УДК 004.932.2:004.032.24

Організація паралельних обчислень при відстежуванні об'єктів у відеопотоці

Ю.В. Ладиженський, А.О. Середа
Донецький національний технічний університет
ly@cs.dgtu.donetsk.ua, aas11@bk.ru

Abstract

Ladyzhenskyy Y., Sereda A. The Organization of Parallel Computing for Object Tracking in a Video Stream. *Approaches to parallel object tracking in videos are analyzed. Parallel algorithms of object tracking are proposed and implemented for MIMD systems. Performance and optimal load balancing are evaluated.*

Вступ

Необхідність відстежування об'єктів у відеопотоці виникає в багатьох областях діяльності. Під відстежуванням об'єктів у відеопотоці розуміється отримання списку видимих об'єктів і їх координат в кожному кадрі.

У [1] для відстежування цілого об'єкту використовується шаблон, що динамічно оновлюється, який складається з зображення об'єкту і матриці приналежності пікселів до об'єкту.

У [2] об'єкт відстежується як множина точок зображення, що мають певні властивості.

У [3] описано метод відстежування об'єктів у відеопотоці на основі відстежування переміщення фрагментів об'єктів, що поєднує властивості приведених вище методів. При виконанні на сучасному одноядерному процесорі метод не забезпечує швидкодію, достатню для аналізу у реальному часі відео високої роздільної здатності або відео з декількох камер. Для його прискорення доцільно використовувати паралельні обчислювальні системи.

Основні типи паралельних обчислювальних засобів, що можуть бути застосовані для паралельного аналізу відео:

1. Спеціально розроблені для вирішення конкретних задач обчислювальні структури. Наприклад, алгоритми пошуку фрагментів зображення можуть бути ефективно реалізовані на систолічних масивах [4], модифікації яких можуть бути використані для пошуку фрагментів по методу [3]. У [5] описаний спецпроцесор для пошуку фрагментів зображення для мобільних пристроїв.

2. Універсальні паралельні архітектури, орієнтовані на обробку векторів. У [6] описано успішне застосування архітектури CUDA, яка визначається як SIMT (single-instruction, multiple-thread) для відстежування облич.

3. Використовування паралельних обчислювальних систем з процесорами загального призначення. До них можна віднести комп'ютери з багатоядерним процесором, які фактично стали стандартними і мають паралельну архітектуру SMP (Symmetric Multiprocessing) UMA (Uniform Memory Access) і кластери, складені з них. Достоїнствами таких систем є їх доступність і можливість розпаралелювати виконання алгоритмів, які погано відображаються на векторно-орієнтовану паралельну архітектуру.

Однією з популярних технологій створення паралельних програм є використання бібліотеки MPI (Message Passing Interface), що являє собою стандартізований набір засобів для обміну повідомленнями і може використовуватись як на SMP системах, так і на кластерах [7].

Можливий паралелізм в послідовному алгоритмі можна виявити як при незалежній обробці масивів однотипних даних, так і для різнорідних етапів алгоритму, які можуть бути виконані одночасно. Для визначення можливості паралельного виконання етапів складного алгоритму використовується побудова графа інформаційних залежностей алгоритму [7].

У даній статті аналізуються підходи до паралельного відстежування об'єктів у відео і будується паралельний алгоритм відстежування об'єктів по методу [3] на системах типу MIMD. При реалізації використовуються технологія MPI. Отримуються оцінки ефективності, а також оптимальне балансування навантаження на різних системах.

Алгоритм відстежування об'єктів

У методі [3] кожен рухомий об'єкт розглядається як сукупність рухомих фрагментів (рис. 1). Кожен фрагмент об'єкту є областю кадру довільного розміру і форми, що містить частину об'єкту або цілий об'єкт, і задається своїм зображенням і матрицею, що визначає приналежність пікселів до фрагмента. Такий підхід дозволяє відстежувати частково перекриті об'єкти.

В алгоритмі використовуються модель статичного фону, множина об'єктів і множина їх фрагментів. На початку роботи алгоритму

множини фрагментів і об'єктів порожні. В процесі відстежування по черзі аналізуються кадри відео, що приходять. UML діаграма діяльності для послідовного алгоритму аналізу одного кадру приведена на рис. 2.

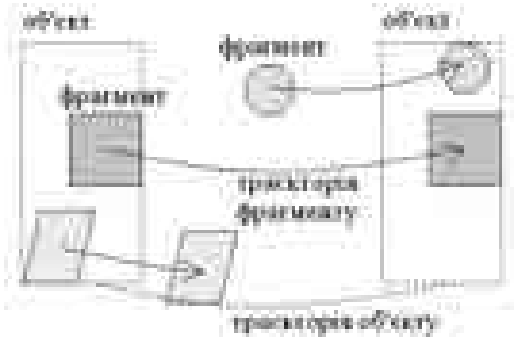

Рисунок 1 – Відстежування об'єкту як множини фрагментів

Рисунок 2 – Послідовний алгоритм аналізу кадру

Віднімання фону (*C*1) використовується для визначення пікселів кадру, які відносяться до об'єктів. Побудова моделі фону описана в [8].

Розпізнаванню об'єктів (*C*3) на багатошаровому персептроні [9] (наприклад, силуету людини) підлягають області кадру, що містять групи близько розташованих пікселів, відмінних від фону (*C*2).

Для кожного фрагмента, положення якого відоме в попередніх кадрах, на основі моделі його руху визначається область (*C*4), в якій він може знаходитися в поточному кадрі, і в ній шукається його положення (*C*5).

Нові відстежувані фрагменти (*C*6) створюються один раз в декілька кадрів, щоб покрити пікселі, які не відносяться ні до фону, ні до фрагментів.

Встановлення відповідності між фрагментами і об'єктами та побудова траєкторій об'єктів (*C*7) здійснюється на основі траєкторій руху їх фрагментів. У об'єкти об'єднуються групи близько розташованих фрагментів, які рухаються узгоджено. Обробляються ситуації, пов'язані з виникненням, зникненням, об'єднанням і розділенням об'єктів.

Підходи до розпаралелювання відстежування об'єктів

Розпаралелювання можна виконати наступними способами:

1. При аналізі даних з декількох відеокамер доцільно виконувати аналіз і операції введення-виведення для кожної камери на окремому процесорі.
2. При аналізі відеозапису можливо розбити його за часом на декілька відрізків і проаналізувати їх паралельно, після чого визначити відповідності між об'єктами на межах сусідніх відрізків.
3. Кожен кадр відеопотоку можна розділити за площею на декілька частин і створити з цих частин декілька окремих відеопотоків меншої роздільної здатності, кожнен з яких аналізується на окремому процесорі.
4. Можливо провести розпаралелювання всередині алгоритму аналізу одного кадру.

Перші три способи розпаралелювання не потребують модифікації послідовного алгоритму відстежування, процеси незалежно виконуються на архітектурі типа MIMD і не обмінюються повідомленнями під час аналізу. Це дозволяє досягти прискорення, близького до числа використовуваних процесорів.

Якщо необхідно аналізувати у реальному часі відео високої роздільної здатності, для роботи з яким не вистачає продуктивності одного процесора, то необхідно використовувати третій або четвертий спосіб розпаралелювання.

Третій спосіб має істотні недоліки: щоб не втратити об'єкти на межах областей, на які поділено кадр, ці області повинні перетинатись. Також потрібно визначення відповідності між об'єктами на межах областей. Це збільшує загальне необхідне число операцій і підвищує ймовірність помилок.

Четвертий спосіб є найбільш складним в реалізації, але потенційно дозволяє одержати максимально можливу швидкодію при аналізі відеопотоку у реальному часі. Він розглядається нижче детальніше.

Оцінка можливості розпаралелювання етапів алгоритму аналізу кадру

Доцільно виконувати паралельно ті етапи алгоритму на рис. 2, які займають багато часу. Експериментально отриманий середній час виконання основних груп операцій при аналізі кадру футбольного матчу розміром 672x368 на одному ядрі процесора Intel Core 2 6320 приведено в табл. 1.

Таблиця 1

етап	$C1$	$C2$	$C3$	$C4$	$C5$	$C6$	$C7$
час, мс	2.36	0.41	1.84	2.24	16.1	0.92	2.81

Операціями розробленого алгоритму для паралельного виконання є:

1. Відстежування фрагментів об'єктів ($C5$) – найбільш ресурсоємний етап алгоритму. Пошук кожного фрагмента може бути виконаний незалежно. Можливо виконувати пошук однакової кількості фрагментів на кожному процесорі. Для кращого балансування можна оцінити число операцій, що потрібне для пошуку кожного фрагменту і розподілити фрагменти по процесорах так, щоб сумарне очікуване число операцій на кожному з них відрізнялося мало. Наприклад, для алгоритму повного пошуку число операцій складає $O(S_f W_s H_s)$, а для швидкого алгоритму Three-Step Search – $O(S_f[1+8\log_2(\max(W_s,H_s)+1)])$, де S_f – площа фрагменту, а W_s і H_s – ширина і висота області пошуку [4].

2. Віднімання фону ($C5$) може бути розпаралелено шляхом розбиття кадру на області. Кожен процесор зберігає модель фону і віднімає фон в одній такій області. При балансуванні навантаження можна вважати, що час віднімання фону пропорційний площі зображення.

3. Розпізнавання об'єктів ($C3$). При використовуванні складного алгоритму розпізнавання час виконання цього етапу може зрости. Розпізнавання кожного об'єкту може бути виконане незалежно; всі області, що підлягають розпізнаванню в кожному кадрі, можна розподілити між різними процесорами.

Операції, які виконуються послідовно:

1. Пошук областей кадру, що підлягають розпізнаванню ($C2$) і створення нових фрагментів ($C6$) займають відносно малу частку процесорного часу.

2. Визначення областей пошуку фрагментів ($C4$) потрібно виконати до розподілу фрагментів по процесорах для виконання балансування навантаження етапу $C5$.

3. Багато кроків алгоритму встановлення відповідності між фрагментами і об'єктами та побудови траєкторій об’єктів ($C7$) потребують наявності у кожного процесора повної множини об'єктів і їх фрагментів, одержаної на попередньому кроці, що при розпаралелюванні викликає необхідність багатократних обмінів даними та синхронізацій.

Паралельний алгоритм аналізу кадру

Назвемо один з процесорів основним. На ньому виконуватимуться операції введення і виведення, етапи $C2$, $C4$, $C6$ і $C7$ алгоритму на рис. 2, а також управляючий код, що зв'язує послідовні і паралельні етапи алгоритму.

Основний процесор розподіляє завдання між рештою процесорів і отримує від них результати. Решта процесорів не обмінюється повідомленнями між собою.

Для побудови паралельного алгоритму замінимо у алгоритмі на рис. 2 кожний з етапів $C1$, $C3$ і $C5$ сукупністю трьох операцій:

1. Розподіл завдань по процесорах.

2. Паралельне обчислення на всіх процесорах, включаючи основний.

3. Об'єднання результатів на основному процесорі.

У програмній реалізації окремо описані потоки команд для основного і додаткових процесорів.

UML діаграма діяльності для отриманого паралельного алгоритму аналізу кадру приведена на рис. 4. Дії етапів послідовного алгоритму $C1$, $C3$ і $C5$, що виконуються на основному процесорі, позначені як $M1$, $M3$ і $M5$, а на інших процесорах – $O1$, $O3$ і $O5$ відповідно. Не показані допоміжні операції по балансуванню даних між процесорами, які виконуються безпосередньо перед пересилання даних.

Перед початком аналізу кадру в основному процесі є множини об'єктів і їх фрагментів, отримані на попередньому кроці, а в кожному процесі – модель фону у відповідній частині кадру.

Оптимізація порядку виконання операцій

Для мінімізації простою процесорів через очікування результатів від інших процесорів, бажано асинхронно відправляти результати, необхідні іншим процесорам, як тільки вони будуть обчислені, і відкладати синхронне отримання даних від інших процесорів до тих пір, поки вони не знадобляться на даному процесорі.

Граф інформаційних залежностей алгоритму приведений на рис. 5. Прямокутниками показані операції, стрілками – залежності між ними. Як видно з рис. 5, етапи $C2$ і $C3$ можна виконувати паралельно з етапами $C4$, $C5$ і $C6$, а етап $C4$ – паралельно з етапом $C1$.

UML діаграма діяльності оптимізованого паралельного алгоритму аналізу кадру приведена на рис. 6.

Програмна реалізація

Існуюча послідовна програмна реалізація алгоритму відстежування [3] для ОС Windows на мові C++ була модифікована згідно з паралельним алгоритмом на рис. 6. Було використано MPI від Microsoft, що базується на реалізації з відкритим кодом MPICH2. Запуск і управління процесами здійснюється системою MS Windows Compute Cluster Server [10].

Перед пересиланням даних, що знаходяться в динамічних структурах, відбувається їх упаковка в безперервну область пам'яті, вміст якої потім пересилається як масив байт.

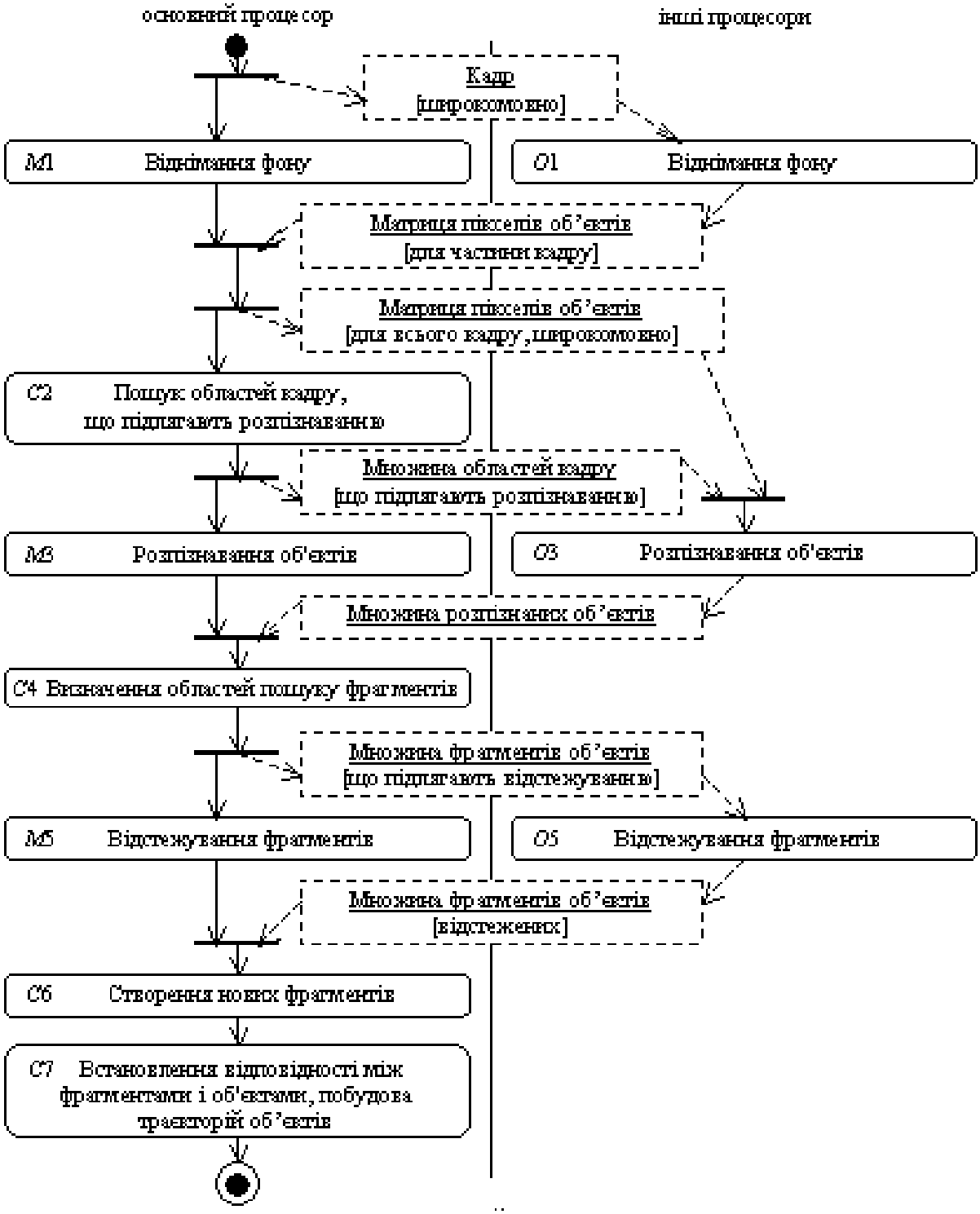

Рисунок 4 – Паралельний алгоритм аналізу кадру

Рисунок 5 – Граф інформаційних залежностей алгоритму відстежування

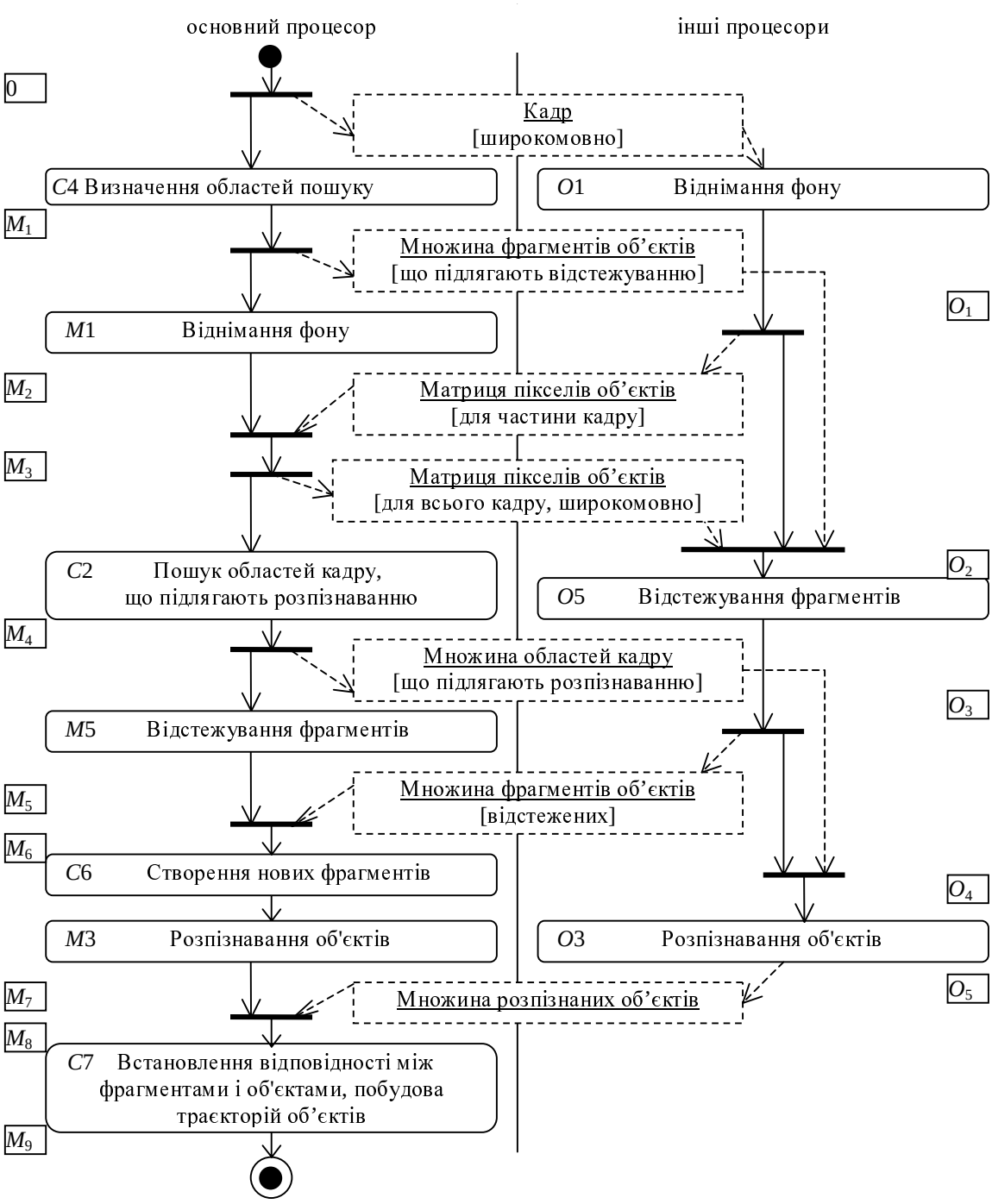

Рисунок 6 – Оптимізований паралельний алгоритм аналізу кадру

Оцінка ефективності

Отримаємо оцінку часу роботи алгоритму на рис. 6 на різних паралельних системах.

Для спрощення приймемо, що навантаження на усіх процесорах, окрім основного, для всіх етапів алгоритму, що виконуються паралельно, збалансоване. При паралельному виконанні найбільш навантаженим є основний процесор і загальний час паралельного аналізу кадру практично дорівнює часу його аналізу на основному процесорі.

Нехай використовується N процесорів, причому на кожному комп'ютері, що входить в кластер, розташовані N_l процесорів.

Позначимо час виконання етапу X алгоритму через T_X. Час виконання паралельних етапів алгоритму на основному та інших процесорах дорівнює $T_{Mi} = T_{Ci}(1 - A_i)$ і $T_{Oi} = T_{Ci}A_i/(N-1)$ відповідно, де $i \in \{1,3,5\}$ – номер етапу, а A_1, A_3 и A_5 – доля обчислень етапів $C1$, $C3$ і $C5$, які виконуються на усіх процесорах, окрім основного.

Будемо вважати, що упакування або розпакування S байт даних, що передаються, займає K_lS одиниць процесорного часу. Для спрощення приймемо, що час передачі S байт по мережі (не включаючи упакування і розпаковування) дорівнює K_nS, а час передачі між процесорами одного обчислювального вузла дорівнює нулю.

Розглянемо асинхронну індивідуальну розсилку $S/(N\text{-}1)$ байт від основного процесора до кожного з інших. Приклад виконання такої розсилки при $N=4$ і $N_l=2$ показаний на рис. 7а. Пунктирними лініями позначені осі часу для кожного з процесорів, цифрами – ранги процесорів (0-основний), стрілками – передача даних, прямокутниками – процесорний час. При використанні мережі Ethernet ($N\text{-}N_l$) асинхронних надсилань конкурують за використання каналу і їх сумарний час передачі дорівнює $K_nS(N\text{-}N_l)/(N\text{-}1)$. Найпізніший момент від початку розсилки, коли може початися пересилка даних по мережі – $K_lS/(N\text{-}1)N_l$. При $K_{local}<K_{net}$ можна вважати, що пересилання всіх даних по мережі буде завершено через $K_lS(N-N_l)/(N-1)$ після її початку. Оскільки пересилання є асинхронним, моменти читання і розпакування даних можуть бути перенесені в майбутнє на будь-який час. Тоді для визначення затримок можна використовувати спрощену схему, представлену на рис. 7б.

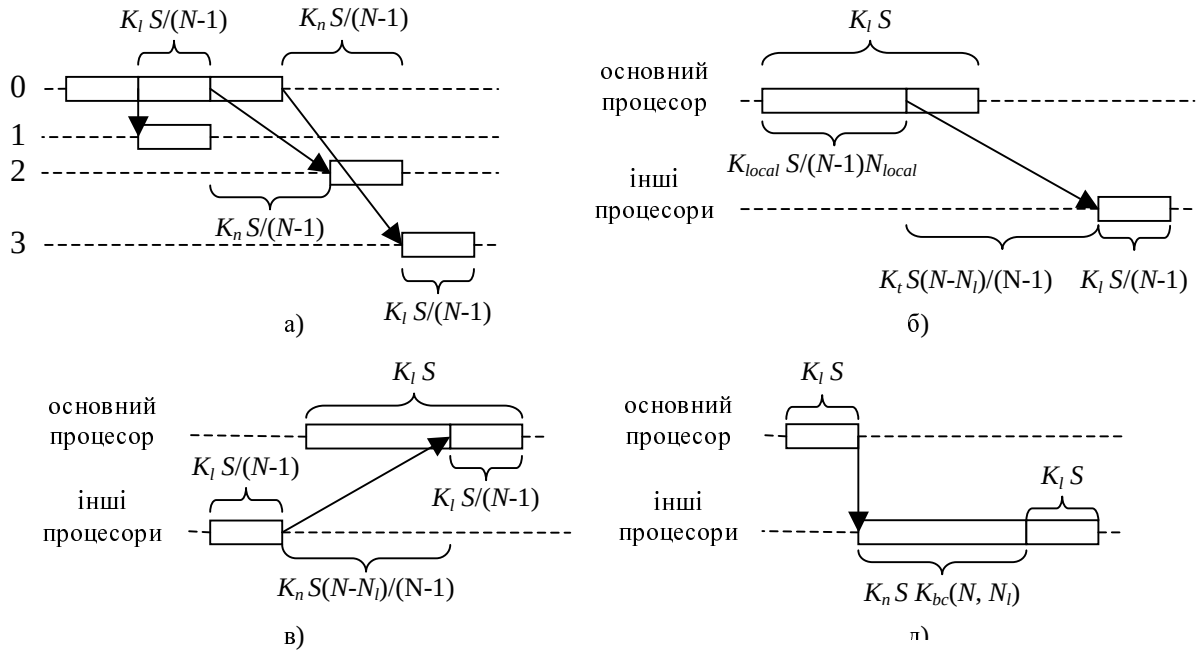

Рисунок 7 – Затримки при обміні даними

Аналогічно отримано спрощені схеми для визначення затримок при асинхронному зборі даних основним процесором (рис. 7в) і широкомовної розсилки (рис. 7д). Коефіцієнт $K_{bc}(N,N_l)$ залежить від реалізації широкомовної розсилки. При використовуванні широкомовних пакетів K_{bc}=1. У реалізації MPI MPICH2 при (N≤7) використовується розсилка по біноміальному дереву і експериментально було встановлено, що $K_{bc}(N,2) = \lceil \log_2(N-2) \rceil$.

Нехай загальний обсяг даних, необхідних для передачі всього кадру, дорівнює S_{frame}, матриці пікселів об'єктів всього кадру – S_{back}, всіх фрагментів – S_{frag} і всіх областей, що підлягають розпізнаванню – S_{rec}.

На рис. 6 показані відмітки часу від початку аналізу кадру для основного процесора $M_1 \ldots M_9$ і для додаткових процесорів $O_1 \ldots O_5$. Виразимо кожну з відміток часу через попередні відмітки часу, а також тривалість обчислень і передачі даних, необхідних щоб перейти до даної відмітки:

$$O_1 = 0 + (K_l + K_n K_{bc} + K_l) S_{frame} + T_{C1} A_1/(N\text{-}1);$$

$$M_1 = 0 + K_l S_{frame} + T_{C4};$$

$$M_2 = M_1 + K_l S_{frag} A_5 + T_{C1} (1\text{-}A_1);$$

$$M_3 = \max(M_2 + K_l S_{back} A_1 (N\text{-}2)/(N\text{-}1), O_1 + K_l S_{back} A_1/(N\text{-}1) + K_n S_{back} A_1 (N\text{-}N_l)/(N\text{-}1)) + K_l S_{back} A_1/(N\text{-}1);$$

$$O_2 = \max(M_3 + K_l S_{back} + K_{bc} K_n S_{back}, O_1 + K_l S_{back} A_1/(N\text{-}1)) + K_l S_{back};$$

$$M_4 = M_3 + K_l S_{back} + T_{C2};$$

$$M_5 = M_4 + K_l S_{rec} A_3 + T_{C5} (1\text{-}A_5);$$

$$M_6 = \max(M_5 + K_l S_{frag} (N\text{-}2)/(N\text{-}1), O_3 + K_l S_{frag} A_5/(N\text{-}1) + K_n S_{frag} A_5 (N\text{-}N_l)/(N\text{-}1)) + K_l S_{frag} A_5 /(N\text{-}1);$$

$$O_4 = \max(M_4 + K_l S_{rec} A_3 N_l/(N\text{-}1) + K_n S_{rec} A_3 (N\text{-}N_l)/(N\text{-}1), O_3 + K_l S_{frag} A_5 /(N\text{-}1)) + K_l S_{rec} A_3 (N\text{-}1);$$

$$M_7 = M_6 + T_{C6} + T_{C3} (1\text{-}A_3);$$

$$O_5 = O_4 + T_{C3} A_3/(N\text{-}1);$$

$$M_8 = \max(M_7 + K_l S_{rec} (N\text{-}2)/(N\text{-}1), O_5 + K_l S_{rec} A_3/(N\text{-}1) + K_n S_{rec} A_3 (N\text{-}N_l)/(N\text{-}1)) + K_l S_{rec} A_3 /(N\text{-}1);$$

$$M_9 = M_8 + T_{C7}. \qquad (1)$$

$T_{C1} \ldots T_{C7}$, S_{back}, S_{frag} і S_{rec} залежать від параметрів алгоритму відстежування і його вхідних даних, а K_l, K_n, N_l і N – від конфігурації обчислювальної системи. При незмінних параметрах алгоритму відстежування, вхідних даних і конфігурації обчислювальної системи, можна розглядати M_9 як функцію від A_1, A_3 и A_5. $M_9(A_1,A_3,A_5)$ можливо мінімізувати, враховуючи обмеження $(0 \le A_1 \le 1)$, $(0 \le A_3 \le 1)$ і $(0 \le A_5 \le 1)$. Для визначення оптимального балансування навантаження обчислимо

$$(A_1, A_3, A_5) = \arg\min M_9(A_1, A_3, A_5). \qquad (2)$$

Експериментально отримані кількісні величини при аналізі відеозапису футбольного матчу роздільної здатності 672x368 на кластері Донецького національного технічного університету, що включає комп'ютери з процесорами Intel Core 2 6320 і мережу Ethernet 100 Мбіт/с, наведено в табл. 2. Залежності оптимального навантаження (2) від N в цих умовах приведено на рис. 8. Як видно з рис. 8, при (N>2) віднімання фону (A_1=0) і відстежування фрагментів (A_5=0) невигідно виконувати на додаткових процесорах через великий час передачі даних по мережі. Розпізнавання (A_3=0) вигідно виконувати повністю на додаткових процесорах через малий обсяг даних, що передаються.

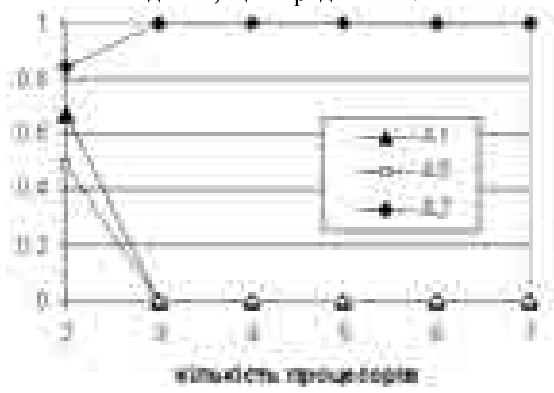

Рисунок 8 – Оптимальне балансування навантаження на кластері ДонНТУ

Оцінка часу (1) при оптимальному навантажені і реальне значення середнього часу аналізу кадру при різному числі процесорів на кластері ДонНТУ показані на рис. 9. Час виконання на одному процесорі наведено для послідовної програми, що не використовує MPI. Стрибкоподібне збільшення часу роботи відбувається через розсилання даних по біноміальному дереву у функції MPI_Bcast із бібліотеки MPI. Як видно з рис. 9, виконання алгоритму на рис. 6 при аналізі відео футбольного матчу при використовуванні мережі Ethernet 100 Мбіт/с призводить не до прискорення, а до уповільнення аналізу.

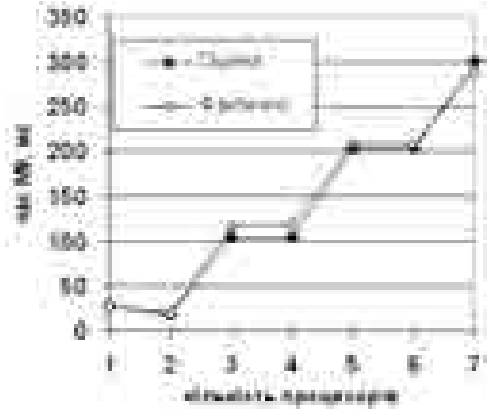

Рисунок – 9. Середній час анализу кадру на кластері ДонНТУ

Більшого прискорення можна досягти:

1. При використанні мережі з більшою пропускною спроможністю.

2. При використанні структур даних, що не потребують упаковки при відправці. Така зміна вимагає істотної переробки наявної послідовної програми.

3. При реалізації алгоритму на системі UMA без використання MPI (наприклад, стандартними засобами для роботи з потоками ОС Windows). Відправку даних можна звести до передачі їх адреси в пам'яті без упаковки даних в безперервний буфер. В цьому випадку можна вважати $K_l = 0$.

Оцінку часу (1) при оптимальному балансуванні навантаження при використанні мережі Ethernet 1 Gbit/s і 10 Gbit/s, а також при використанні системи UMA з K_l=0, показано на рис. 10. Усі використані чисельні параметри, окрім K_l і K_n, наведено у табл. 1 та табл. 2.

Таблиця 2

величина	значення
S_{frame}	715198
S_{back}	250168
S_{frag}	77716
S_{rec}	193
K_l	$1.63*10^{-6}$ мс
K_n	$9.78*10^{-5}$ мс
N_l	2

Як видно з рис. 10, при реалізації алгоритму з використанням MPICH2 на кластері з двопроцесорних комп'ютерів прискорення можна досягти при використанні мережі Ethernet 10 Gbit/s. Залежність оптимального балансування навантаження (2) від N для мережі Ethernet 10 Gbit/s показана на рис. 11. Стрибкоподібні зменшення A_1 при зростанні N пояснюються стрибкоподібними збільшеннями K_{bc}.

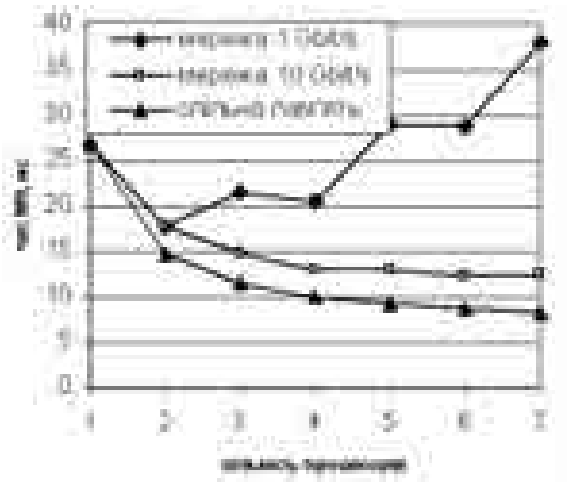

Рисунок 10 – Оцінка часу анализу кадру на різних параллельных системах

Як видно з рис. 10, ще більше прискорення при менших апаратних витратах можна отримати при реалізації алгоритму на рис. 6 на паралельній системі UMA з K_l=0 (наприклад, як багатопотокової програми, яка виконується на багатоядерному процесорі). Таким чином був реалізований алгоритм на рис. 4 і середній час аналізу кадру в два потоки на одному комп'ютері з кластера ДонНТУ склав 17.2 мс.

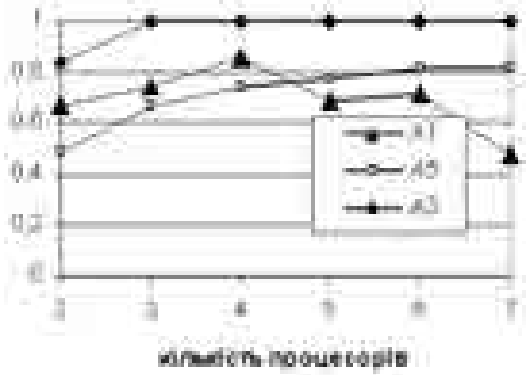

Рисунок 11 – Оптимальне балансування з використанням Ethernet 10 Gbit/s

Висновки

Проаналізовані підходи до паралельного відстежування об'єктів.

Розроблено паралельний алгоритм відстежування об'єктів у відеопотоці по методу [3] на паралельній архітектурі MIMD. Алгоритм використовує асинхронну відправку даних та асиметричне статичне балансування навантаження. Алгоритм реалізовано за допомогою технології MPI.

Отримані теоретичні та експериментальні оцінки часу роботи паралельного алгоритму, а також оптимального балансування навантаження на різних системах.

Доцільно виконувати аналіз одного відеопотоку у реальному часі за розробленим алгоритмом на одному комп'ютері з числом процесорів від двох до чотирьох. При виконанні на одному багатопроцесорному комп'ютері, використовувати багатопоточность в рамках одного процесу вигідніше, ніж декілька процесів, що обмінюються даними по протоколу MPI, через менші накладні витрати на пересилку даних. Практично отримано прискорення на двох процесорах складає 1.33 разів з використанням MPI і 1.57 разів при виконанні у два потока.

Для досягнення ще більшого прискорення аналізу, доцільно виконувати окремі операції, зокрема, пошук фрагментів зображень, на спеціалізованих паралельних співпроцесорах, наприклад, графічних прискорювачах, а також використовувати динамічне балансування навантаження.

Література

1. R. Cucchiara, C. Grana, G. Tardini. Track-based and object-based occlusion for people tracking refinement in indoor surveillance. //Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks 2004, New York, NY, USA, October 15, 2004. – 10 p. (http://doi.acm.org/10.1145/1026799.1026814)
2. Pabboju Sateesh Kumar, Prithwijit Guha, Amitabha Mukerjee. Clolour and Feature Based Multiple Object Tracking Under Heavy Occlusions. //ICAPR 2007, Calcutta, 2-4 January 2007 (http://www.cse.iitk.ac.in/research/pdfs/ddmtech/sateesh-guha-mukerjee-icapr07_ colour-feature-multi-obj-tracking-occlusion.pdf)
3. Ладиженський Ю.В., Середа А.О. Відстежування об'єктів у відеопотоці на основі відстежування переміщення фрагментів об'єктів / Ю.В.Ладиженський, А.О.Середа // Наукові праці Донецького національного технічного університету. Серія: Обчислювальна техніка та автоматизація. – 2009. – Вип. 17 (148). – 215 с.
4. Sheu-Chih Cheng, Hsueh-Ming Hang. A comparsion of Block-Matching Algorithms Mapped to Systolic-Array Implementation. IEEE Transactions on Circuits and Systems for Video Technology, vol.7, № 5, 1997 – 17 p.
5. Adaptive Motion Estimation Processor for Autonomous Video Devices. T. Dias, S. Momcilovic, N. Roma, L. Sousa. EURASIP Journal on Embedded Systems, vol. 2007, Article ID 57234, 2007. – 10 p. (http://www.hindawi.com/GetArticle.spx?doi=10.1155/2007/57234)
6. O.M. Lozano, Kazuhiro Otsuka. Real-time Visual Tracker by Stream Processing. Journal of Signal Processing Systems for Signal, Image, and Video Technology, 2008 (http://www.springerlink.com/content/pk22n1632859082k/)
7. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления / В.В.Воеводин, Вл.В.Воеводин. – СПб.: БХВ-Петербург, 2002. – 608 с.: ил.
8. Ладиженський Ю.В. Моделирование статичного фона в видеопотоке с большим числом движущихся объектов // Ю.В.Ладиженський, А.О.Середа // Наукові праці Донецького національного технічного університету. Серія: Обчислювальна техніка та автоматизація. – 2009. – Вип. 16 (147) – С. 152 – 160.
9. Уоссермен Ф. Нейрокомпьютерная техника: теория и практика / Ф.Уоссермен. — М.: Мир, 1992.
10. High-Performance Computing with Windows Compute Cluster Server 2003 [електронний ресурс] (http://www.microsoft.com/india/windowsserver2003/ ccs/default.asp

Надійшла до редакції 30.03.2010