УДК 519.681.5

М.С. ЯДЖАК
Інститут прикладних проблем механіки і математики ім. Я.С. Підстригача НАН України; Львівський національний університет імені Івана Франка, Львів, Україна, e-mail: *yadzhak_ms@ukr.net*.

ПАРАЛЕЛЬНІ АЛГОРИТМИ ЦИФРОВОЇ ФІЛЬТРАЦІЇ ДАНИХ

Анотація. Запропоновано паралельні алгоритми розв'язання задач цифрової фільтрації різної розмірності на сучасних обчислювальних засобах універсального призначення. Одержано теоретичні оцінки складності та прискорення, які підтверджують високу ефективність цих алгоритмів. Здійснено програмну реалізацію деяких із запропонованих паралельних алгоритмів на комп'ютерах із багатоядерним процесором і одержано реальні оцінки прискорення, які добре узгоджуються із теоретичними.

Ключові слова: цифрова фільтрація, паралельний алгоритм, прискорення обчислень, обмежений паралелізм, еквівалентність алгоритмів, обчислювальна система.

ВСТУП

Задачі цифрової фільтрації (ЗЦФ) зазвичай виникають під час попереднього оброблення сигналів, плоских та просторових зображень [1, 2], великих масивів експериментальних даних тощо. У багатьох випадках такі задачі потрібно розв'язувати в режимі реального часу. З цією метою було запропоновано квазісистолічний метод обчислень [3, 4], на підставі якого розроблено оптимальні за швидкодією та використанням пам'яті паралельноконвеєрні алгоритми (ПКА) розв'язання ЗЦФ будь-якої розмірності. Оптимальність доводили у відповідних класах алгоритмів, які є еквівалентними за інформаційним графом з точністю до виконання співвідношень асоціативності та комутативності для операції додавання. Розроблені ПКА були зорієнтовані на реалізацію на спеціалізованих обчислювальних системах — квазісистолічних структурах. Однак квазісистолічні структури як обчислювальні засоби не є доступними для широкого кола користувачів. Тому у разі, коли маса та розміри обчислювальної системи не є суттєвими обмеженнями, для реалізації алгоритмів розв'язання задачі фільтрації можна використати універсальні обчислювальні засоби зі спільною (багатоядерні комп'ютери) та паралельною (кластерні системи, системи зі структурно-процедурною організацією обчислень [5, 6]) пам'яттю. Для цього з використанням оптимальних за швидкодією алгоритмів фільтрації було розроблено алгоритми [7] з обмеженим паралелізмом (АОП) для розв'язання одно-, дво- та тривимірної ЗЦФ. Унаслідок проведених досліджень [8] було встановлено, що в окремих АОП збільшення кількості змінних, значення яких переобчислюється в кожній з паралельних гілок, не спричиняє збільшення кількості обмінів між цими гілками або між заздалегідь визначеними їхніми групами. Отже, для таких гілок або їхніх груп є можливість збільшення частки обчислювальних операцій порівняно з обмінними. При цьому наведено умови, за яких відбувається суміщення обчислювальних і обмінних операцій. У [9] показано можливість ефективної реалізації деяких АОП на системах зі структурно-процедурною організацією обчислень.

Ця робота присвячена розробленню та дослідженню паралельних алгоритмів цифрової фільтрації для обчислювальних систем універсального призначення, зокрема алгоритмів, які використовують синхронну та асинхронну схеми обчислень [10].

ФОРМУЛЮВАННЯ ПРОБЛЕМИ

Розглянемо одновимірну ЗЦФ, для якої загалом передбачається переобчислення значень змінних x_{i_1} $(i_1 = \overline{1, l_1})$ за формулою

$$x_{i_1} = \sum_{s_1=-m_1}^{m_1} x_{i_1+s_1} f_{s_1}, \qquad (1)$$

де вагові коефіцієнти f_{s_1} $(s_1 = \overline{-m_1, m_1})$, а також «замежові» значення x_{1-m_1}, $x_{2-m_1}, \dots, x_0$; $x_{l_1+1}, x_{l_1+2}, \dots, x_{l_1+m_1}$ є відомими константами. За умовою задачі здійснюється C переобчислень згладжування на N точках через рухоме вікно розміром M, де $N = l_1$, $M = 2m_1 + 1$. Зазвичай для більшості практичних задач виконуються нерівності $M << N$, $C << N$.

У двовимірному випадку формула (1) набуває вигляду

$$x_{i_1,i_2} = \sum_{s_1=-m_1}^{m_1} \sum_{s_2=-m_2}^{m_2} x_{i_1+s_1,i_2+s_2} f_{s_1,s_2}. \tag{2}$$

У разі переобчислення значень x_{i_1,i_2} $(i_1 = \overline{1, l_1};\ i_2 = \overline{1, l_2})$ згідно з (2) одержуємо $N = l_1 l_2$, $M = (2m_1 + 1)(2m_2 + 1)$.

Звичайний послідовний алгоритм розв'язання одновимірної ЗЦФ має вигляд

$$\begin{array}{l} FOR\ \ t = 1, C\ \ DO \\ \{\ \ FOR\ \ i_1 = 1, l_1\ \ DO \\ \{\ \ \ \ p_1 = 0 \\ \quad FOR\ \ s_1 = -m_1, m_1\ \ DO \\ \{\ \ p_1 = p_1 + x_{i_1+s_1} * f_{s_1}\ \ \} \\ \quad x_{i_1} = p_1\ \ \ \} \ \ \}. \end{array} \tag{3}$$

Згідно з алгоритмом (3) для переобчислення x_{i_1} на t-му кроці використовують значення $x_{i_1-m_1}, x_{i_1-m_1+1}, \dots, x_{i_1-1}$, які є вже переобчисленими на цьому ж кроці.

Послідовний алгоритм розв'язання двовимірної ЗЦФ має вигляд

$$\begin{array}{l} FOR\ \ t = 1, C\ \ DO \\ \{\ \ FOR\ \ i_1 = 1, l_1\ \ DO \\ \{\ \ FOR\ \ i_2 = 1, l_2\ \ DO \\ \{\ \ p_1 = 0 \\ \qquad FOR\ \ s_1 = -m_1, m_1\ \ DO \\ \{\ \ FOR\ \ s_2 = -m_2, m_2\ \ DO \\ \{\ \ p_1 = p_1 + x_{i_1+s_1,i_2+s_2} * f_{s_1,s_2}\ \ \}\ \ \} \\ \quad x_{i_1,i_2} = p_1\ \ \ \}\ \ \ \}\ \ \ \}. \end{array} \tag{4}$$

Характерною особливістю алгоритму (4) є те, що для переобчислення значення змінної x_{i_1,i_2} на t-му кроці використовуються значення

$$x_{i_1-m_1,i_2-m_2}, x_{i_1-m_1,i_2-m_2+1}, \dots, x_{i_1-m_1,i_2}, x_{i_1-m_1,i_2+1}, \dots,\ x_{i_1-m_1,i_2+m_2};$$

$$x_{i_1-m_1+1,i_2-m_2}, x_{i_1-m_1+1,i_2-m_2+1}, \dots, x_{i_1-m_1+1,i_2}, x_{i_1-m_1+1,i_2+1}, \dots, x_{i_1-m_1+1,i_2+m_2}; \dots$$

$$\dots; x_{i_1-1,i_2-m_2}, x_{i_1-1,i_2-m_2+1}, \dots, x_{i_1-1,i_2}, x_{i_1-1,i_2+1}, \dots, x_{i_1-1,i_2+m_2};$$

$$x_{i_1,i_2-m_2}, x_{i_1,i_2-m_2+1}, \dots, x_{i_1,i_2-1},$$

які також вже є переобчисленими на цьому ж кроці.

Зазначені вище особливості алгоритмів (3) та (4) дають змогу збільшувати швидкість процесу згладжування, тому їх потрібно повністю або частково використовувати під час паралельної організації обчислень.

Інший послідовний алгоритм розв'язання одновимірної ЗЦФ має вигляд

$$\begin{array}{l} FOR\ \ t = 1, C\ \ DO \\ \{\ \ FOR\ \ i_1 = 1, l_1\ \ DO \\ \{\ \ p = 0 \\ \quad FOR\ \ s_1 = -m_1, m_1\ \ DO \\ \{\ \ p = p + x_{i_1+s_1}^{t-1} * f_{s_1}\ \ \} \\ \quad x_{i_1}^{t} = p\ \ \ \} \ \ \}. \end{array} \tag{5}$$

Тут $x_{i_1}^0$, $x_{i_1}^t$ — відповідно початкове значення змінної x_{i_1} та значення цієї ж змінної, переобчислене на t-му кроці. Для виконання наведений алгоритм потребує час $T_1 = t_{op}(2m_1 + 1)Cl_1$, де t_{op} — час виконання подвійної операції $a + b * c$.

Для розв'язання двовимірної ЗЦФ розглянемо послідовний алгоритм

$$
\begin{array}{l}
\quad FOR\ \ t = 1, C\ \ DO \\
\quad \{\ \ FOR\ \ i_1 = 1, l_1\ \ DO \\
\{\ \ FOR\ \ i_2 = 1, l_2\ \ DO \\
\quad \{\ \ p = 0 \\
\quad FOR\ \ s_1 = -m_1, m_1\ \ DO \\
\{\ \ FOR\ \ s_2 = -m_2, m_2\ \ DO \\
\quad \{\ \ p = p + x^{t-1}_{i_1+s_1, i_2+s_2} * f_{s_1, s_2}\ \ \}\ \ \} \\
\quad x^t_{i_1, i_2} = p\ \ \}\ \ \}\ \ \}.
\end{array}
\tag{6}
$$

У наведеній конструкції x_{i_1, i_2}^0, x_{i_1, i_2}^t — відповідно початкове значення змінної x_{i_1, i_2} та значення цієї ж змінної, переобчислене на t-му кроці. Для виконання цей алгоритм потребує час $T_2 = t_{op}(2m_1 + 1)(2m_2 + 1)Cl_1l_2$.

В алгоритмах (5), (6) для переобчислення значень змінних на t-му кроці використовуються значення, переобчислені тільки на $(t-1)$-му кроці.

Вирішувана нами проблема полягає в розробленні та аналізі паралельних алгоритмів розв'язання одно- та двовимірної ЗЦФ на основі використання синхронної та асинхронної схем обчислень.

РОЗРОБЛЕННЯ ТА АНАЛІЗ ПАРАЛЕЛЬНИХ АЛГОРИТМІВ ФІЛЬТРАЦІЇ

Для одновимірної задачі фільтрації паралельний режим оброблення отримуємо унаслідок одночасного переобчислення значень змінних x_{i_1} $(i_1 = \overline{1, l_1})$ за алгоритмом

$$
\begin{array}{l}
\quad FOR\ \ t = 1, C\ \ DO \\
\{\ \ FOR\ \ i_1 = 1, l_1\ \ DO\ \ PAR \\
\{\ \ p_{i_1} = 0 \\
\quad FOR\ \ s_1 = -m_1, m_1\ \ DO \\
\{\ \ p_{i_1} = p_{i_1} + x_{i_1+s_1} * f_{s_1}\ \ \} \\
\quad x_{i_1} = p_{i_1}\ \ \}\ \ \}.
\end{array}
\tag{7}
$$

У разі, коли типом паралелізму *PAR* є *SIM* (simultaneous), наведений паралельний алгоритм буде реалізовувати синхронний метод [10, 11] обчислень. Із (7) випливатиме, що всі значення змінних для переобчислення на t-му кроці визначаються на $(t-1)$-му кроці, що зменшує швидкість процесу згладжування порівняно із (3). В окремих випадках цей недолік можна частково усунути, використовуючи в (7) замість *PAR* тип паралелізму *CONC* (concurrent). Тоді такий алгоритм буде реалізовувати асинхронний метод обчислень. У цьому разі кожне значення x_{i_1} переобчислюється незалежно від інших. При цьому як аргументи використовуються поточні значення

$$x_{i_1-m_1}, x_{i_1-m_1+1}, \ldots, x_{i_1}, x_{i_1+1}, \ldots, x_{i_1+m_1}.$$

Розглянемо подвійну операцію $a + b + c$. Якщо знехтувати часом на забезпечення синхронізації, то алгоритм (7) реалізовуватиме синхронний метод обчислень за час $T_s = t_{op}(2m_1 + 1)C$.

Використовуючи ідеї методу пірамід [11, 12] для розпаралелювання циклів, реалізацію синхронного методу обчислень можна здійснити за таким алгоритмом:

$$
\begin{array}{l}
\quad FOR\ \ k=1, l_1\ \ DO\ \ PAR \\
\{\ \ FOR\ \ t=1, C\ \ DO \\
\{\ \ FOR\ \ i_1=\max\{1, m_1(t-C)+k\}, \min\{l_1, m_1(C-t)+k\}\ \ DO \\
\{\ \ p=0 \\
\quad FOR\ \ s_1=-m_1, m_1\ \ DO \\
\quad \{\ \ p=p+x_{i_1+s_1}^{t-1} * f_{s_1}\ \ \} \\
\qquad x_{i_1}^{t}=p\ \ \ \}\ \ \}\ \ \}.
\end{array} \tag{8}
$$

У цьому разі *PAR* є типом паралелізму *AUTON* (autonomous) [11], тобто конструкція (8) задає паралельне виконання l_1 гілок, які не взаємодіють між собою, отже виконуються автономно. Час функціонування цього алгоритму обчислюється за формулою $T_p = t_{op}(2m_1+1)C(1+m_1(C-1))$.

Процес згладжування можна дещо покращити, якщо в (8) останні чотири рядки замінити фрагментом

$$
\begin{array}{l}
\{\ \ p_{i_1}=0 \\
\quad FOR\ \ s_1=-m_1, m_1\ \ DO \\
\{\ \ p_{i_1}=p_{i_1}+x_{i_1+s_1} * f_{s_1}\ \ \} \\
\quad x_{i_1}=p_{i_1}\ \ \ \}\ \ \}.
\end{array}
$$

Отримана унаслідок цього паралельна конструкція дає змогу в кожній гілці під час переобчислення значення деякої змінної на заданому кроці використовувати значення, уже переобчислені на цьому ж кроці.

Паралельний режим оброблення під час розв'язування двовимірної ЗЦФ, який пов'язаний із одночасним переобчисленням значень всіх змінних x_{i_1,i_2} ($i_1=\overline{1, l_1}$; $i_2=\overline{1, l_2}$), можна реалізувати, наприклад, за допомогою алгоритму

$$
\begin{array}{l}
FOR\ \ t=1, C\ \ DO \\
\{\ \ FOR\ \ ALL\ \ (i_1,i_2)\in\{(i_1,i_2): i_1=\overline{1, l_1};\ i_2=\overline{1, l_2}\}\ \ DO\ \ PAR \\
\quad \{\ \ p_{i_1,i_2}=0 \\
\quad\ \ FOR\ \ s_1=-m_1, m_1\ \ DO \\
\quad\ \{\ \ FOR\ \ s_2=-m_2, m_2\ \ DO \\
\quad\ \{\ \ p_{i_1,i_2}=p_{i_1,i_2}+x_{i_1+s_1,i_2+s_2} * f_{s_1,s_2}\ \ \}\ \ \} \\
\quad\ \ x_{i_1,i_2}=p_{i_1,i_2}\ \ \ \}\ \ \}.
\end{array} \tag{9}
$$

Аналогічно, як і в алгоритмі (7), у конструкції (9) тип паралелізму PAR може бути SIM або CONC. Тоді цей паралельний алгоритм реалізуватиме відповідно синхронний або асинхронний метод обчислень. У другому випадку кожне із значень x_{i_1,i_2} переобчислюватиметься незалежно від інших. При цьому як аргументи використовуються поточні значення $x_{i_1-m_1,i_2-m_2}$, $x_{i_1-m_1,i_2-m_2+1},\ldots,x_{i_1,i_2}$, $x_{i_1,i_2+1},\ldots,x_{i_1+m_1,i_2+m_2}$. Для реалізації синхронного методу обчислень за (9) потрібен час $T_{s_1}=t_{op}(2m_1+1)(2m_2+1)C$.

Використовуючи метод пірамід для розпаралелювання циклів, реалізацію синхронного методу обчислень у даному випадку можна здійснити з допомогою такого алгоритму:

$$
\begin{array}{l}
FOR\ \ ALL\ \ (k_1,k_2)\in\{(k_1,k_2): k_1=\overline{1,l_1};\ k_2=\overline{1,l_2}\}\ \ DO\ \ PAR\\
\quad\{\ \ FOR\ \ t=1,C\ \ DO\\
\qquad\{\ \ FOR\ \ i_1=\max\{1,(t-C)m_1+k_1\},\min\{l_1,(C-t)m_1+k_1\}\ \ DO\\
\qquad\{\ \ FOR\ \ i_2=\max\{1,(t-C)m_2+k_2\},\min\{l_2,(C-t)m_2+k_2\}\ \ DO\\
\qquad\quad\{\ \ p=0\\
\qquad\quad FOR\ \ s_1=-m_1,m_1\ \ DO\\
\qquad\quad\{\ \ FOR\ \ s_2=-m_2,m_2\ \ DO\\
\qquad\qquad\{\ \ p=p+x^{t-1}_{i_1+s_1,i_2+s_2}*f_{s_1,s_2}\ \ \}\\
\qquad\qquad x^{t}_{i_1,i_2}=p\ \ \}\ \ \}\ \ \}\ \ \}\ \ \}.
\end{array}
\tag{10}
$$

У цьому разі *PAR* є типом паралелізму *AUTON* і (10) задає паралельне виконання $l_1 l_2$ автономних гілок. Час виконання цього алгоритму обчислюється за формулою

$$T_{p_1}=t_{op}(2m_1+1)(2m_2+1)C(1+((2/3)m_1m_2(2C-1)+m_1+m_2)(C-1)).$$

Аналогічно, як і в одновимірному випадку, в (10) процес згладжування можна дещо покращити, якщо останні п'ять рядків замінити фрагментом

$$
\begin{array}{l}
\{\ \ p_{i_1,i_2}=0\\
\quad FOR\ \ s_1=-m_1,m_1\ \ DO\\
\{\ \ FOR\ \ s_2=-m_2,m_2\ \ DO\\
\{\ \ p_{i_1,i_2}=p_{i_1,i_2}+x_{i_1+s_1,i_2+s_2}*f_{s_1,s_2}\ \ \}\\
\quad x_{i_1,i_2}=p_{i_1,i_2}\ \ \}\ \ \}\ \ \}\ \ \}\ \ \}.
\end{array}
$$

Отримана унаслідок цього паралельна конструкція дає змогу в кожній гілці під час переобчислення значення деякої змінної на заданому кроці використовувати значення, що є вже переобчисленими на цьому ж кроці.

АЛГОРИТМИ З ОБМЕЖЕНИМ ПАРАЛЕЛІЗМОМ

Наведеними паралельними алгоритмами (7)–(10) можна виконати N паралельних гілок, що потребує N окремих обчислювальних пристроїв (процесорів). Однак у реальних обчислювальних системах, зокрема кластерах, кількість одночасно задіяних процесорів (ядер) є обмеженою та наперед визначеною. Тому для розв'язання задач фільтрації доцільніше розглядати АОП, в яких кількість паралельно виконуваних гілок P є меншою за N $(P<N)$ [7]. Для спрощення подальшого викладу вважатимемо, що N є кратним P.

На підставі викладеного розглянемо деякі АОП розв'язання одно- та двовимірної ЗЦФ. Зокрема, для розв'язання одновимірної задачі фільтрації розглянемо паралельний алгоритм з синхронізацією:

$$
\begin{array}{l}
\quad FOR\ \ t=1,C\ \ DO\\
\{\ \ FOR\ \ p=1,P\ \ DO\ \ SIM\\
\{\ \ FOR\ \ i_1=J_1(p),J_2(p),J_3\ \ DO\\
\{\ \ p_{i_1}=0\\
\quad FOR\ \ s_1=-m_1,m_1\ \ DO\\
\{\ \ p_{i_1}=p_{i_1}+x_{i_1+s_1}*f_{s_1}\ \ \}\ \ \}\ \ \}\\
\quad FOR\ \ i=1,l_1\ \ DO\\
\{\ \ x_i=p_i\ \ \}\ \ \}.
\end{array}
\tag{11}
$$

У наведеному алгоритмі $J_1(p)=p,\ \ J_2(p)=p+(l_1/P-1)(m_1+1),\ \ J_3=m_1+1.$ Цей паралельний алгоритм задає виконання P гілок, в кожній з яких переоб-

числюють значення l_1 / P змінних. До того ж $P = m_1 + 1$. Враховуючи припущення, зроблені щодо оцінювання часу виконання алгоритмів у попередньому розділі, отримуємо, що час виконання алгоритму (11) обчислюється за формулою $T_{so} = t_{op}(2m_1 + 1)l_1 C / P$.

Розглянемо конструкцію, яка у певному сенсі реалізує загалом асинхронну схему обчислень:

$$
\begin{array}{l}
\quad FOR\ \ t = 1, C\ \ DO \\
\{\ \ FOR\ \ p = 1, P\ \ DO\ \ CONC \\
\{\ \ FOR\ \ i_1 = J_1(p), J_2(p), J_3\ \ DO \\
\{\ \ p_{i_1} = 0 \\
\quad FOR\ \ s_1 = -m_1, m_1\ \ DO \\
\{\ \ p_{i_1} = p_{i_1} + x_{i_1 + s_1} * f_{s_1}\ \ \} \\
\quad x_{i_1} = p_{i_1}\ \ \ \} \ \ \ \} \ \ \ \}.
\end{array}
\tag{12}
$$

Ще один паралельний алгоритм має вигляд

$$
\begin{array}{l}
\quad FOR\ \ t = 1, C\ \ DO \\
\{\ \ FOR\ \ p = 1, P\ \ DO\ \ CONC \\
\{\ \ FOR\ \ i_1 = J_1^0(p), J_2^0(p)\ \ DO \\
\{\ \ p_{i_1} = 0 \\
\quad FOR\ \ s_1 = -m_1, m_1\ \ DO \\
\{\ \ p_{i_1} = p_{i_1} + x_{i_1 + s_1} * f_{s_1}\ \ \} \\
\quad x_{i_1} = p_{i_1}\ \ \ \} \ \ \ \} \ \ \ \}.
\end{array}
\tag{13}
$$

У цьому алгоритмі $J_1^0(p) = 1 + l_1(p-1) / P,\ J_2^0(p) = l_1 p / P$.

Якщо немає потреби особливо економити пам’ять, то конструкцію (11) можна подати у вигляді

$$
\begin{array}{l}
\quad FOR\ \ t = 1, C\ \ DO \\
\{\ \ FOR\ \ p = 1, P\ \ DO\ \ SIM \\
\{\ \ FOR\ \ i_1 = J_1(p), J_2(p), J_3\ \ DO \\
\{\ \ p_{i_1} = 0 \\
\quad FOR\ \ s_1 = -m_1, m_1\ \ DO \\
\{\ \ p_{i_1} = p_{i_1} + x_{i_1 + s_1}^{t-1} * f_{s_1}\ \ \} \\
\quad x_{i_1}^t = p_{i_1}\ \ \ \} \ \ \ \} \ \ \ \}.
\end{array}
\tag{14}
$$

У разі обмеження кількості паралельно виконуваних гілок алгоритм (8) подаємо у вигляді

$$
\begin{array}{l}
\quad FOR\ \ k = 1, P\ \ DO\ \ AUTON \\
\{\ \ FOR\ \ t = 1, C\ \ DO \\
\{\ \ FOR\ \ i_1 = \max\{1, m_1(t-C) + (k-1)l_1 / P + 1\}, \min\{l_1, m_1(C-t) + kl_1 / P\}\ \ DO \\
\{\ \ p = 0 \\
\quad FOR\ \ s_1 = -m_1, m_1\ \ DO \\
\quad \{\ \ p = p + x_{i_1 + s_1}^{t-1} * f_{s_1}\ \ \} \\
\quad\ \ x_{i_1}^t = p\ \ \ \} \ \ \ \} \ \ \ \}.
\end{array}
\tag{15}
$$

Аналогічно, як і для (8), у цьому випадку можна покращувати процес згладжування значень змінних. Час виконання паралельного алгоритму (15) обчислюємо за формулою $T_{po} = t_{op}(2m_1 + 1)C(l_1 / P + m_1(C-1))$.

Далі розглянемо деякі паралельні алгоритми з обмеженою кількістю гілок для розв'язання двовимірної ЗЦФ. Зокрема, такий алгоритм, що реалізує синхронний метод обчислень, матиме вигляд

$$
\begin{array}{l}
FOR\ \ t=\overline{1,C}\ \ DO \\
\{\ \ FOR\ \ p=\overline{1,P}\ \ DO\ \ SIM \\
\ \{\ \ FOR\ \ ALL\ \ (i_1,i_2)\in\{(i_1,i_2): i_1=\overline{J_1(p),J_2(p),J_3};\ i_2=\overline{1,l_2}\}\ \ DO \\
\qquad \{\ \ p_{i_1,i_2}=0 \\
\qquad\ \ FOR\ \ s_1=\overline{-m_1,m_1}\ \ DO \\
\qquad \{\ \ FOR\ \ s_2=\overline{-m_2,m_2}\ \ DO \\
\qquad \{\ \ p_{i_1,i_2}=p_{i_1,i_2}+x_{i_1+s_1,i_2+s_2}*f_{s_1,s_2}\ \ \}\ \ \}\ \ \}\ \ \} \\
\qquad\ \ FOR\ \ ALL\ \ (j_1,j_2)\in\{(j_1,j_2): j_1=\overline{1,l_1};\ j_2=\overline{1,l_2}\}\ \ DO \\
\qquad \{\ \ x_{j_1,j_2}=p_{j_1,j_2}\ \ \}\ \ \}.
\end{array}
\tag{16}
$$

Тут запис $i_1=\overline{J_1(p),J_2(p),J_3}$ означає, що змінна i_1 набуває значень від $J_1(p)$ до $J_2(p)$ з кроком J_3.

Алгоритм (16) працює за умови, що l_1 є кратним P, де $P=m_1+1$. Якщо знехтувати часовими витратами на синхронізацію паралельних гілок та реалізацію подвійного циклу за змінними j_1, j_2, то час виконання цього алгоритму можна обчислити за формулою

$$T_{sob}=t_{op}(2m_1+1)(2m_2+1)l_1l_2C/P.$$

У наведеній конструкції (16) третій рядок можна замінити фрагментом

$$\{\ \ FOR\ \ ALL\ \ (i_1,i_2)\in\{(i_1,i_2): i_1=\overline{1,l_1};\ i_2=\overline{J^1(p),J^2(p),J^3}\}\ \ DO.$$

Тут $J^1(p)=p,\ J^2(p)=p+(l_2/P-1)(m_2+1),\ J^3=m_2+1$. До того ж l_2 має бути кратним P, де $P=m_2+1$. Час роботи одержаного унаслідок такої заміни АОП збігається з часом виконання алгоритму (16), якщо $m_1=m_2$.

Не викликає особливих труднощів побудова аналогів алгоритмів (12) та (14) для розв'язання двовимірної задачі. Аналог (13) у цьому випадку матиме такий вигляд:

$$
\begin{array}{l}
FOR\ \ t=\overline{1,C}\ \ DO \\
\{\ \ \ FOR\ \ p=\overline{1,\ P}\ \ DO\ \ CONC \\
\ \{\ \ \ FOR\ \ ALL\ \ (i_1,i_2)\in\{(i_1,i_2): i_1=\overline{J_1^0(p),J_2^0(p)};\ i_2=\overline{1,l_2}\}\ \ DO \\
\qquad \{\ \ \ p_{i_1,i_2}=0 \\
\qquad\ \ FOR\ \ s_1=\overline{-m_1,m_1}\ \ DO \\
\qquad \{\ \ \ FOR\ \ s_2=\overline{-m_2,m_2}\ \ DO \\
\qquad \{\ \ \ p_{i_1,i_2}=p_{i_1,i_2}+x_{i_1+s_1,i_2+s_2}*f_{s_1,s_2}\ \ \}\ \ \} \\
\qquad\ \ x_{i_1,i_2}=p_{i_1,i_2}\ \ \}\ \ \}\ \ \}.
\end{array}
\tag{17}
$$

Зауважимо, що у наведеній конструкції l_1 є кратним P. Окрім цього, третій рядок в (17) можна подати фрагментом

$$\{\ \ FOR\ \ ALL\ \ (i_1,i_2)\in\{(i_1,i_2): i_1=\overline{1,l_1};\ i_2=\overline{J^{01}(p),J^{02}(p)}\}\ \ DO,$$

де $J^{01}(p)=1+l_2(p-1)/P,\ J^{02}(p)=l_2p/P$, а l_2 є кратним P.

Обмежуючи кількість паралельно виконуваних гілок, алгоритм (10) можна записати у такому вигляді:

$$
\begin{array}{l}
\quad FOR\ \ ALL\ \ (p,r) \in \{(p,r):\ p = \overline{1, P_1};\ r = \overline{1, P_2}\}\ \ DO\ \ PAR \\
\{\ \ FOR\ \ t = 1, C\ \ DO \\
\{\ FOR\ \ i_1 = \max\{1, (t-C)m_1 + (p-1)l_1 / P_1 + 1\}, \min\{l_1, (C-t)m_1 + pl_1 / P_1\}\ \ DO \\
\{\ FOR\ \ i_2 = \max\{1, (t-C)m_2 + (r-1)l_2 / P_2 + 1\}, \min\{l_2, (C-t)m_2 + rl_2 / P_2\}\ \ DO \\
\quad \{\ \ p = 0 \\
\quad FOR\ \ s_1 = -m_1, m_1\ \ DO \\
\quad \{\ \ FOR\ \ s_2 = -m_2, m_2\ \ DO \\
\quad\ \ \{\ \ p = p + x^{t-1}_{i_1+s_1,\, i_2+s_2} * f_{s_1, s_2}\ \ \} \\
\quad\ \ x^t_{i_1, i_2} = p\ \ \}\ \ \}\ \ \}\ \ \}\ \ \}.
\end{array}
\tag{18}
$$

У наведеній конструкції кількість паралельних гілок становить $P = P_1 P_2$, при цьому l_1 є кратним P_1, а l_2 є кратним P_2. Час виконання паралельного алгоритму (18) обчислюється за формулою

$$
T_{pob} = (l_1 l_2 / (P_1 P_2) + (m_1 l_2 / P_2 + m_2 l_1 / P_1)(C-1) + (2/3) m_1 m_2 (C-1)(2C-1)) \times
$$
$$
\times\, t_{op} (2m_1 + 1)(2m_2 + 1) C\,.
$$

ОЦІНЮВАННЯ ПРИСКОРЕННЯ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ

Зазначимо, що з метою спрощення викладу оцінки прискорення деяких розглянутих вище паралельних алгоритмів цифрової фільтрації будуть одержані за умови, що часові витрати на синхронізацію паралельних гілок та виконання циклів, тіла яких складаються тільки з оператора присвоювання, є незначними і ми будемо ними нехтувати.

Наведемо оцінки прискорення таких паралельних алгоритмів розв'язання одно- та двовимірної ЗЦФ:

- (7) та (9), що реалізують відповідні синхронні схеми обчислень;
- (8) та (10), які використовують метод пірамід для розпаралелювання циклів;
- алгоритмів (11) та (16) з обмеженою кількістю паралельних гілок, обчислення в яких синхронізуються;
- алгоритмів (15) і (18) з обмеженою кількістю повністю автономних гілок, тобто не зв'язаних між собою обмінами.

Твердження 1. Алгоритми (5), (7) (синхронна схема), (8), (11) та (15) є еквівалентними за інформаційним графом.

Доведення цього твердження випливає із аналізу терм-історій (дерев) відповідних алгоритмів обчислення значення змінної x_i на заданому кроці [10, 11]. Аналогічно, як і в [3, 4], нами встановлено, що такі дерева обчислюють той самий вираз.

Використовуючи оцінки часу виконання паралельних алгоритмів (7), (8), (11), (15) та послідовного алгоритму (5), отримуємо відповідні оцінки прискорення паралельних обчислень. Зокрема, прискорення алгоритму (7), який реалізує синхронну схему, обчислюватимемо за формулою

$$
S_{(7)} = T_1 / T_s = l_1,
$$

тобто за умови нехтування часовими витратами на синхронізацію прискорення набуває оптимального значення. Для прискорення паралельного алгоритму (8) отримуємо вираз

$$
S_{(8)} = T_1 / T_p = l_1 / (1 + m_1 (C-1))\,.
$$

Оскільки на практиці $l_1 >> C$, $l_1 >> m_1$ і l_1 відрізняється ($l_1 > m_1 C$) від $m_1 C$ не менш, як на декілька порядків, прискорення $S_{(8)}$ буде суттєвим. Наприклад, для $l_1 = 1000$, $m_1 = 1, C = 5$ маємо $S_{(8)} = 200$, а для $l_1 = 10000$, $m_1 = 7$, $C = 15$ маємо $S_{(8)} \approx 101.1$.

Прискорення алгоритму (11) обчислюватимемо за формулою

$$S_{(11)} = T_1 / T_{so} = P,$$

тобто воно набуває свого оптимального значення, а для прискорення алгоритму (15) отримуємо вираз

$$S_{(15)} = T_1 / T_{po} = l_1 / (l_1 / P + m_1 (C - 1)).$$

За зроблених вище припущень щодо співвідношень між величинами l_1, C, m_1 та $m_1 C$ отримуємо $S_{(15)} \approx P$, тобто у цьому разі прискорення є близьким до оптимального значення.

Твердження 2. Алгоритми (6), (9) (синхронна схема), (10), (16) та (18) є еквівалентними за інформаційним графом.

Доведення цього твердження випливає із аналізу терм-історій (дерев) відповідних алгоритмів обчислення значення деякої змінної $x_{i,j}$ на заданому кроці [3]. На підставі цього аналізу аналогічно, як і в [13], нами встановлено, що такі дерева обчислюють один і той самий вираз.

Використовуючи оцінки часу виконання паралельних (9), (10), (16), (18) та послідовного (6) алгоритмів, отримуємо відповідні оцінки прискорення паралельних обчислень. Зокрема, прискорення алгоритму (9) (синхронна схема) обчислюємо так:

$$S_{(9)} = T_2 / T_{s_1} = l_1 l_2,$$

тобто воно набуває свого оптимального значення за зробленого вище припущення щодо синхронізації паралельних гілок.

Прискорення паралельного алгоритму (10) подамо формулою

$$S_{(10)} = T_2 / T_{p_1} = l_1 l_2 / (1 + (2 m_1 m_2 (2C - 1) / 3 + m_1 + m_2)(C - 1)).$$

Зазвичай на практиці $l_1 >> m_1$, $l_2 >> m_2$, $l_1 >> C$, $l_2 >> C$ та l_1, l_2 відрізняються від $m_1 C^2$, $m_2 C^2$ ($l_1 > m_1 C^2, l_2 > m_2 C^2$) відповідно не менш, як на декілька порядків, тому $S_{(10)}$ буде суттєвим. Наприклад, для $l_1 = 100$, $l_2 = 200$, $m_1 = m_2 = 1$, $C = 5$ отримуємо $S_{(10)} \approx 606$, а для $l_1 = l_2 = 1000$, $m_1 = m_2 = 5$, $C = 10$ маємо $S_{(10)} \approx 340$.

Для алгоритму (16) прискорення подамо формулою

$$S_{(16)} = T_2 / T_{sob} = P,$$

тобто воно набуває оптимального значення за зроблених вище припущень щодо часу синхронізації паралельних гілок.

Прискорення паралельного алгоритму (18) обчислюємо за формулою

$$S_{(18)} = T_2 / T_{pob} = \left(\frac{1}{P_1 P_2} + \left(\frac{m_1}{P_2 l_1} + \frac{m_2}{P_1 l_2} \right)(C - 1) + 2 m_1 m_2 (C - 1)(2C - 1) / (3 l_1 l_2) \right)^{-1}.$$

За зроблених припущень щодо співвідношень між l_1, l_2, m_1, m_2, C, P_1, P_2 та $m_1 C$, $m_2 C$ одержуємо $S_{(18)} \approx P_1 P_2 = P$. Отже, прискорення алгоритму (18) у цьому разі є близьким до свого оптимального значення.

РЕЗУЛЬТАТИ ЧИСЕЛЬНОГО ЕКСПЕРИМЕНТУ

Здійснено програмну реалізацію АОП (13), (15), (16), (17) та (18) на комп'ютерах з багатоядерним процесором і отримано значення їхнього реального прискорення. Зокрема, алгоритми (13) та (17) було реалізовано мовою C# з використанням бібліотеки паралельних задач (TPL) [14]. При цьому здійснювалась фільтрація пошкоджених одновимірних сигналів (гармонічного, широкого прямокутного імпульсу та пиловидного), а також відновлювалось бінарне зображення, спотворене імпульсними завадами. Вагові коефіцієнти вибирали, використовуючи формули Даніеля, Бартлета, Хемінга [15], та задавали випадково. Було розглянуто низку варіантів розв'язання ЗЦФ за (13) та (17) на комп'ютерах з багатоядерним процесором у таких конфігураціях: 2 ядра — 2 потоки, 2 ядра — 4 потоки, 4 ядра — 4 потоки. Для кожного варіанта в цих конфігураціях фіксувався реальний час виконання відповідного АОП. Встановлено, що унаслідок реалізації паралельних алгоритмів фільтрації на двох ядрах процесора з організацією чотирьох потоків порівняно з їхнім виконанням на тих самих ядрах із організацією двох потоків одержуємо незначне прискорення обчислень (в 1.03–1.32 раза). Однак реалізація згаданих алгоритмів на чотирьох ядрах процесора з організацією чотирьох потоків порівняно з їхнім виконанням на двох ядрах процесора дає більш суттєве прискорення (в 1.23–1.92 раза).

Алгоритми (15) та (18) було реалізовано на мові C++ з використанням комп'ютера з восьмиядерним процесором для різних варіантів розв'язання одно- та двовимірної ЗЦФ. Зокрема, прискорення алгоритму (15) порівняно з виконанням за (5) із залученням двох, чотирьох та восьми ядер становило відповідно 1.80–1.94; 3.04–3.70 та 3.73–6.15 раза. Прискорення алгоритму (18) порівняно з (6) із залученням чотирьох та восьми ядер становило відповідно 2.80–3.62 та 3.10–7.50.

Паралельні алгоритми (16) та (18) було реалізовано з використанням мови C++ на комп'ютері з двоядерним процесором. Для різних значень параметрів двовимірної ЗЦФ їхнє прискорення порівняно з алгоритмом (6) становило відповідно 1.86–1.91 та 1.50–2.09 раза. Значні прискорення алгоритмів (16) (навіть за наявності синхронізації на кожному кроці переобчислення) та (18) у цьому випадку можна пояснити ефективним використанням кеш-пам'яті процесора.

ВИСНОВКИ

У роботі запропоновано паралельні алгоритми розв'язання задач фільтрації різної розмірності на обчислювальних засобах універсального призначення. Для цих алгоритмів отримано теоретичні оцінки складності та прискорення, які підтверджують їхню високу ефективність. Здійснено програмну реалізацію запропонованих АОП на комп'ютерах з багатоядерним процесором і для окремих варіантів розв'язання ЗЦФ встановлено реальні оцінки прискорення обчислень, які добре узгоджуються із відповідними теоретичними оцінками. Одержані наукові результати можна використовувати для попереднього опрацювання в режимі реального часу великих масивів даних у різних предметних галузях з використанням сучасних програмних та апаратних засобів [16] універсального призначення.

СПИСОК ЛІТЕРАТУРИ

1. Тимченко О.В. Різницеві методи цифрової фільтрації. Львів: Фенікс, 1999. 388 с.
2. Ярославский Л.П. Цифровая обработка сигналов в оптике и голографии. Введение в цифровую оптику. Москва: Радио и связь, 1987. 296 с.

3. Anisimov A.V., Yadzhak M.S. Construction of optimal algorithms for mass computations in digital filtering problems. *Cybernetics and Systems Analysis*. 2008. Vol. 44, N 4. P. 465–476. https://doi.org/10.1007/s10559-008-9018-8.
4. Yadzhak M.S., Tyutyunnyk M.I. An optimal algorithm to solve digital filtering problem with the use of adaptive smoothing. *Cybernetics and Systems Analysis*. 2013. Vol. 49, N 3. P. 449–456. https://doi.org/10.1007/s10559-013-9528-x.
5. Каляев А.В., Левин И.И. Модульно-наращиваемые многопроцессорные системы со структурно-процедурной организацией вычислений. Москва: Янус-К, 2003. 380 с.
6. Штейнберг Б.Я. Математические методы распараллеливания рекуррентных циклов для суперкомпьютеров с параллельной памятью. Ростов н/Д: Изд-во Рост. ун-та, 2004. 192 с.
7. Jadzhak M.S. On construction of algorithms with the bounded parallelism for solving problems of digital filtering. *Journal of Automation and Information Sciences*. 2002. Vol. 34, N 12. P. 12–21. https://doi.org/10.1615/JAutomatInfScien.v34.i12.20.
8. Яджак М.С. Аналіз реалізації алгоритмів з обмеженим паралелізмом для цифрової фільтрації. *Відбір і обробка інформації.* 2009. Вип. 30 (106). С. 162–167.
9. Яджак М.С. Вирішення проблеми реалізації деяких паралельних алгоритмів цифрової фільтрації даних. *Відбір і обробка інформації*. 2011. Вип. 35 (111). С. 116–121.
10. Valkovskii V.A. An optimal algorithm for solving the problem of digital filtering. *Pattern Recognition and Image Analysis*. 1994. Vol. 4, N 3. P. 241–247.
11. Вальковский В.А. Распараллеливание алгоритмов и программ. Структурный подход. Москва: Радио и связь, 1989. 176 с.
12. Вальковський В.О., Яджак М.С. Проблеми подальшого розвитку та модифікації методу пірамід для розпаралелювання циклів. *Математичні методи та фізико-механічні поля*. 2000. Т. 43, № 1. С. 68–75.
13. Valkovskiy V.A., Yadzhak M.S. The optimal solution algorithm for the two-dimensional problem of digital filtering. *Journal of Automation and Information Sciences*. 1999. Vol. 31, N 12. P. 72–80. https://doi.org/10.1615/JAutomatInfScien.v31.i12.90.
14. Toub S. Patterns of parallel programming. Microsoft Corporation, 2010. 118 p.
15. Файнзильберг Л.С. Адаптивное сглаживание шумов в информационных технологиях обработки физиологических сигналов. *Математичні машини і системи*. 2002. № 3. С. 96–104.
16. Штейнберг Б.Я., Штейнберг О.Б. Преобразования программ — фундаментальная основа создания оптимизирующих распараллеливающих компиляторов. *Программные системы: теория и приложения.* 2021. Т. 12, № 1. С. 21–113. https://doi.org/10.25209/2079-3316-2021-12-1-21-113.

M.S. Yadzhak

PARALLEL ALGORITHMS OF DIGITAL DATA FILTERING

Abstract. The paper proposes parallel algorithms for solving digital filtering problems of different dimensions using modern universal computing systems. Theoretical estimates of the complexity and speed-up are obtained, which confirm the high efficiency of these algorithms. The software implementation of some of the proposed parallel algorithms using computers with a multi-core processor is carried out, and real estimates of the speed-up are obtained, which agree well with the theoretical ones.

Keywords: digital filtering, parallel algorithm, speed up of computations, limited parallelism, equivalence of algorithms, computing system.

Надійшла до редакції 21.07.2021