4. Mérindol P. Improving Load Balancing with Multipath Routing / Mérindol P., Pansiot J., Cateloin S. // Proc. of the 17th International Conference on Computer Communications and Networks, IEEE ICCCN 2008. – 2008. – P. 54–61.
5. Лемешко А.В. Усовершенствование потоковой модели многопутевой маршрутизации на основе балансировки нагрузки / А. В. Лемешко, Т. В. Вавенко // Проблеми телекомунікацій. – 2012. – № 1 (6). – С. 12–29 [Электронный ресурс]. – Режим доступа: http://pt.journal.kh.ua/2012/1/1/121_lemeshko_multipath.pdf
6. Mantar H.A. A scalable model for interbandwidth broker resource reservation and provisioning / H.A. Mantar, J. Hwang, I.T.Okumus [and other] // IEEE Journal on Selected Areas in Communications. – 2004. – Vol. 22, Issue 10. – P. 2019 – 2034.
7. Aukia P. RATES: a server for MPLS traffic engineering / P. Aukia, M. Kodialam, P.V.N. Koppol [and other] // IEEE Network. – 2000. – Vol. 14, Issue 2. – P. 34 – 41.

Розпаралелення обчислень для швидкого вейвлет-перетворення

Кулик Ярослав Анатолійович
аспірант Вінницького національного технічного університету

Найбільша проблема, яка виникає для швидкого вейвлет-перетворення – порівняно великі часові затрати, хоча алгоритмічна складність обчислення пропорційна розміру вхідних даних. Підвищити швидкість виконання обчислень можна, або використовуючи більш ефективних комп'ютерів, або зменшенням алгоритмічної складності обчислень. Час обчислення у векторному процесорі(при F – кількість операцій з плаваючою крапкою, T – повний час виконання, t_v - час обчислення для одного числа з плаваючою крапкою, t_s - час запуску) повинен мати наступний характер [1]

$$T=t_s+t_vF \tag{1}$$

Продуктивність виражена в кількості операцій з плаваючою крапкою, які можуть бути за 1 секунду [2]

$$R_\infty = \lim_{F\to\infty}\frac{F}{T} = \lim_{F\to\infty}\frac{F}{t_s+t_vT} = \frac{1}{t_v} \tag{2}$$

Тривалість виконання частинного вейвлет-перетворення має також лінійний характер, як в моделі.

$$T_{FWT}(N) = t_s\log_2 N + t_vF_{FWT}(N) \tag{3}$$

Двовимірне швидке вейвлет-перетворення визначали як

$$\tilde{X} = X^{\lambda_M}X(W^{\lambda_N})^T \tag{4}$$

Дане перетворення дозволяє використати метод розділення-переміщення. Ігноруючи час для операції перестановки, проста модель тривалості виконання для двовимірного швидкого вейвлет-перетворення виглядає

$$T_{FWT2}(M,N) = T_{MFWT}(M,N) + T_{MFWT}(N,M), \tag{5}$$

T_{MFWT} визначається (5). Рисунок 1 показує, що прогнозування виконується досить точно. Асимптотична продуктивність з використанням (4) та (5) $(R_\infty)_{FWT2}$ може бути визначена.

$$R_{FWT2} = \frac{4DMN\left(2-\frac{2}{2^{\lambda_M}}-\frac{2}{2^{\lambda_N}}\right)}{2N\left(1-\frac{2}{2^{\lambda_N}}\right)(2DMt_v+t_s)+2M\left(1-\frac{2}{2^{\lambda_M}}\right)(t_s+2DNt_v)} \tag{6}$$

Прискаючи, що $\lambda_M = \log_2 M$ і $\lambda_M = \log_2 N$, а числа M і N – достатньо великі, отримуємо оцінку

$$R_{FWT2} \approx \frac{1}{\frac{M+N}{4DNM}t_s+t_v}, \tag{7}$$

$$(R_\infty)_{FWT2}(N) \approx \frac{1}{\frac{t_s}{4DN}+t_v}, \tag{8}$$

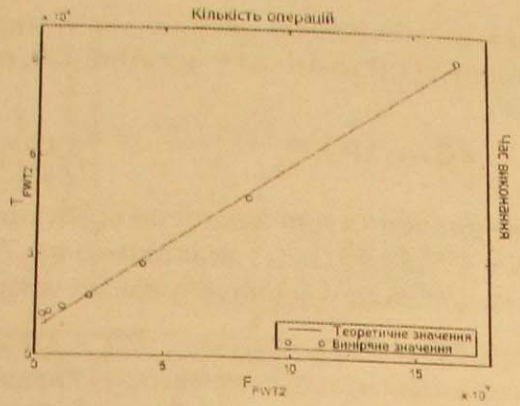

Рисунок 1 – Виміряний час виконання для двовимірного швидкого вейвлет-перетворення та теоретичний час виконання на основі моделі (6).

Розглянемо величину затрат ресурсів на зв'язок між процесорами, потрібної для паралельного одновимірного швидкого вейвлет-перетворення. Виконання обчислень та розподіл вектора даних показаний на рисунку 2. Обчислення на процесорі p потребує D - 2 елементів від процесора $p + 1$. На рисунку 2 $D = 6$ і $N/(P2^l) = 8$. Ширина ліній D указують на розмір фільтра підвекторів, оскільки вони застосовуються для різних значень n.

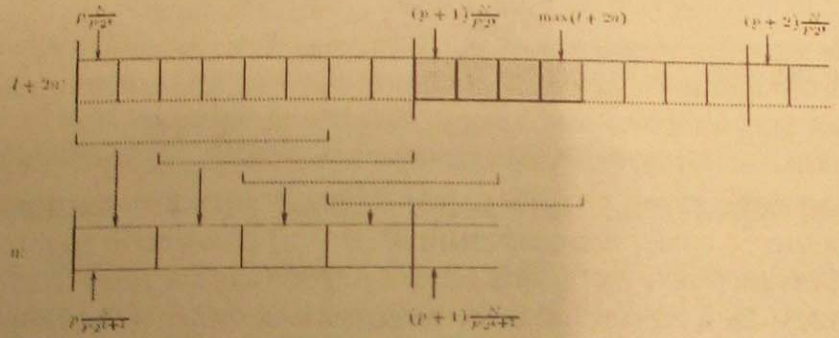

Рисунок 2 – Розподіл вектора даних між процесора для кращого збалансування навантаження

Величина необхідних зв'язків між процесорами за даних умов складає $M(D - 2)$ замість $(D - 2)$, колонка X розподіляється між процесорами поблоково, і перетворення X включає рекурсивну формулу XW_N^T.

Розглянемо теоретично оптимальні досяжні параметри продуктивності для багатократного одновимірного швидкого вейвлет-перетворення.

З урахуванням ()вираз для операцій з плаваючою крапкою

$$F_{MFWT}(N) = 4MDN\left(1 - \frac{1}{2^{\lambda}N}\right) \tag{9}$$

Оскільки є λ кроків для вейвлет-перетворення, то проста модель протягом повного часу для обміну даними між процесорами, складає

$$C_{MFWT} = \lambda(t_l + M(D-2)t_d) \tag{10}$$

Об'єднуючи вираз для обчислення часу і часу для обміну даними, отримуємо модель, що описує повну тривалість виконання на P процесорах ($P > 1$)

$$T_{MFWT}^{P}(N) = \frac{T_{MFWT}^{0}(N)}{p} + C_{MFWT} \tag{11}$$

і продуктивність для паралельного алгоритму

$$R_{MFWT}^{P}(N) = \frac{F_{MFWT}(N)}{T_{MFWT}^{P}(N)} \tag{12}$$

Об'єднання виразів для продуктивності (10), (11) і (12) дає формулу для збільшення швидкодії

$$S_{MFWT}^{P}(N) = \frac{T_{MFWT}^{0}(N)}{T_{MFWT}^{P}(N)} = \frac{P}{1+P\frac{C_{MFWT}(N)}{T_{MFWT}^{0}(N)}} \tag{13}$$

Ефективність паралельних обчислень визначимо як значення збільшення швидкодії на один процесор. Як видно з (13) цього для постійної N, ефективність падає при збільшенні числа процесорів P.

$$E^P_{MFWT}(N) = \frac{S^P_{MFWT}(N)}{P} = \frac{1}{1+P\frac{C_{MFWT}(N)}{T^0_{MFWT}(N)}} \quad (14)$$

Нехай N має постійне значення для задачі на один процесор. Загальний об'єм обчислювальної задачі стає $N = PN_1$ і з (9) і (11) знаходимо що $T^0_{MFWT}(PN_1) = PT^0_{MFWT}(N_1)$. Це означає, що ефективність для задачі масштабування набуває форми

$$E^P_{MFWT}(PN_1) = \frac{1}{1+\frac{PC_{MFWT}}{PT^0_{MFWT}(N_1)}} = \frac{1}{1+\frac{C_{MFWT}}{T^0_{MFWT}(N_1)}} \quad (15)$$

Оскільки $E^P_{MFWT}(PN_1)$ незалежна від P, то ефективність при масштабуванні є постійною величиною. Тому багаторазовий одновимірне швидке велет-перетворення є повністю масштабованим, і використання паралельних обчислень для цієї задачі є ефективним.

Найпряміший спосіб розділити роботу, потрібну для двовимірного швидкого вейвлет-перетворення серед багатьох процесорів є розпаралелення серед розмірності по X так, що послідовність одного рядка перетворення виконуються незалежно на кожному процесорі. Затемнена ділянка показує переміщення блоку даних від процесора від $p = 1$ до процесора $p = 0$.

$$T^P_{RFWT}(N) = \frac{T^0_{FWT_2}(N)}{P} + C_{RFWT}, \quad (16)$$

і теоретична швидкодія при $N = PN_1$

$$S^P_{RFWT}(PN_1) = \frac{P}{1+\frac{C_{RFWT}}{T^0_{FWT_2}(N_1)}} \quad (17)$$

Зв'язок між процесорами потрібен, для багатогократного одновимірного швидкого вейвлет-перетворення, а саме, передавання $M(D-2)$ елементів між найближчими сусідами, при цьому більша частина блоків даних перебуває на тому ж самому процесорі при виконанні обчислень. Цей метод назвемо швидким вейвлет-перетворенням з ефективним використанням зв'язків.

Рисунок 3 – Швидке вейвлет-перетворення з ефективним обміном даних Затемнений блок даних залишається в процесорі

Модель продуктивності для даного методу - пряме розширення багатогократного швидкого вейвлет-перетворення, тому частка ресурсів, потрібних для обміну даними така сама, завдяки чому отримуємо теоретичне збільшення швидкодії.

$$S^P_{CFWT}(PN_1) = \frac{P}{1+\frac{C_{MFWT}}{T^0_{FWT_2}(N_1)}} \quad (18)$$

Порівняємо теоретичну продуктивність алгоритмів швидкого вейвлет-перетворення при звичайному переміщенні (17) та переміщенні з ефективним обміном даних (18) щодо їх відповідних залежностей від кількості процесорів P і розміру вектора вхідних даних N та затрат на зв'язок між процесорами. Порівняння швидкодії для швидкого вейвлет-перетворення при звичайному переміщенні блоків даних, переміщенні з ефективним обміном даних та теоретичної межі швидкодії показано на рисунку 4. З рисунка 3 видно характер залежності від кількості процесорів – для переміщення з ефективним обміном даних залежність лінійна

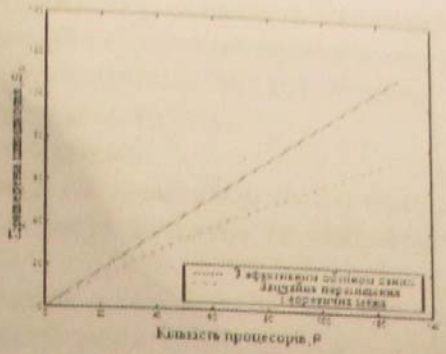

Рисунок 4 – Порівняння залежності швидкодії від кількості процесорів для різних алгоритмів.

Для випадку швидкого вейвлет-перетворенням з ефективним використанням зв'язків співвідношення $\frac{c_{MFWT}}{T^0_{FWT_2}(N_1)}$ лінійно залежить від P, а оскільки величина P наперед визначена, дане співвідношення можна вважати постійною величиною. Для випадку швидкого вейвлет-перетворенням зі звичайним переміщенням співвідношення $\frac{c_{MFWT}}{T^0_{FWT_2}(N_1)}$ у () складає залежність від $P - O(P)$, як показано на рисунку 4.

$$\frac{c_{MFWT}}{T^0_{FWT_2}(N_1)} \approx \frac{(P-1)t_l + \frac{P-1}{P^2} M N_1 t_d}{8DMN_1} = O(P) \tag{19}$$

Це означає, що ефективність для швидкого вейвлет-перетворення зі звичайним переміщенням погіршуватиметься при зростанні P, тоді як для швидкого вейвлет-перетворення з ефективним використанням зв'язків ефективність буде залишатись постійною. Якщо величина P стала і розмір вектора даних N_1 зростає, то співвідношення $\frac{c_{MFWT}}{T^0_{FWT_2}(N_1)}$ прямує до нуля, що означає, що приведена ефективність буде наближатись до теоретично можливого ідеального значення 1. Для швидкого вейвлет-перетворення з ефективним використанням зв'язків відповідне співвідношення наближається до позитивного постійного значення при зростанні $N_1 \to \infty$.

$$\frac{c_{MFWT}}{T^0_{FWT_2}(N_1)} \to \frac{(P-1)t_d}{8DP^2}, \tag{20}$$

Це означає, що приведена ефективність для швидкого вейвлет-перетворення зі звичайним переміщенням буде константа, менша за 1, при цьому не важливо, наскільки великий розмір обчислювальної задачі та розмір вектора вхідних даних. Асимптотична приведена ефективність двох алгоритмів показана в таблиці 1

	$P \to \infty$	$N_1 \to \infty$
Швидке вейвлет-перетворення з переміщенням	$\frac{1}{1+O(P)}$	$\frac{1}{1+\frac{(P-1)t_d}{8DP^2}}$
Швидке вейвлет-перетворення з ефективним використанням зв'язків	$\frac{1}{1+\frac{c_{MFWT}}{T^0_{FWT_2}(N_1)}}$	1

Таблиця 1 - Асимптотичне значення приведеної ефективності

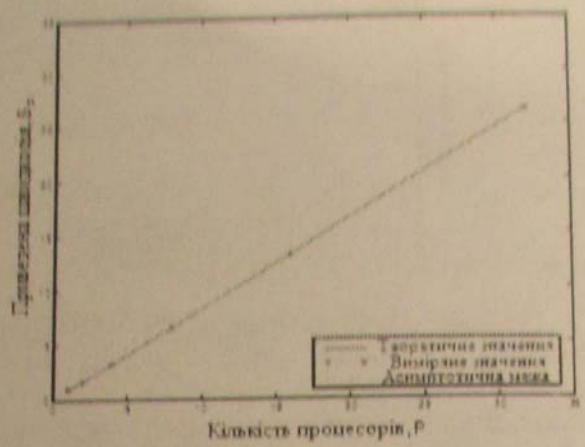

Рисунок 5 – приведена швидкодія для швидке вейвлет-перетворення з ефективним використанням зв'язків (IBM SP2).

Приведена продуктивність залежить від числа процесорів, і добре узгоджується з прогнозованою швидкодією, як показано на рисунку 5. Ці графіки показують, що теоретична модель продуктивності виконує реалістичне прогнозування фактичні параметри.

Література:

1. Воеводин В. В., Воеводин Вл. В. Параллельные вычисления. — СПб: БХВ-Петербург, 2002. — 608 с.
2. Гергель В.П., Фурсов В.А. Лекции по параллельным вычислениям. Учебное пособие. - Самара: Изд-во Самар. гос. аэрокосм. ун-та, 2009. – 164 с.

Оптимізація запитів в реляційних системах управління базами даних

Уколов Андрій Сергійович
студент Національного технічного університету «Київський політехнічний інститут»

Оптимізація запитів в реляційних системах управління базами даних (СУБД) являє собою як проблему, так і можливість поліпшення ефективності функціонування інформаційних систем. Як проблема, оптимізація запитів виражається в необхідності проведення оптимізації задля досягнення необхідного рівня продуктивності. В свою чергу, як можливість, оптимізація запитів дозволяє досягнути більш високої продуктивності інформаційних систем при роботі з СУБД: підвищення швидкості обробки запитів, зменшення часу відклику, тощо. В загальному, призначення оптимізації запитів полягає у виборі ефективної стратегії для обчислення тих чи інших реляційних виразів.

Оптимізація запитів в реляційних СУБД – спосіб обробки запитів, при якому виконується перетворення самого запиту в найбільш оптимальну форму. Такі перетворення можуть виконуватись вручну користувачем (програмістом, адміністратором СУБД), але, зазвичай, вони виконуються спеціальним компонентом СУБД – оптимізатором запитів. Оптимізатори запитів в сучасних СУБД виконують перетворення запиту на семантичному рівні в більш зручний для машинних маніпуляцій та знаходять найбільш оптимальний шлях виконання такої форми, що забезпечить максимальну продуктивність. Зазвичай існує не один спосіб виконання запиту, які дають однаковий результат. Завданням оптимізатора є вибір найкращого з таких способів. [1, с. 215]

Суттєвою перевагою автоматичної оптимізації є відсутність необхідності проведення трансформації запиту користувачем в оптимальну форму. Крім того, цілком ймовірно, що оптимізатор запитів сформує запит значно краще, ніж сам користувач. [2, с. 682]