УДК 519.6

І.А. Назарова[1], канд. техн. наук, доц.,
А.О. Чорна[2], студент
[1]Донецький національний технічний університет, м. Красноармійськ, Україна,
[2]Дніпропетровський національний університет, м. Дніпропетровськ, Україна
ianazard@gmail.com, anastasiyachorna@mail.ru

Дослідження масштабованості паралельних систем на основі функції ізоефективності

В статті представлено результати дослідження масштабованості паралельних систем з використанням функції ізоефективності. Засади ізоефективного аналізу застосовано для паралельних методів матричного добутку таких, як алгоритми Кеннона, Фокса, DNS та Штрассена. Розроблено модифікований алгоритм Кеннона для тривимірного тору. Визначено ступінь масштабованості алгоритмів та отримано пріоритетні області їх застосування залежно від розміру задачі та кількості процесорів для топологій 2D-тор і гіперкуб.

Ключові слова: паралельні алгоритми/архітектури, масштабованість, ізоефективний аналіз, прискорення, ефективність, загальні накладні витрати, матричний добуток.

Вступ

Сучасний етап розвитку обчислювальної техніки дозволяє будувати суперкомп'ютери, що використовують величезну кількість процесорів. Наявність таких комп'ютерних систем викликає інтерес у дослідженні продуктивності їх застосування для паралельного розв'язання реальних динамічних задач. Розумно було б очікувати зниження часу виконання, а отже й зростання якості паралельної обробки пропорційне обсягу витрачених ресурсів. Однак, зауважимо, що два широко відомих в практиці паралельних обчислень показника якості паралелізму, коефіцієнти прискорення та ефективності, не завжди дають однакові результати. Наприклад, збільшення числа процесорів обчислювальної системи в межах, що не перевищують максимальний ступінь паралелізму, часто призводить до росту прискорення, але, як правило, до зменшення ефективності [1]. Цей і численні факти свідчать про те, що запропоновані динамічні характеристики не є вичерпними, тобто не надають повної інформації про відображення паралельного алгоритму на певну архітектуру. Тому виникає необхідність у введенні нових, додаткових характеристик, таких як, наприклад, масштабованість паралельних алгоритмів у комбінації із високопродуктивною архітектурою, на якій його реалізовано [1-4].

В даній статті розглянуто існуючі підходи до визначення та оцінки масштабованості паралельних систем (алгоритм/архітектура), проведено огляд різноманітних метрик масштабованості, порівняння їх з найбільш вживаним методом ізоефективного аналізу. Технологію застосування цього математичного апарату продемонстровано на реальних паралельних алгоритмах і, відповідно, архітектурах. Розглянуто клас паралельних методів матричного добутку для щільно заповнених матриць та розроблено їх функції ізоефективності, що дозволяють порівняти алгоритми та вибрати пріоритетні області їх використання за кількістю процесорів та розміром задачі. Приведено приклади побудови функцій ізоефективності для паралельних методів інтегрування систем звичайних диференційних рівнянь. Для проведення експерименту застосовано топології двовимірний/тривимірний тор та гіперкуб. Паралельні програмні додатки реалізовано за допомогою бібліотеки MPI для мови програмування C++.

Математичний апарат ізоефективного аналізу

Поняття масштабованості паралельної системи може бути визначено по-різному. У декількох джерелах [1-4] визначення масштабованості можуть досить значно відрізнятися один від одного, відбиваючи той факт, що поняття масштабованості складне і вимагає ретельного аналізу.

Найбільш вживаною на даний момент є метрика, що розроблена авторами A. Gupta, V. Kumar [1-4], і заснована на введенні функції рівної ефективності або ізоефективності. Вводиться нова динамічна характеристика, яка має і самостійне значення для оцінки масштабованості, а саме, загальні накладні витрати, T_O. Вона містить сумарні витрати всіх

процесорів паралельної системи на реалізацію обмінів, послідовну частину розпаралелених алгоритмів та інші непродуктивні витрати.

Загальні накладні витрати обчислюються за наступною формулою:

$$T_o(m,p) = p \cdot T_p - T_1, \quad (1)$$

де T_1 – час для вирішення завдання заданого розміру на одному процесорі за допомогою найкращого послідовного алгоритму;

T_p – загальний час реалізації паралельного алгоритму на паралельній архітектурі:

$$T_p = T_{p,comp} + T_{p,comm} \quad (2)$$

$T_{p,comp}$ – час вирішення завдання заданого розміру m з використанням паралельного алгоритму на паралельному комп'ютері з p процесорів без урахування обмінних операцій.

$T_{p,comm}$ – час виконання міжпроцесорних операцій обміну при реалізації паралельного алгоритму розв'язання задачі заданого розміру. Ця характеристика враховує t_s – латентність, тривалість підготовки повідомлення для передачі та t_w – час передачі одного байту. Характеристикою продуктивності процесора є кількість операцій з плаваючою точкою, час виконання яких позначається t_{op}. Прийнято, що будь-який вид операції (додавання і добуток) виконуються за однаковий час $t_{ad} = t_{mul} = t_{op}$. Залежно від виду топології і комунікаційної операції, для розрахунку $T_{p,comm}$ використовується модель Хокні [1].

Нехай $E_p = const$ задає необхідний рівень ефективності використання паралельного обладнання, тоді:

$$T_o / T_1 = (1-E)/E,$$

$$T_1 = E/(1-E) \cdot T_0 = K(pT_p - T_1), \quad (3)$$

де $K = E/(1-E)$ – є коефіцієнт масштабованості, який залежить тільки від значення показника ефективності.

Породжувану останнім співвідношенням залежність $m = f_E(p)$ між складністю розв'язуваної задачі і числом процесорів називають функцією ізоефективності (isoefficiency function). Вираз (3) – центральне співвідношення ізоефективного аналізу. Функція f_E визначає, як треба збільшувати розмір задачі при збільшенні числа процесорів для підтримки постійної ефективності.

Паралельна система вважається масштабованою, якщо її функція ізоефективності існує [5]. Однак функція ізоефективності може бути дуже великою, тобто вона може давати дуже високі темпи зростання розміру проблемної задачі від кількості процесорів. На практиці розмір задачі може бути збільшено лише асимптотично зі швидкістю, що допускається обсягами пам'яті, доступними на кожному процесорі.

Таким чином, функція ізоефективності поєднує в одному аналітичному виразі характеристики задачі, методу вирішення, паралельної архітектури та дозволяє оцінити ступінь їх впливу всіх чинників на якість використання паралелізму.

Альтернативні базові метрики масштабованості паралельних систем

Масштабованість паралельного алгоритму на паралельній архітектурі є мірою його здатності ефективно використовувати велику кількість процесорів. Аналіз масштабованості може бути застосований, щоб вибрати найкращу комбінацію «алгоритм-архітектура» для задач при різних обмеженнях на зростання розміру задачі і числа процесорів. Також він використовується для оцінки ефективності реалізації паралельного алгоритму на заданій паралельній архітектурі і прогнозування поведінки алгоритму при збільшенні числа процесорів. При фіксованому розмірі задачі масштабованість може допомогти у визначенні оптимального числа процесорів, які будуть використовуватися і максимально можливого прискорення. Аналіз масштабованості може також передбачити вплив зміни апаратної технології на продуктивність системи і, таким чином, допомогти в розробці більш ефективних паралельних архітектур для вирішення різних динамічних завдань.

Серед альтернативних підходів для кількісної оцінки масштабованості паралельного алгоритму в сукупності з архітектурою, на якій його реалізовано, можна виділити такі як: масштабоване прискорення, послідовна частина, ізошвидкісна міра, середній показник паралелізму, рівні умови продуктивності, функції ефективності процесорів та ефективності даних тощо [2-10].

Масштабоване прискорення (scaled speedup) визначається, як прискорення, отримане для лінійно зростаючого розміру задачі із зростанням числа процесорів. Якщо крива масштабованості прискорення близька до лінійної, то паралельна система вважається масштабованою. Цей показник пов'язаний з ізоефективністю, якщо розглянутий паралельний алгоритм має лінійну або майже лінійну функцію ізоефективності [5].

Експериментальна послідовна частина f (serial fraction) – метрика, яка також призначена для вимірювання продуктивності паралельної системи [6]. Значення f дорівнює послідовній частині алгоритму s.

Застосовується, якщо втрата прискорення пов'язана тільки з послідовним компонентом (паралелізм не має інших накладних витрат), тоді, чим менше f, тим краще.

Функція ефективності процесорів і ефективності даних: PEF – processor efficiency function; DEF – data efficiency function. В якості верхньої межі на кількість процесорів p Чандран і Девіс ввели функцію ефективності процесора PEF [2], яка може бути використана для розв'язання задач вхідного розміру m таким чином, що час паралельного виконання на p процесорах того ж порядку, що і послідовний час, тобто $T_p = \Theta(T_1/p)$. Зворотна функція має назву функцією ефективності даних визначається як найменший розмір проблемної задачі на задане число процесорів, таких, що має місце співвідношення вище. Концепція цільової функції даних дуже схожа на функцію ізоефективності.

Середній показник паралелізму (average parallelism) A визначається як середнє число процесорів, які зайняті під час виконання паралельної програми, за умови їх доступності в необмеженій кількості. Після того, як аналітично або експериментально визначається A, прискорення і ефективність у системі з p процесорів обмежені знизу $(pA/(p+A-1))$ та $(A/(p+A-1))$ відповідно [8]. Середній показник паралелізму корисний, тільки якщо паралельна система не несе ніяких накладних витрат зв'язку (або якщо ці накладні витрати можуть бути проігноровані). Цілком можливо, що паралельний алгоритм з великим ступенем паралелізму гірше, ніж той же з меншим ступенем паралелізму і мінімальними накладними витратами зв'язку.

Для оцінки продуктивності паралельного комп'ютера Ван-Кетледжем була розроблена оцінка під назвою рівних умов продуктивності [9]. Ця модель служить для визначення відносної продуктивності програм паралельних комп'ютерів по відношенню до продуктивності тих же програм на суперкомп'ютерах. Рівні умови продуктивності визначаються як кількість процесорів, необхідних у паралельному комп'ютері, щоб порівняти його продуктивність з обраним суперкомп'ютером. Нову метрику масштабованості під назвою ізошвидкісна міра (isospeed) ввели Сан і Ровер [10]. Ця метрика визначає, як повинен збільшуватися розмір проблемної задачі, щоб середня одинична швидкість обчислень залишалася постійною при збільшенні кількості процесорів від p до p'.

З усіх методів оцінки масштабованості не можна вибрати один, який був би кращим, ніж інші, тому що різні методи підходять для різних ситуацій. Але той факт, що при виконанні певних умов, більшість метрик дають результати порівняні із характеристиками, що отримані за теорією ізоефективного аналізу, і пояснює пріоритетність використання цього апарату для вивчення масштабованості паралельних систем. Перевагами цього підходу є й те, що в одному рівнянні лаконічно фіксуються характеристики паралельного алгоритму та паралельної архітектури, на якій його реалізовано.

Аналіз масштабованості паралельних алгоритмів матричного добутку

В даній статті представлено результати застосування методу ізоефективного аналізу до паралельних алгоритмів множення щільно заповнених матриць із блоковим представленням даних – алгоритмів Кеннона, Фокса та DNS [1], а також для модифікованого алгоритму Кеннона для тривимірної решітки.

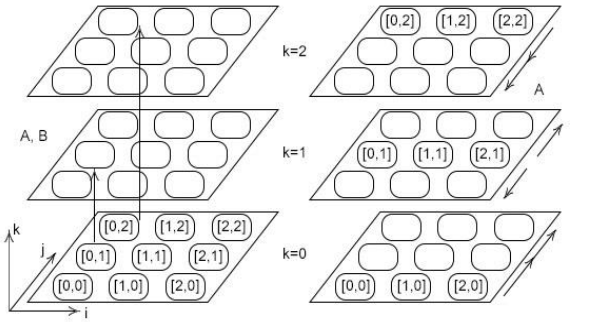
Алгоритм Кеннона – це паралельний алгоритм блокового систолічного множення квадратних матриць. Вихідні матриці мають розмірність $m \times m$ і розбиваються на квадратні матриці порядку $q, q^2 = p$ – кількість блоків та процесорів. На етапі ініціалізації відбувається циклічний зсув на 1 рядків матриці A косо вліво та стовбців матриці B вгору, крім перших. В основному циклі обчислюється добуток та додається до попередніх в матриці C, $q-1$ раз для всіх рядків виконується косий зсув блоків вліво, а стовпців – вправо та обчислюється $C_{ij} = C_{ij} + A_{ij} * B_{ij}$.

Початкові дані для алгоритму Фокса такі ж, як і для алгоритму Кеннона, але висхідні дані пересилаються по-іншому [4]. На етапі обчислень на кожній ітерації здійснюються такі операції: для кожного рядка i блок A_{ij} пересилається на всі підзадачі того ж рядка сітки i, де індекс $j = (i+1) mod \sqrt{p}$; отримані блоки A'_{ij}, B'_{ij} перемножуються і додаються до блоку C_{ij}, блоки B'_{ij} кожної підзадачі пересилаються підзадачам зверху в стовпцях сітки по колу.

Далі приведено опис алгоритму, що відомий, як алгоритм DNS – за іменами його розробників Dekel, Nassimi і Sahni [3]. Припустимо, що $p = m^3$ процесорів доступні для множення двох матриць розмірності m^2. Ці процесори розташовані в m^3 тривимірному кубі. Процесори позначені відповідно до їх розташування в масиві. Вертикальний стовпець процесорів P_{ij*} обчислює скалярний добуток рядка A_{i*} і стовпця B_{*j}. Таким чином, рядки A і стовбці B повинні бути переміщені відповідним

чином, щоб кожна вертикальна колонка процесорів P_{ijk} мала рядки A_{ik} і стовпці B_{kj}.

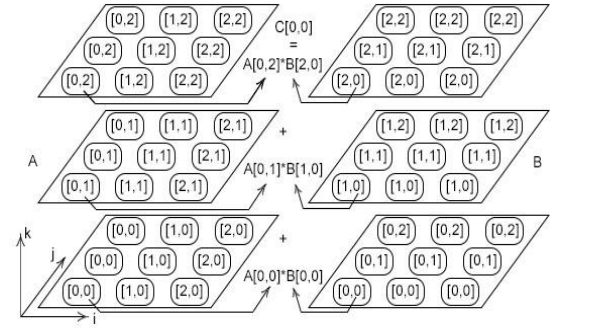
На рисунку 1 показано покрокове виконання алгоритму DNS. Спочатку (рис. 1(а)) матриці розподілені між m^2 процесорами площини шару $k=0$ в основі масиву тривимірного куба. Кожен стовпець A переміщається на той же рядок площини з номером $k=j$ (рис. 1(б)). Далі кожен елемент копіюється m разів у своїй відповідній площині по колонці j за принципом «один - всім» (рис. 2(в)). Для матриці B кроки алгоритму схожі, але ролі індексів i та j змінюються.



(а) Ініціалізація матриць (б) Після пересилки $A_{i,j}$

Рисунок 1 - Кроки алгоритму DNS

Після того, як кожен вертикальний стовпець процесорів P_{ijk} містить B_{kj} та A_{ik}, відбувається множення $A_{ik} * B_{kj}$ на кожному процесорі. Далі кожен елемент матриці C_{ij} обчислюється шляхом додавання отриманих після множення складових по осі k за принципом «всі - одному». На рисунку 2 продемонстровано кроки алгоритму для обчислення C_{00}.



(в) Передача A_{ij} по j (г) Передача B_{ij} по i

Рисунок 2 - Обчислення елементу C_{00}

Всі ці операції виконуються в рамках груп по m процесорів. Тому паралельний час виконання множення двох $m \times m$ матриць на m^3 процесорах має складність $\Theta(\log_2 m)$. З попереднього опису, алгоритм DNS використовує набагато менше циклів для обчислення матричного добутку. Отже, його загальні накладні витрати будуть значно меншими, ніж алгоритмів Кеннона та Фокса. Тому зроблено модифікацію алгоритму Кеннона для розташування m^3 процесорів в тривимірному кубі. Опишемо кроки алгоритму Cannon Modify (CM). Спочатку дані матриць A і B розташовуються на шарі $k=0$. На нульовому шарі циклічно зрушуємо рядки матриці A вліво і стовбці матриці B вгору. Копіюємо дані нульового шару на верхні рівні, але при цьому для матриці A індекс i обчислюється таким чином: $(i-k+q) \bmod q$, де k – шар, на який копіюємо, q – кількість процесорів в ряду ($q = \sqrt[3]{p}$ всіх процесорів). Після цього кроку на кожному шарі буде знаходитися нульовий шар, вже зрушений на $1,2,\ldots k$ вліво. Для матриці B аналогічні дії для індексу j: $(j-k+q) \bmod q$. На кожному шарі на кожному процесорі перемножуємо елементи матриць A і B, які містяться в ньому. Копіюємо отримані добутки з процесорів усіх шарів по осі k (по стовпцях) до відповідного процесору шару, де обчислюємо суму C_{ij} з накопичених добутків. Таким чином, всі добутки виконуються одночасно на різних шарах, тобто ітерація всього одна. Паралельний час виконання множення з використанням алгоритму CM має складність $\Theta(\log_2 m)$. Визначимо обчислювальну складність наведених алгоритмів для топологій 2D-тор та гіперкуб.

Час послідовного алгоритму для всіх алгоритмів: $T_1 = 2t_{op}m^3$. Висхідні дані для топології 2D-тор: матриця розміром m^2, кількість блоків $q^2 = p$, розмірність блоків $k = m/q$.

Функції загальних накладних витрат алгоритмів для топології 2D-тор:

а) алгоритм Кеннона:

$$T_o^C = \sqrt{p}m^2 t_{op} + 4(\sqrt{p}-1)m^2 t_w + 4p(\sqrt{p}-1)t_s;$$

б) алгоритм Фокса:

$$T_o^F = \sqrt{p}m^2 t_{op} + (\sqrt{p} + p/2 - 1)m^2 t_w + p(2\sqrt{p}-1)t_s$$

Функції загальних накладних витрат алгоритмів для топології гіперкуб:

1) алгоритм Кеннона:

$$T_o^C = \left(\sqrt[3]{p^2}m^2 + 2(\sqrt[3]{p}-1)m^3\right)t_{op} +$$

$$+\left(4\sqrt[3]{p^2}\log_2 p - 4\sqrt[3]{p}\log_2 pm^2\right)t_w + 4p(\sqrt[3]{p}-1)t_s;$$

2) алгоритм Фокса:

$$T_o^F = \left(2(\sqrt[3]{p}-1)m^3 + \sqrt[3]{p^2}m^2\right)t_{op} +$$

$$+ log_2\, p\left(\frac{2}{\sqrt[3]{p}} - \frac{1}{\sqrt[3]{p^2}}\right)m^2 t_w + p\sqrt[3]{p}(log_2\, p - 1)t_s;$$

3) модифікація алгоритму Кеннона:

$$T_o^{CM} = m^3 t_{op} + 2p\left(\frac{m^2}{\sqrt[3]{p^2}} + \frac{m^2}{\sqrt[3]{p^2}}t_w\right)(2log_2\, p - 1)t_w +$$

$$+ (3p\, log_2\, p + 2p)t_s.$$

4) алгоритм DNS:

$$T_o^{DNS} = m^3 t_{op} + 2p\left(\frac{m^2}{\sqrt[3]{p^2}} + \frac{m^2}{\sqrt[3]{p^2}}log_2\, p\right)t_w +$$

$$+ (3p\, log_2\, p + 2p)t_s.$$

Для порівняння якості реалізації двох паралельних алгоритмів на паралельній архітектурі та визначення пріоритетних областей застосування цих алгоритмів, також використовують засади ізоефективного аналізу. Спочатку обчислюють для кожного з алгоритмів загальні накладні витрати. Потім із рівності: $T_o\_Alg1 = T_o\_Alg2$ виводиться співвідношення, що зв'язує розмір задачі із числом процесорів паралельної системи: $m = f_{Equal-T_o}(p)$. Таким чином, отримується поріг або таке значення розміру задачі, при якому накладні витрати двох алгоритмів будуть однакові для заданого числа процесорів. Якщо $m > f_{Equal-T_o}(p)$, то перший алгоритм має кращі характеристики виконання на паралельній системі. У протилежному випадку, другий алгоритм має переваги.

Зауваження: визначення діапазонів для m та p, що дають підставу для вибору одного з декількох паралельних алгоритмів, який працює краще ніж інші, також залежить від багатьох чинників (параметрів комп'ютеру та задачі). Але перевага апарату ізоефективного аналізу в тому і полягає, що він дає єдиний аналітичний вираз, який зв'язує всі динамічні параметри і дозволяє систематизувати проведення багаторазових експериментів та досліджень. Наприклад, із рівності загальних накладних витрат алгоритмів Кеннона та Фоксу $T_o\_Cannon = T_o\_Fox$, отримується аналітичний вираз функції ізоефективності для дослідження областей застосування цих алгоритмів: $m = f_{Equal-Cannon\_to\_Fox}$. Функції ізоефективності алгоритмів Кеннона і Фокса (рис. 3):

1) топологія 2D-тор:

$$m = \sqrt{\frac{t_s p(2\sqrt{p}-3)}{t_w(\frac{p}{2}+3-3\sqrt{p})}};$$

2) гіперкуб:

$$m = \sqrt{\frac{t_s p(\sqrt[3]{p}(log_2\sqrt[3]{p}-3)+3)}{t_w(2\sqrt[3]{p^2}\, log_2\sqrt[3]{p} - 3\sqrt[3]{p}\, log_2\sqrt[3]{p})}}.$$

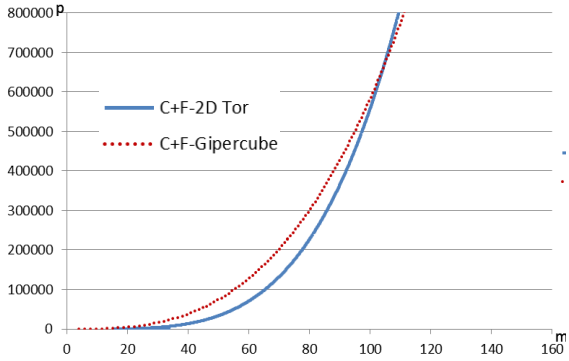


Рисунок 3 - Графіки функцій ізоефективності алгоритмів Кеннона і Фокса

Як бачимо з графіку 3, для матриць малої розмірності топологія 2D-тор потребує меншу кількість процесорів для забезпечення постійної ефективності. Це виконується для матриць розміру менше 100. При зростанні m найбільшими перевагами буде володіти топологія гіперкуб. Для топології 2D-тор вище графіка функції маємо область таких значень параметрів m та p, де ефективнішим є алгоритм Фокса (його накладні витраті менші за витрати алгоритму Кеннона), також це вірно і для топології гіперкуб.

Функція ізоефективності для блокового алгоритму Фокса та рекурсивного поліалгоритму Штрассена, що детально описано в [11-12], також обчислюється шляхом прирівнювання їх загальних накладних витрат $m = f_{Fox\_to\_Strassen}$:

$$m = \sqrt{\frac{(2\sqrt{p}-3)pt_s}{(\sqrt{p}-(5\left(\frac{7}{4}\right)^d - 4)(\sqrt{p}-1))t_{op} + (\frac{p}{2}-3\sqrt{3}+3)t_w}},$$

змінна d – глибина рекурсії, яка дуже сильно впливає на вигляд функції накладних витрат, а, отже, і функції ізоефективності. На рисунку 4 приведено графіки функцій ізоефективності Фокса і Штрассена разом з функціями алгоритмів Кеннона і Фокса при значенні глибини рекурсії =3. Вище лінії відповідного графіку краще алгоритм Фокса, нижче – поліалгоритм Штрассена. Як бачимо, функція Фокса – Штрассена потребує меншу кількість процесорів для обчислення добутку матриць тієї ж розмірності, що і функції Кеннона – Фокса.

Для порівняння усіх алгоритмів побудовано графіки їх функцій загальних витрат для різної кількості процесорів (рис. 5).

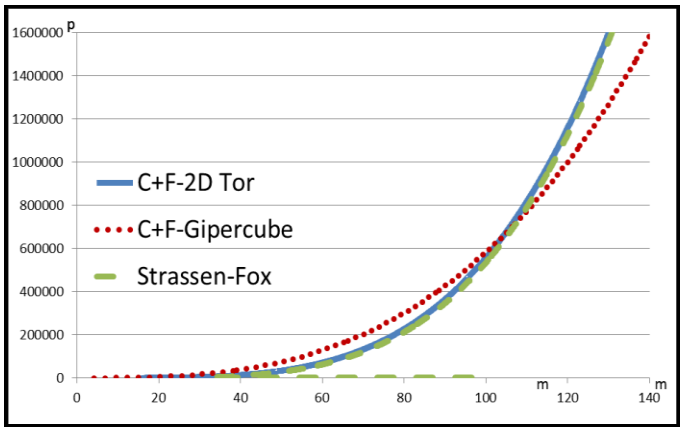


Рисунок 4 - Функції ізоефективності алгоритмів Кеннона, Фокса, Штрассена при $d = 3$

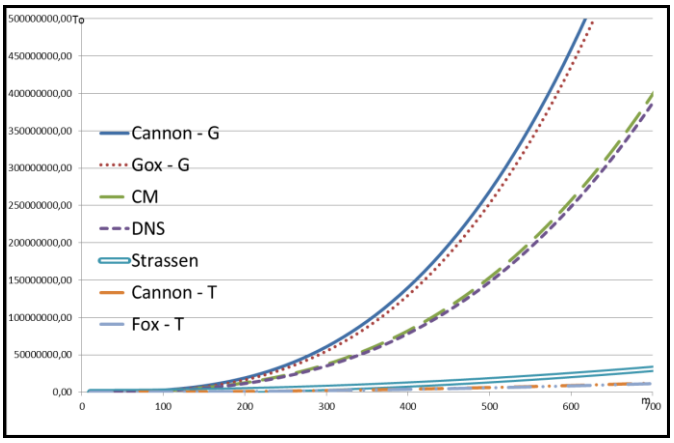


Рисунок 5 - Функції загальних накладних витрат паралельних алгоритмів для топологій 2D-тор (T) та гіперкуб (G) при $p=2^{15}=32768$

Висновки

В даній статті приведено результати досліджень, що присвячені оцінці ефективності та масштабованості паралельних алгоритмів на паралельній архітектурі. Проведено аналітичний огляд метрик масштабованості на основі таких динамічних характеристик, як прискорення, загальні накладні витрати та ефективність. Приведено приклади застосування найбільш вживаної характеристики масштабованості, а саме функції ізоефективності, для реалізації паралельних алгоритмів матричного добутку таких, як Кеннона, Фокса, DNS, Штрассена тощо. Розроблено модифікацію алгоритму Кеннона для тривимірної замкнутої сітки, загальні накладні витрати якого набагато менші, ніж висхідного алгоритму. Отримані експериментальні результати та виявлені переваги та недоліки відображення цих алгоритмів на топології гіперкуб та 2D-тор.

Перспективним напрямом роботи можна вважати застосування теорії ізоефективного аналізу до дослідження масштабованості паралельних алгоритмів для інтегрування систем алгебраїчних та звичайних диференційних рівнянь.

Список використаної літератури

1. Гергель В.П. Теория и практика параллельных вычислений / В.П. Гергель. – М.: Бином. Лаборатория знаний, 2007. – 423с.
2. Grama A. Isoefficiency: Measuring the scalability of parallel algorithms and architectures / A. Grama, A. Gupta, V. Kumar // IEEE Parallel and Distributed technology. – 2003. – №1 (3). – P. 12-21.
3. Kumar V. Analysing scalability of parallel algorithms and architectures / A. Gupta, V. Kumar // Technical Report TR 91-18. Computer science Department, University of Minnesota. – 1993. – P. 1-21.

4. Kumar V. Parallel depth-first search, part II: Analysis / V. Kumar, V.N. Rao // International journal of Parallel Programming. – 1987. – №16 (6). – P. 501-519.
5. Gustafson J. Development of parallel methods for a 1024-processor hypercube / J. Gustafson, G. Montry, R. Benner // SIAM Journal on Scientific and Statistical Computing. – 1988. – №9 (4). – P. 609-638.
6. Karp A. Measuring parallel processor performance / A. Karp, H. Flatt // Communications of the ACM. – 1990. – №33 (5). – P. 539-543.
7. Zorbas J.R. Measuring the scalability of parallel computer systems / J.R. Zorbas, D.J. Reble, R.E. VanKooten // In Supercomputing'89 Proceedings. – 1989. – P.832-841.
8. Eager D.L. Speedup versus efficiency in parallel systems / D.L. Eager, J. Zahorjan, E.D. Lazowska // IEEE Transactions on Computers. – 1989. – 38 (3). – P. 408-423.
9. Fredric A. Van-Catledge. Towards a general model for evaluating the relative performance of computer systems. International Journal of Supercomputer Applications. – 1989. – №3 (2). – P. 100-108.
10. Sun X-H. Scalability of parallel algorithm-machine combinations / Xian-He Sun, Diane Thiede Rover // Technical Report IS-5057, Ames Laboratory, Iowa State University, Ames, IA. – 1991.
11. Чорна А.О. Аналіз масштабованості паралельних алгоритмів матричного добутку на різноманітних топологіях / А.О. Чорна, І.А. Назарова // 5 Всеукраїнська науково-технічна конференція студентів, аспірантів та молодих вчених ІУС та КМ-2014. – Донецьк: ДонНТУ, 2014. – Т.1. – С. 249-254.
12. Назарова І.А. Підвищення ефективності паралельного розв'язання лінійної задачі Коші на основі методу рекурсивного множення матриць / І.А. Назарова // Научно-теоретический журнал ИПИИ НАН Украины «Искусственный интеллект». – 2008. – №3. – С. 706-714.

Надійшла до редакції 30.09.2015

И.А. НАЗАРОВА[1], А.О. ЧЕРНАЯ[2]
[1] Донецкий национальный технический университет
[2] Днепропетровский национальный университет

ИССЛЕДОВАНИЕ МАСШТАБИРУЕМОСТИ ПАРАЛЛЕЛЬНЫХ СИСТЕМ НА ОСНОВЕ ФУНКЦИИ ИЗОЭФФЕКТИВНОСТИ

В статье представлены результаты исследования масштабируемости параллельных систем с использованием функции изоэффективности. Основы изоэффективного анализа применены для параллельных методов матричного умножения таких, как алгоритмы Кеннона, Фокса, DNS и Штрассена. Разработан модифицированный алгоритм Кеннона для 3-мерного тора. Определена степень масштабируемости алгоритмов и получены приоритетные области их использования в зависимости от размера задачи и числа процессоров для топологий 2D-тор и гиперкуб.

Ключевые слова: параллельные алгоритмы/архитектуры, масштабируемость, изоэффективный анализ, ускорение, эффективность, общие накладные расходы, матричное умножение.

I.A. NAZAROVA[1], A.O. CHORNA[2]
[1] Donetsk National Technical University
[2] Dnipropetrovsk National University

RESEARCH OF PARALLEL SYSTEM SCALABILITY BASED ON THE ISOEFFICIENCY FUNCTION

In this article the results of research devoted to evaluating the scalability of parallel algorithms on parallel architecture are given. Analytical review of the metrics of scalability, which are based on dynamic characteristics of algorithms like speedup, general overhead and efficiency, is conducted.

Keywords: parallel algorithms/architecture, scalability, isoefficiency analysis, speedup, effectiveness, general overhead, matrix multiplication.