARALLEL EXTRAPOLATION ALGORITHMS

The article considers the problems of integration step control for parallel simulation of dynamic objects using extrapolation algorithms. We suggest an approach focused on the automatic selection of the optimal step size and order at each grid point. These algorithms are based on explicit and implicit extrapolation methods and focused on minimizing the computational work for a single step. We suggest using idle compute nodes for preventive computing tasks with reduced integration step. This approach allows reducing significantly the time of waiting for the results.

When implementing on a cluster system using the standard MPI we evaluated the usefulness of collective operations and point-to-point operations. On the basis of these algorithms we made parallel implementation of the test problems.

Key words: parallel modeling, stage method, explicit and implicit extrapolation, step adaptation, cluster architecture.