УДК 004.75

М.О. Алєксєєв, канд. техн. наук, Л.С. Глоба, д-р техн. наук, К.О. Єрмакова, В.В. Кушнір

Застосування паралельних обчислень при розрахунку якості обслуговування черг заявок

В статье рассматривается проблематика использования высокопроизводительных вычислений в научных и инженерных исследованиях. Описывается метод эффективного распараллеливания процесса обработки информации. Анализируется эффективность применения высокопроизводительных вычислений с использованием библиотеки OpenMP при решения задач в области телекоммуникаций на примере расчета показателей качества обслуживания очередей заявок.

The usage of high-performance computing technologies in the areas of scientific and engineering researches is considered. The method of the effective data processing paralleling is described. The using of high-performance computing based on the OpenMP library for solving problems in the field of Telecommunication, e.g. computation of the queues QoS parameters, is also analyzed.

Ключевые слова: *OpenMP, високопродуктивні обчислення, функціональний паралелізм, паралелізм даних, обслуговування черг заявок.*

Вступ

Розробка алгоритмів та методів паралельних обчислень для розв'язку складних науково-технічних завдань часто являє собою значну проблему. Причина полягає в тому, що перехід до використання багатопроцесорних систем вимагає розв'язання певної кількості специфічних завдань. Необхідним є структурний аналіз алгоритму, виявлення його внутрішнього паралелізму, що вимагає глибокого розуміння суті завдання. Здатність алгоритму до розпаралелювання потенційно пов'язана з одним з двох або одночасно з обома внутрішніми властивостями, які характеризуються як функціональний паралелізм та паралелізм даних. Для того, щоб повною мірою використати структурні властивості алгоритму, необхідно перш насамперед виявити, до якого типу він відноситься.

При наявності в алгоритмі властивості паралелізму даних одна операція може виконуватися відразу над усіма елементами масиву даних. У цьому випадку різні фрагменти масиву можуть оброблятися незалежно на різних процесорах. Для алгоритмів цього типу розподіл даних між процесорами зазвичай здійснюється до виконання завдання на ЕОМ.

В випадку функціонального паралелізму обчислювальна задача розбивається на окремі, відносно самостійні підзадачі, кожна з яких завантажується на окремий процесор. Кожна підзадача реалізується незалежно, але використовує спільні дані та обмінюється результатами своєї роботи з іншими підзадачами. Для реалізації такого алгоритму на багатопроцесорній системі необхідно виявляти незалежні підзадачі, які можуть виконуватися паралельно. Часто це виявляється далеко не прозорим та вкрай важким завданням.

Питанням паралельних обчислень присвячена значна кількість наукових робіт, але більшість з них [1, 2, 3, 4, 5] розглядає загальний підхід щодо паралельної обробки даних та підвищення продуктивності обчислювального процесу в цілому. Разом з тим для реальних систем, які працюють в інформаційно-телекомунікаційних мережах, необхідно провести детальний аналіз конкретних алгоритмів з метою досягнення максимального підвищення продуктивності.

1. Застосування паралельних обчислень при розв'язуванні задачі розрахунку показників якості обслуговування черг заявок

На сьогоднішній день розроблені та застосовуються на практиці різні способи обробки черг, від самого простого FIFO до алгоритмів орієнтованих на обслуговування окремих потоків або класів трафіків. У роботі [6] доведено, що спосіб обробки черг істотно впливає на показники якості обслуговування мультисервісного трафіку, перш за все, - на затримку повідомлень. Ефективний розподіл канального ресурсу між різними типами інформаційних потоків є необхідною умовою управління. Таким чином, актуальною є задача розрахунку показників якості обслуговування черг заявок з найменшими часовими затримками.

Розглянемо головні чинники, які впливають на процес обробки черг заявок в комунікаційному центрі мережі провайдера зв'язку. Перш за все, усі заявки, які поступають на обслуговування є заявками від передачі відео-, аудіо- трафіку, а також трафіку даних. Як високопріоритетні за-

явоки розглядається відео- та аудіо-трафік, а низькопріоритетні – трафік даних. Для Інтернет трафіку характерною є самоподібна модель розподілу вхідного інформаційного потоку, що визначає практично непередбачувані піки зростання навантаження. З метою ефективної передачі самоподібного трафіку в періоди пікового навантаження у комунікаційному центрі [6] пропонується використовувати вдосконалений алгоритм обробки черг, який є модифікацією алгоритму зваженого кругового обслуговування (WRR – Weighted Round Robin).

Приведемо розрахунок затримки повідомлень високопріоритетного та низькопріоритетного трафіку в комутаційному центрі на прикладі роботи базової станції мережі UMTS. Вихідні дані є середньостатистичними показниками для даної базової станції:

- максимальна кількість високопріоритетних заявок, N1 = 29;
- максимальна кількість низькопріоритетних заявок, N2 = 35;
- кількість обслуговуючих пристроїв, які виділені для високопріоритетного трафіку, k = 8;
- кількість пакетів в черзі, *l* = 18;
- кількість обслуговуючих пристроїв для низькопріоритетного трафіку, які виділяються для високопріоритетного у разі переповнення черги довжини *l* з m = 3;
- кількість обслуговуючих пристроїв для низькопріоритетного трафіку, n = 6;
- інтенсивність обслуговування високопріоритетних заявок, μ1 = 3 1/с;
- інтенсивність обслуговування низькопріоритетних заявок, μ2 = 450 1/с;
- інтенсивність надходження високопріоритетних заявок, λ1 = 20 1/с;
- інтенсивність надходження низькопріоритетних заявок, λ2 = 500 1/с.

Середня затримка повідомлень високопріоритетного трафіку в комутаційному центрі визначається як:

$$T_1 = \frac{K_1}{\lambda_1} \qquad (1)$$

де K1 – середня кількість високопріоритетних заявок в системі, яка визначається як:

$$K_1 = \sum_{i=0}^{N_1} \pi_i i \qquad (2)$$

Розподіл ймовірностей знаходження в системі високопріоритетних заявок виз-начається через розв'язання наступної системи рівнянь (3):

$$\begin{cases} \pi_i = \begin{cases} \pi_0 \dfrac{\lambda^i}{\mu_1(2\mu_1)\cdots(i\mu_1)}, & i \le k \\ \pi_0 \dfrac{\lambda^i}{\mu_1^i \kappa! k^{i-k}}, & k < i \le k+l \\ \pi_0 \dfrac{\lambda^i}{\mu_1^i \kappa^l (r-l)!}, & k+l < i < k+l+m \\ \pi_0 \dfrac{\lambda^i}{\mu_1^i \kappa^l (k+m)!(k+m)^{i-(k+m+l)}}, & k+l+m \le i \end{cases} \\ \sum_{i=0}^{N_1} \pi_i = 1 \end{cases} \qquad (3)$$

Розрахунок затримки низькопріоритетного трафіку є більш складним через те, що процес надходження заявок низькопріорітетного трафіку не є Марківським, а залежить від заявок високопріоритетного трафіку, які поступають на обслуговування. Середня затримка повідомлень низькопріоритетного трафіку дорівнює:

$$T_2 = \frac{K_2}{\lambda_2}, \qquad (4)$$

де K_2 – середня кількість низькопріоритетних заявок в системі, яка визначається залежністю (5):

$$K_2 = \sum_{i=0}^{N_1} \sum_{j=0}^{N_2} \pi_{ij} j \qquad (5)$$

$$\begin{cases} \pi_{0,j}(\lambda_{(i-1,j\to i,j)} + \lambda_{(i,j-1\to i,j)}) = \\ = \pi_{1,j}\lambda_{(1,j\to 0,j)} + \pi_{0,j+1}\lambda_{(0,j+1\to 0,j)} \\ \pi_{i,0}(\lambda_{(i-1,j\to i,j)} + \lambda_{(i,j-1\to i,j)}) = \\ = \pi_{i+1,0}\lambda_{(i+1,0\to i,j)} + \pi_{i,1}\lambda_{(i,1\to i,0)} \\ \pi_{i,j}(\lambda_{(i,j\to i+1,j)} + \lambda_{(i,j\to i,j+1)} + \lambda_{(i,j\to i-1,j)} + \\ +\lambda_{(i,j\to i,j-1)}) = \pi_{i-1,j}\lambda_{(i-1,j\to i,j)} + \\ +\pi_{i,j-1}\lambda_{(i,j-1\to i,j)} + \pi_{i+1,j}\lambda_{(i+1,j\to i,j)} + \\ +\pi_{i,j+1}\lambda_{(i,j+1\to i,j)} \\ \pi_{N_1,j}(\lambda_{(N_1,j\to N_1,j+1)} + \lambda_{(N_1,j\to N_1-1,j)} + \\ +\lambda_{(N_1,j\to N_1,j-1)}) = \pi_{N_1-1,j}\lambda_{(N_1-1,j\to N_1,j)} + \\ +\pi_{N_1,j-1}\lambda_{(N_1,j-1\to N_1,j)} + \pi_{N_1,j+1}\lambda_{(N_1,j+1\to N_1,j)} \\ \pi_{i,N_2}(\lambda_{(i,N_2\to i+1,N_2)} + \lambda_{(i,N_2\to i-1,N_2)} + \\ +\lambda_{(i,N_2\to i,N_2-1)}) = \pi_{i-1,N_2}\lambda_{(i-1,N_2\to i,N_2)} + \\ +\pi_{i,N_2-1}\lambda_{(i,N_2-1\to i,N_2)} + \pi_{i+1,N_2}\lambda_{(i+1,N_2\to i,N_2)} \\ \sum_{i=0}^{N_1}\sum_{j=0}^{N2} \pi_{i,j} = 1 \end{cases} \qquad (6)$$

де $\lambda_{(i-1,j\to i,j)} = \lambda_1$, $\lambda_{(i,j-1\to i,j)} = \lambda_2$,

$$\lambda_{(i,j\to i-1,j)} = \begin{cases} \mu_1(2\mu_1)\cdots(i\mu_1), & i \le k \\ \mu_1^{\,i}\kappa!k^{i-k}, & k < i \le k+l \\ \mu_1^{\,i}\kappa^{l}(i-l)!, & k+l < i < k+l+m \\ \mu_1^{\,i}\kappa^{l}(k+m)!(k+m)^{i-(k+m+l)}, & k+l+m \le i \end{cases} \quad (7)$$

$$\lambda_{(i,j+1\to i,j)} = \begin{cases} \text{1. } \textit{Якщо} \quad i \le k+l & \\ i\mu_2 & \textit{а)} \quad j+1 \le m+n \\ (m+n)\mu_2 & \textit{б)} \quad j+1 > m+n \\ \text{2. } \textit{Якщо} \quad i \ge k+l+m & \\ (j+1)\mu_2 & \textit{а)} \quad j+1 \le n \\ n\mu_2 & \textit{б)} \quad j+1 > n \\ \text{3. } \textit{Якщо} \quad k+l < i < k+l+m & \\ (k+l+m-i+n)\mu_2 & \textit{а)} j+1 > k+l+m-i+n \\ (j+1)\mu_2 & \textit{б)} j+1 < k+l+m-i+n \end{cases} \quad (8)$$

У даному випадку розподіл ймовірностей знаходження в системі високопріоритетних та низькопріоритетних заявок розраховується шляхом рішення системи рівнянь (6).

Системи рівнянь (3) та (6) демонструють той факт, що розрахунок ймовірності переповнення черги, яка фактично оцінює ймовірність втрат заявок заданого типу, є нетривіальною задачею. Відео-, аудіо- та Інтерент трафіки одночасно, проте вимоги до якості обслуговування цих трафіків відрізняються. Кількість пакетів, яка забирається за одну транзакцію з черги Інтерент трафіку, є функцією від кількості пакетів відеотрафіку, які надійшли в систему. Розрахунок ймовірностей двох сильно залежних випадкових подій є складною математичною задачею, умови розв'язку якої досить жорсткі через необхідність обробки потоків заявок в комутаційному центрі в режимі реального часу. Виходячи з цього, для підвищення ефективності вирішення даної задачі пропонується використання паралельних обчислень.

Для побудови схеми обчислювального процесу спочатку проводиться аналіз вихідних даних з метою подальшої організації ефективного процесу обчислень у паралельному режимі.

Оскільки для поставленої задачі максимальна кількість високопріоритетних заявок N1 дорівнює 29, немає сенсу організовувати паралельне заповнення матриці-вектору $\{a_i\}_{i=\overline{1,N_1}}$ (рис. 1). Не потребує також паралельного виконання процес заповнення матриці $\{\lambda_{ij}\}_{i=\overline{1,N_1},\ j=\overline{1,N_2}}$ (рис. 1), оскільки її розмір для даної задачі дорівнює N1 × N2, а саме 29×35.

Проте доцільно виконувати обчислення підзадач розрахунку середніх затримок повідомлень високопріоритетного та низькопріоритетного трафіків у паралельному режимі.

Для розв'язання системи лінійних рівнянь застосовуються математичні методи, які базуються на роботі з матрицями, тому необхідно за даною системою побудувати матрицю-вектор невідомих величин, які треба знайти, матрицю коефіцієнтів та матрицю-вектор вільних членів. Багато елементів матриці-коефіцієнтів дорівнюють нульовим значенням, тобто матриця є розрідженою, отже заповнення даної матриці нульовими значеннями є однотипна операція, яка характеризується паралелізмом даних. Заповнення не нульовими значеннями даної матриці відноситься до функціонального паралелізму, оскільки рівняння системи (6) є незалежними одне від одного.

Проте для вирішення заданої системи рівнянь також необхідно враховувати різноманітні умови та припущення з боку предметної області, тому неможна простим загальноприйнятим методом, наприклад, методом Гауса, розв'язати таку систему. Алгоритм розв'язання системи рівнянь з паралельною схемою реалізації обчислювального процесу показаний на рис. 1.

2. Реалізація обчислювального процесу та аналіз отриманих результатів

Для проведення експерименту по визначенню ефективності організації виконання обчислювального процесу в паралельному режимі по заданій моделі обчислень був написаний програмний код на мові програмування С. Дослідження проводились на комп'ютерах на базі двохядерного процесора Genuine Intel, 1,3 ГГц та на базі чотирьохядерного процесора Intel Core i7, 2,67 ГГц під операційною системою Microsoft Windows 7. Розробка програм проводилась в середовищі Microsoft Visual Studio 2008 з підключенням бібліотеки OpenMP.

Результати виконання обчислювального процесу наступні:

- час виконання послідовних обчислень T_1 = 0,3164 с;
- час виконання паралельних обчислень на двохядерному процесорі T_2 = 0,2861 с;
- час виконання паралельних обчислень на чотирьохядерному процесорі T_4 = 0,1588 с.

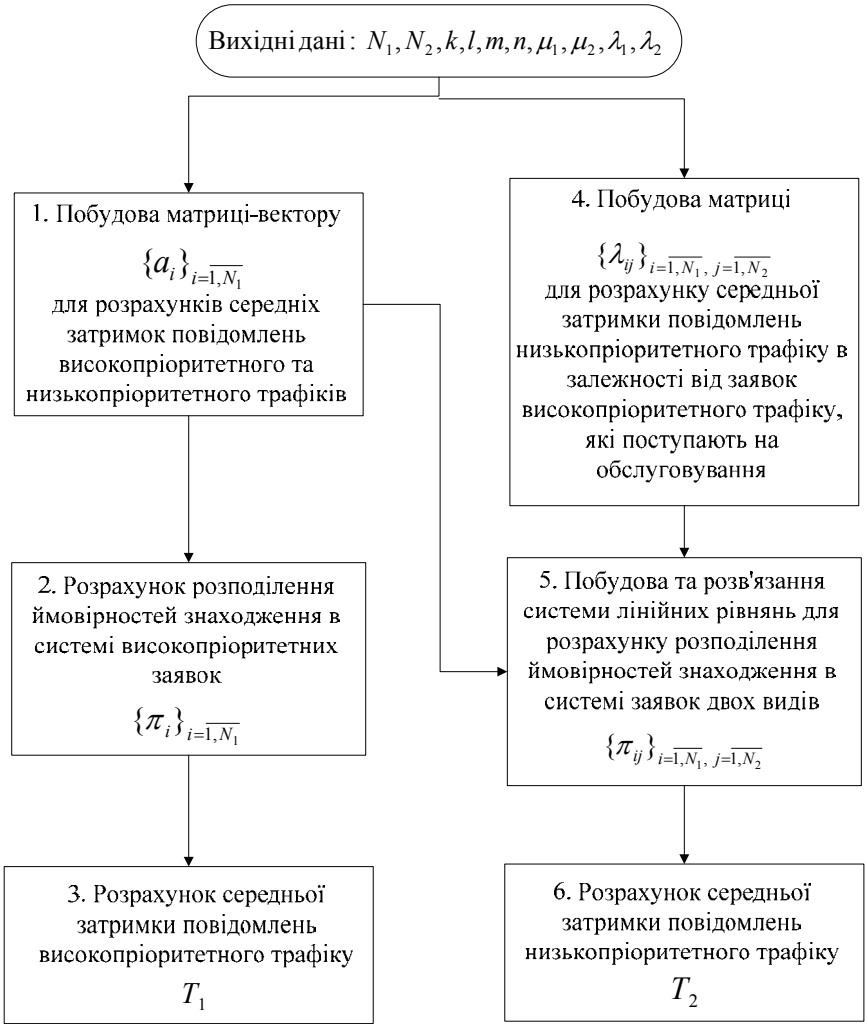

Рис. 1. Узагальнена схема обчислювального процесу

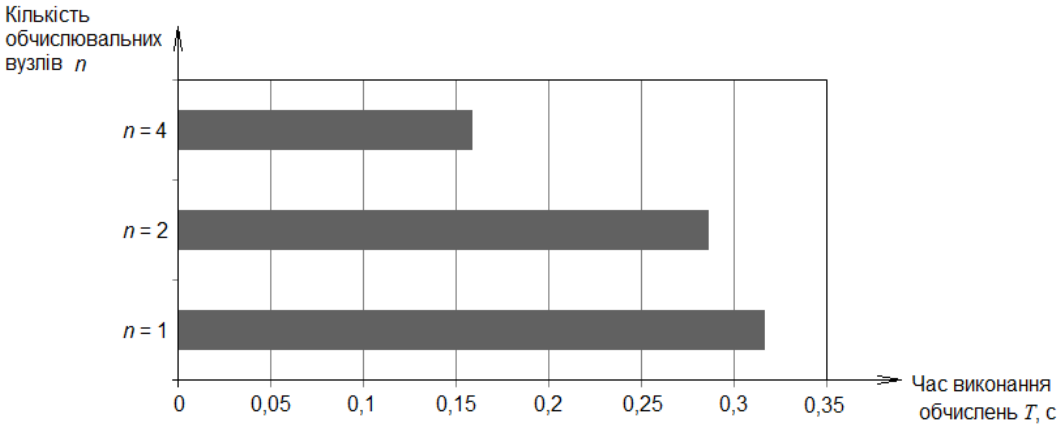

Рис. 2. Час виконання послідовного та паралельного обчислювальних процесів

Для оцінки отриманих результатів обчислюються коефіцієнти прискоренні та ефективності паралельних обчислень:

$$S_2 = \frac{T_1}{T_2} = \frac{0{,}3164\ \text{с}}{0{,}2861\ \text{с}} = 1{,}11; \qquad (9)$$

$$E_2 = \frac{S_2}{n} = \frac{1{,}11}{2} = 0{,}55; \qquad (10)$$

$$S_4 = \frac{T_1}{T_4} = \frac{0{,}3164\ \text{с}}{0{,}1588\ \text{с}} = 1{,}99; \qquad (11)$$

$$E_4 = \frac{S_4}{n} = \frac{1{,}99}{4} = 0{,}50; \qquad (12)$$

де S_2 та S_4 – коефіцієнти прискорення при виконанні обчислювального процесу на двохядерному та чотирьохядерному процесорах

відповідно, а E_2 та E_4 – коефіцієнти ефективності при виконанні обчислювального процесу на двохядерному та чотирьохядерному процесорах відповідно (рис. 3).

Найкращим показником коефіцієнта ефективності є одиниця. Аналізуючи рис. 3 та підраховані коефіцієнти ефективності під час виконання паралельного обчислювального процесу на двохядерному та чотирьохядерному процесорі, можна виявити, що алгоритм не є масштабованим, оскільки у разі зростання кількості ядер ефективність зменшується. Причиною цього є накладні витрати T_0 на організацію паралельних та послідовних секцій,та на синхронізацію потоків даних.

Це можна перевірити через підрахунку накладних витрат у разі використання двохядерного та чотирьохядерного процесорів.

$$T_{0\,(2)} = 2 \cdot T_2 - T_1 = 2 \cdot 0{,}2861 - 0{,}3164 = 0{,}2558; \quad (13)$$

$$T_{0\,(4)} = 4 \cdot T_4 - T_1 = 4 \cdot 0{,}1588 - 0{,}3164 = 0{,}3188; \quad (14)$$

де $T_{0(2)}$ та $T_{0(4)}$ – накладні витрати у разі виконання обчислювального процесу на двохядерному та чотирьохядерному процесорах відповідно. З отриманих результатів видно, що даний обчислюваний процес найбільш ефективно виконувати на двохядерному процесорі.

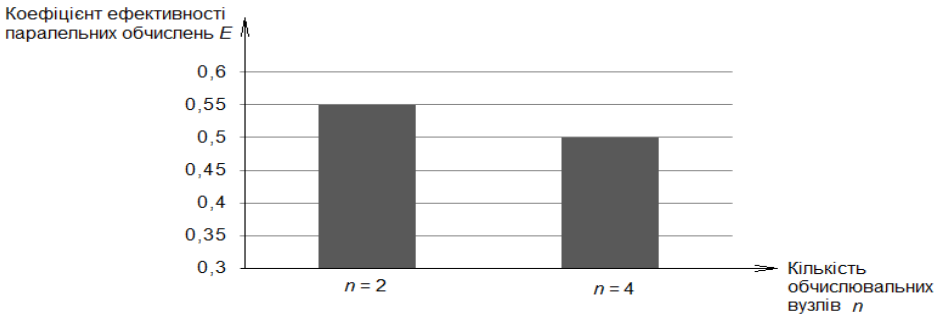

Рис. 3. Порівняння коефіцієнтів ефективності паралельних обчислень на двох- та чотирьохядерному процесорах

Висновки

1. Здатність алгоритму до розпаралелювання потенційно пов'язана з одним з двох або одночасно з обома внутрішніми властивостями, які характеризуються як функціональний паралелізм та паралелізм даних. Розглянутий алгоритм характеризується наявністю одночасно як паралелізмому даних, так і функціо-нального паралелізмому.

2. Аналіз отриманих результатів показав, що коефіцієнт ефективності виконання паралельного обчислювального процесу на двохядерному процесорі більший ніж на чотирьохядерному, тому даний алгоритм не є масштабованим, оскільки у разі зростання кількості ядер ефективність зменшується. Причиною цього є накладні витрати T_0 на організацію паралельних та послідовних секцій, а також на синхронізацію потоків даних.

3. Отже, не зважаючи на те, що коефіцієнт прискорення даного алгоритму на чотирьохядерному процесорі більший ніж на двохядерному дану задачу слід вирішувати на базі комп'ютера з меншою кількістю ядер.

Національний технічний університет України «Київський політехнічний інститут»

Література

1. *Інформаційно*-аналітичні матеріали по паралельним обчисленням [Електронний ресурс]. – Режим доступу: http://parallel.ru
2. *Гергель В.П.* Лабутина А.А. Учебно-образовательный комплекс по методам параллельного программирования. – Нижний Новгород: ННГУ им. Н.И. Лобачевского, 2007. – 138 с.
3. *Гергель В.П.,* Фурсов В.А. Лекции по параллельным вычислениям: учеб. пособие – Самара: Самар. гос. аэрокосм. ун-т, 2009. – 164 с.
4. *Dean J.* and Ghemawat S. MapReduce: Simplified data processing on large clusters. In Proceedings of Operating Systems Design and Implementation (OSDI). San Francisco, CA. 137-150, 2004
5. *Apache* Hadoop [Електронний ресурс]. – Електронні текстові дані – Режим доступу: http://hadoop.apache.org/, Friday, 15 January 2010 11:25:50.
6. *Скулиш М.А.* Модель управления информационными потоками в сетях с реализацией эмуляции постоянного соединения (RWE3) // СВЧ-техника и телекоммуникационные технологи: матер. 19-й междунар. науч.-техн. конф. КрыМиКо. – 2009. –С. 348–351.