УДК 004.896

Сушко С.В.,
orcid.org/0000-0001-5107-5115,
Чемерис О.А., д.т.н.,
orcid.org/0000-0002-8134-5152

ІНТЕЛЕКТУАЛЬНИЙ МЕТОД РОЗБИТТЯ ІТЕРАЦІЙНОГО ПРОСТОРУ ОПЕРАТОРУ ЦИКЛІВ ПРОГРАМ

Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України

sergii.sushko@gmail.com
a.a.chemeris@gmail.com

Вступ

Впродовж розвитку обчислювальної техніки апаратне та програмне забезпечення постійно вдосконалюється. Водночас, з екстенсивним розвитком застосовується і інтенсивний розвиток, а саме значна увага приділяється більш ефективному використанню наявних ресурсів. Наприклад, більш ефективне використання ресурсів центру обробки даних може суттєвим чином вплинути на енергоспоживання, тим самим на кількість електроенергії необхідної для системи кондиціонування і, як наслідок, на операційні витрати. Ефективніше використання мобільних пристроїв дозволяє довше працювати без підзарядки та застосовувати апаратні пристрої з кращими обчислювальними можливостями. Ефективне використання вбудованих систем дозволяє розширювати функціональні та сервісні можливості пристроїв.

Ефективність обчислень це багатофакторна характеристика. По-перше, це міра спроможності апаратного комплексу та компілятора реалізовувати програмний код високого рівня [1]. По-друге, ефективність обчислень може бути визначена, як доля реальної обчислювальної потужності відносно пікової обчислювальної потужності, що доступна для даного апаратного забезпечення. По-третє, ефективність обчислень опирається на ефективність алгоритмів, що використовуються. Вказані фактори стосуються усіх обчислювальних систем – персональних комп'ютерів, серверів [2-3], мобільних пристроїв [4], ПЛІС [5], контролерів, вбудованих систем [6] і т. і.

Ефективність обчислень розглядається також в контексті об'єднання обчислювальних засобів. Наприклад у [7] наводяться оцінки ефективності різноманітних варіантів комплексування обчислювальних засобів в системі с заданим складом апаратури. Автор виділяє такий показник як коефіцієнт зниження реальної продуктивності, що характеризує витрати продуктивності обчислювальних засобів на організацію сумісної роботи комп'ютерів або процесорів.

$$K_k = \frac{P}{\sum_{i=1}^{N} V_i}, \qquad (1)$$

де *Vi* – ефективна продуктивність і-го обчислювального засобу при індивідуальному функціонуванні і обчислюванню визначеного класу задач; P – те саме для системи в цілому; N – кількість комп'ютерів або процесорів.

Ефективність реалізації деякого алгоритму або класу алгоритмів на обчислювальній системі майже цілком визначається співвідношенням структури вибраного алгоритму і структури цієї системи. При практичному використанні обчислювальних систем гнучкість логічної структури алгоритму, її здатність до перетворення набувають особливо важливого значення.

Постановка задачі

Одним з підходів до створення ефективних обчислювальних систем високої швидкодії є процес розпаралелювання алгоритмів на частини, які виконуються одночасно, зменшуючи час виконання програми. Найбільш велика частина роботи в алгоритмах сконцентрована в циклічних частинах, які відображаються в програмах відповідними операторами. Саме обчислювальні цикли являють собою найбільш ефективне місце для оптимізації програм за часом виконання.

Надаємо далі декілька визначень, які необхідні для пояснень при постановці задачі розпаралелювання та оптимізації операторів циклів програм.

Визначення. Між двома операторами S1 і S2 існує залежність, якщо вони обидва звертаються до однієї і тієї ж чарунки пам'яті, і, принаймні, одне з цих звернень є запис [1].

Відомо, що в програмах виділяють чотири типи залежностей, пов'язаних з доступом до одних і тих же чарунок пам'яті.

Розглянемо два оператора S_i і S_j ($S_i \prec S_j$) і відповідно до [8] визначимо типи залежностей в програмі. Тут використовується знак '$\prec$', який визначає лексикографічний порядок операторів S_i і S_j, тобто оператор S_i передує в програмі оператору S_j. Обидві команди виконують операції читання-запис з пам'яттю комп'ютера.

Визначення. Лексикографічний порядок (впорядкована послідовність) n -мірного декартового добутку $A^n = A \otimes A \otimes ... \otimes A$ визначається наступним чином: якщо $a=(a_1, a_2, \ldots, a_n)$ та $b = (b_1, b_2, \ldots, b_n)$, то $a \prec b$, якщо $a_1<b_1$ або $a_1=b_1$, $a_2=b_2$, …, $a_k = b_k$, та $a_{k+1}<b_{k+1}$, де $1 \le k \le n - 1$.

Між операторами програм існують такі типи залежностей:

Залежність за даними *(Data – Flow Dependence)* між операторами S_i та S_j ($S_i \prec S_j$) виникає в тому випадку, коли оператор програми S_j зчитує дані, які обчислюються в S_i.

$$S_i : X = F_1(In(S_i));$$
$$S_j : Y = F_2(In(S_j)); X \in In(S_j) \;\&\; j \ge i+1;$$

де $In(S)$ – множина вхідних змінних оператора S. Залежність за даними визначається так званим порядком «запис перед читанням» [9]. Відповідно до позначень, які використовуються в [10], напишемо дану залежність як $S_i \delta S_j$, що інтерпретується як «читання змінної в S_j залежить від запису цієї змінної в S_i».

Залежність по виходу (*Output Dependence*) двох операторів S_i та S_j ($S_i \prec S_j$) виникає в тому випадку, коли запис однієї і тієї ж змінної здійснюється в обох операторах, тобто S_i: $X = F_1(In(S_i))$;

$$S_j : X = F_2(In(S_j)); j \ge i+1.$$

Ця залежність запобігає від використання при виконанні будь-якого поточного оператора неправильних значень X. Позначається залежність по виходу як $S_i \delta^0 S_j$.

Антизалежність (*Antidependece)* між операторами у програмі S_i и S_j ($S_i \prec S_j$) виникає в тому випадку, якщо читання змінної з пам'яті проводиться раніше, ніж її запис в пам'ять комп'ютера, тобто S_i : $X = F_1(In(S_i)); Y \in In(S_i)$

$$S_j : Y = F_2(In(S_j)); j \ge i+1.$$

Антизалежність запобігає зміну Si до виконання Sj, що може привести до невірних значень. Записуємо цю залежність у вигляді $S_i \delta^{-1} S_j$.

Залежність четвертого типу є залежність по управлінню. Вона визначається операторами програми, які змінюють порядок виконання операторів в залежності від деякої умови. Так, наприклад, у фрагменті коду на мові С

```
S1:   if( x == 0 )
S2:           y = ToDo( arg );
```

Таким чином, в програмах визначені наведені вище типи залежностей між операторами. Ці залежності характерні для будь-яких ділянок програм.

Розглянемо вкладені обчислювальні циклі, кожен з яких визначений своєю індексною змінною. Множина індексних змінних визначає індексний вектор вкладених циклів $I=(I_1,I_2,..,I_n)$.

Для будь-якого циклу, в якому індекс циклу I змінюється від значення L до U з

кроком *S*, номер ітерації *i* дорівнює значенню (*I* - *L* + *S*) / *S*, де *I* – це значення індексу для цієї ітерації [7].

Для вкладених *n* циклів вектор ітерації *I* для самого внутрішнього циклу є вектором, що містить ціле число ітерацій для кожного циклу в порядку вкладеності циклів. Іншими словами, номер ітерації багатовимірного вкладеного циклу визначається відповідно до форми $I = \{i_1, i_2 ...., i_n\}$, де i_k, $1 \le k \le n$, являє собою номер ітерації циклу для рівня вкладеності *k*.

Визначення. Ітераційний простір це множина всіх цілочисельних векторів $I = (I_1, I_2, ..., I_n)$, що задовольняють нерівності:

$$L_i \le x_i \le \mathrm{U}_i, i = 1..n, \qquad (2)$$

Нерівність (2) визначає межі циклу, що обмежують ітераційний простір опуклим багатогранником.

Таким чином, визначено модель представлення ітераційного простору, яка складається з обмежень, що визначають границі простору, множини вузлів, які відповідають ітераціям циклу, та множини залежностей між ітераціями.

Задачею розпаралелювання є розбиття ітераційного простору на окремі блоки, при тому маємо витримати за можливістю наступні умови: 1) відповідність до структури обчислювальної системи, наприклад, врахувати архітектуру і кількість процесорів; 2) забезпечити однакове навантаження на процесори системи; 3) мінімізувати зв'язки між блоками простору, забезпечуючи паралельне їх виконання.

Розглядаючи різноманіття підходів модифікації обчислювальних циклів, слід відзначити такий метод як розбиття циклу на блоки (tiling) та його модифікації. Даний метод вводить додаткові цикли, які розбивають весь ітераційний простір на невеликі блоки, обчислення відбувається спочатку по кожному блоку. Розмір блоку вибирається в кожному конкретному випадку індивідуально. Такий підхід покращує локальність даних, що виражається в більш ефективному використанні кеша і внутрішніх регістрів процесора, і це може призводити до зниження навантаження на шину пам'яті, прискоренню роботи обчислень, зниження енергоспоживання.

Сформулювати постановку задачі розбиття ітераційного простору на окремі блоки, вважаючи на множину існуючих методів розбиття, але коли не відомо, який призведе до найкращих результатів, можна таким чином.

В загальному вигляді, реалізація процесу розпаралелювання та оптимізації програмного забезпечення здійснюється за допомогою послідовності перетворень, алгоритмів, методів, що аналізують програму та змінюють її для отримання семантично еквівалентного варіанта, що більш ефективний з точки зору якогось набору цілей оптимізації. В [1] показано, що деякі проблеми оптимізації коду є NP-повними чи навіть такими, що не можуть бути розв'язаними. На практиці багато з них вирішуються евристичними методами, що дають результат за задовільний час обробки вихідного коду програми.

Перетворення програм здійснюють задля зменшення деякого цільового параметру. Нехай argmin це функція, що позначає значення аргумента при якому досягається мінімум функції (3):

$$\underset{x}{argmin}\, f(x) \in \{x \mid \forall y : f(y) \le f(x)\} \qquad (3)$$

Тоді розглядаючи розпаралелювання та оптимізацію програмного забезпечення як процес спрямованого застосування або перебору кінцевого числа методів з кінцевим набором параметрів можна описати формулою (4):

$$F_i = argmin(f(\overrightarrow{M_l}, \overrightarrow{P_k})) \qquad (4)$$

де F – цільова функція, якою може бути один або декілька параметрів, що потребують покращення – зменшення часу виконання, енергоспоживання, розміру пам'яті програм, даних тощо; M – набір можливих методів розпаралелювання і оптимізації; P – множина значень параметрів цих методів.

Вираз (4) означає, що для розбиття ітераційного простору потрібно використати декілька відповідних методів зі своїми оптимізаційними параметрами. Вибір методів може бути обраний заздалегідь

(визначений явно або встановлений за замовчуванням) або визначений в результаті оцінки коду за допомогою різноманітних метрик.

Відповідно, процес розбиття ітераційного простору на окремі блоки має складнощі через те, що вибір набору конкретних методів і їх конкретних параметрів невизначений заздалегідь і вимагає додаткових експериментів для перевірки ефективності та їх комбінацій. Через архітектурні особливості та обмеження підбір методів та їх параметрів унікальний для кожного практичного випадку.

Інтелектуальний метод розбиття циклів на окремі блоки

Як сказано було вище, методів розбиття ітераційного простору операторів циклів на блоки декілька і один з найпопулярніших є тайлинг (tiling). Але існують його модифікації, які працюють більш ефективно для конкретних видів циклів.

За основу проведення розробки, що пропонується, було взято типовий процес, який використано в програмному пакеті Pluto [11]. Його основні етапи наведено на рис.1. Цей пакет включає низку програмних модулів, що почергово спочатку створюють поліедральну модель на основі вихідного коду мов C/C++, потім власне сам пакет Pluto модифікує цю поліедральну модель згідно заданих методів розбиття на блоки та паралелізму, далі інші програмні модулі спрощують отриману модель не змінюючи лексикографічної послідовності, і на останньому етапі програмні модулі створюють вихідний код мовами програмування C або C++ по отриманій моделі.

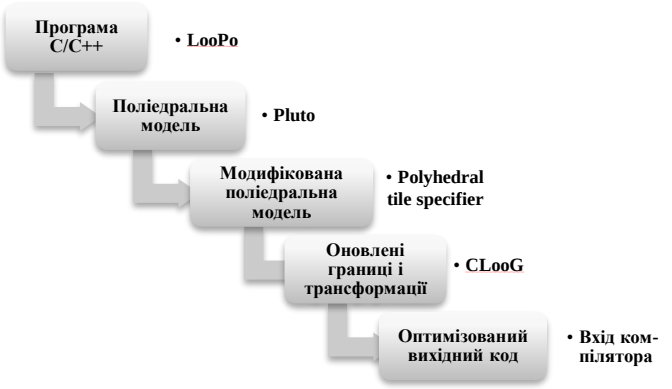

Рис. 1. Типовий підхід до розпаралелювання та оптимізації програм

Для ефективного розбиття ітераційного простору операторів циклів необхідно обрати відповідний метод, який дасть максимальну швидкість обчислень та знайти відповідні параметри цього методу. Для типової схеми на рис. 1 це залежить від ефективності реалізованих методів розбиття та досвіду програміста.

Тим не менш, автоматизація цього процесу привносить свої видатки щодо неефективності перетворювань. Тому потрібно шукати і розробляти свої підходи до реалізації розпаралелювання та оптимізації програм.

На етапі модифікації поліедральної моделі проходить процес вибору параметрів розбиття ітераційного простору. Експерименти, що наведено в [12] показують складну залежність ефективності обчислень від методу розбиття та розмірів блоків, на які розбито простір. Визначивши мінімум цієї функції, зможемо отримати метод та розміри блоків розбиття.

Авторами, для пошуку мінімуму цільової функції, запропоновано використати оптимізаційні методи роєвого інтелекту, а саме, метод рою часток [13]. Використовуючи дискретну версію методу, автори пропонують інтелектуальний метод розбиття ітераційного простору операторів циклу в програмах C/C++. Це дозволить зменшити час пошуку оптимального рішення при розпаралелюванні алгоритмів

для багатопроцесорних обчислювальних систем.

Структурна схема методу, що пропонується, наведена на рис.2.

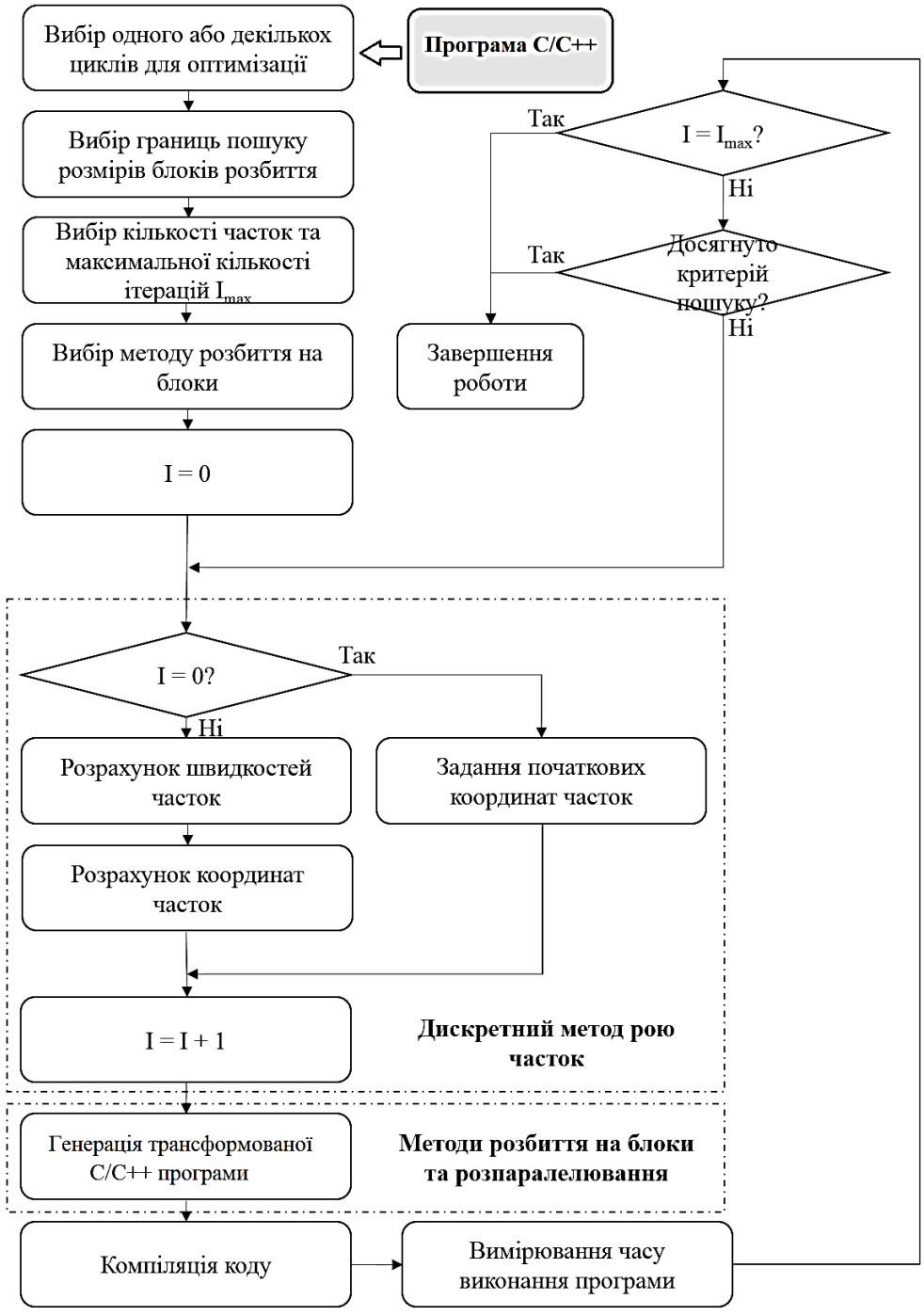

Рис. 2. Алгоритм інтелектуального розбиття ітераційного простору циклів з використанням методу рою часток

Алгоритм методу інтелектуального блочного розбиття, як зображено на рис.2, використовує дискретний метод рою часток для пошуку кращих розмірів блоків розбиття шляхом ітеративного підбору блоків та оцінкою ефективності підібраних блоків шляхом компіляції та вимірювання часу виконання комп'ютерної програми.

Такий алгоритм достатньо гнучкий і дозволяє використовувати різноманітні варіанти методу розбиття на блоки як окремо, так і з розпаралелюванням одночасно.

Авторами перевірено ефективність роботи методу на блоках розбиття прямокутної форми [14]. Значно зменшено час пошуку параметрів розбиття простору при розробці паралельного програмного забезпечення мікропроцесорних обчислювальних систем. Метод може бути також застосовано до методів розбиття простору, що використовують блоки іншої форми – трикутники, паралелограми, ромби тощо.

Але, тим не менш, на поточний момент метод має деякі обмеження, а саме, 1) обчислювальні цикли мають використовувати ключове слово for; 2) в середині обчислювальних циклів не має бути викликів функцій або застосування макросів; 3) в середині обчислювальних циклів заборонено використання вказівників - адресація даних має бути тільки через явні індекси масивів.

Висновки

Для прискорення підготовки програмного забезпечення обчислювальних систем з багатопроцесорною архітектурою, використання автоматизованих систем розпаралелювання та оптимізації програм є актуальною та перспективною задачею. Особливо слід підкреслити потенціальний ефект для мікропроцесорних систем управління, систем IoT, вбудованих систем, мобільних пристроїв, тощо.

З множини підходів до аналізу та розпаралелювання операторів циклів поліедральна модель представлення ітераційного простору оператору циклу, яка формується системою обмежень змінних циклу і являє собою N-мірний опуклий багатогранник, є перспективною до застосування для побудови ефективної системи розпаралелювання циклічних частин алгоритмів.

Аналіз використання методів розбиття ітераційного простору на частини показує складну залежність часу виконання програми (і відповідно, ефективності обчислень) від параметрів блоків розбиття. Використання методів оптимізації для пошуку мінімуму такої функції дозволяє пришвидшити процес розпаралелювання програми. Для цього запропоновано метод розбиття ітераційного простору, що базується на дискретному методі рою часток і надає субоптимальне або оптимальне рішення щодо вибору параметрів методу розбиття, що дозволяє покращувати швидкодію програм за рахунок кращого підбору параметрів.

Експерименти показують ефективність методу для 2-мірного випадку і, на окремих класах задач, можна отримати покращення швидкодії до 15 раз.

Розроблений метод оптимізації розбиття ітераційного простору операторів циклів програм, який перевірено на 2-мірному випадку розбиття на прямокутні частини, не має принципових обмежень щодо використання для інших видів розбиття (трикутниками, паралело-грамами, ромбами, тощо).

Література

1. *Воеводин В.В., Воеводин Вл.В.* Параллельные вычисления. – СПб.: БХВ-Петербург, 2002. – 608 с.

2. *Дружиніна О. О.* Підвищення ефективності функціонування веб-серверів з використанням технології прогнозування часових рядів на основі нейромереж / О.О. Дружиніна, Р.Н. Квєтний // Інформаційні технології та комп'ютерна інженерія. – 2013. – № 1. – С. 15-21.

3. *Луцкий Г.М.* Повышение эффективности кластеров на основе Infiniband / Г.М. Луцкий, И.С. Райзин // Вісник університету «Україна». Інформатика, обчислювальна техніка та кібернетика. – 2011. – № 8. – С. 133.

4. *Havinga, P., Smit, G.* Low power systems design techniques for mobile computers. Centre for Telematics and Information Technology University of Twente, Enschede (1997). ISSN 1381-3625

5. *Клименко І.А.* Методи та засоби підвищення ефективності обробки інформації в реконфігурованих комп'ютерних системах на базі ПЛІС: дис. Клименко Ірини Анатоліївни д-ра техн. наук: 05.13.05. – Київ, 2017. – 377 с.

6. *Борисова Н.В.* Ефективне управління ресурсами вбудованих систем для обчислювальної техніки реального часу / Н.В. Борисова, Л.В. Шабанова-Кушнаренко // Системи обробки інформації. – 2018. – № 1(152). – С. 87-93.

7. *Душин В.К.* Теоретические основы информационных процессов и систем/ В.К. Душин – М.: Дашков и Ко, 2003. – 348 с.

8. *Kennedy Ken* Optimizing Compilers for Modern Architectures – A dependence based approach / Ken Kennedy, Allen Randy // San Francisco, San Diego, New York: Morgan Kaufmann Publishers. – 2001.

9. *Huang T.-C.* Data dependence analysis for array references / T.-C. Huang, C.-M. Yang // Journal of Systems and Software. – 2000. – Vol. 52. – P. 55–65.

10. *Banerjee U.* Time and Parallel Processor Bounds for Fortran-like Loops / U. Banerjee et al. // IEEE Trans. on Computers. – 1979. – № 9. – P. 660-670.

11. *Uday Bondhugula, Albert Hartono, J. Ramanujam, Ponnuswamy Sadayappan* A practical automatic polyhedral parallelizer and locality optimizer // In Conference: PLDI '08: Proceedings of the 2008 ACM SIGPLAN conference on Programming language design and implementation, May 2008. – ACM SIGPLAN Notices 43(6).

12. *Sushko S.* Dependency between Tiles' Sizes and Program Execution Time. / Sushko S., Chemerys A. // in Proceedings: Reconfigurable Ubiquitous Computing (RUC-2018), 12 жовтня 2018, Дзівнув, Польща.

13. *M.A. El-Shorbagy, Aboul Ella Hassanien* Particle Swarm Optimization from Theory to Applications // International Journal of Rough Sets and Data Analysis. – Vol. 5. – Issue 2. – 2018. – P. 1-22.

14. *Chemeris A.* Usage of Discrete Particle Swarm Optimization Method for the Searching of Optimal Tile Size / A. Chemeris, S. Sushko // 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T-2019), October 8-11, 2019, Kyiv. – P. 202-206. ISBN: 978-1-7281-4183-1.

Сушко С.В., Чемерис О.А..

ІНТЕЛЕКТУАЛЬНИЙ МЕТОД РОЗБИТТЯ ІТЕРАЦІЙНОГО ПРОСТОРУ ОПЕРАТОРУ ЦИКЛІВ ПРОГРАМ

Стаття присвячена методам автоматичного розпаралелювання та оптимізації програмного забезпечення. Автори сфокусовані на розпаралелюванні циклічних частин алгоритмів, зокрема, методах розбиття ітераційного простору операторів циклів програм на мовах C/C++. Проблема швидкого вибору методу розбиття та визначення його параметрів є задачею актуальною і її рішення дає зменшення часу підготовки програмного забезпечення обчислювальних систем з багатопроцесорною архітектурою. Особливо це актуально для мікропроцесорних систем керування, систем IoT, мобільних пристроїв, систем Індустрії 4.0, тощо. Для побудови автоматизованої системи розпаралелювання програм авторами запропоновано використовувати дискретний метод рою часток як оптимізаційний метод, що дозволяє знайти локальний або глобальний мінімум часу виконання програм при різному характері залежності між розмірами блоків і часу виконання. У статті запропоновано підхід щодо оптимізації процесу розбиття ітераційного простору операторів циклів з використанням методів роєвого інтелекту. Розроблений метод оптимізації розбиття ітераційного простору операторів циклів програм, який перевірено на 2-мірному випадку розбиття на прямокутні частини, не має принципових обмежень щодо використання для інших видів розбиття (трикутниками, паралелограмами, ромбами, тощо).

***Ключові слова**: паралельні програми, розпаралелювання програм, розпаралелювання циклів, метод рою часток, тайлінг.*

Sushko S.V., Chemeris O.A.

INTELLECTUAL METHOD OF THE ITERATIVE SPACE PARTITIONING FOR PROGRAM LOOP OPERATORS

The article is devoted to the methods of automatic parallelization and software optimization. The authors focus on parallelizing the cyclic parts of algorithms, in particular, methods for splitting the iterative space of loop operators for C/C ++ programs. The problem of quickly choosing a partitioning method and determining its parameters is an urgent problem and its solution provides a reduction in the preparation time for software for computing systems with multiprocessor architecture. This is especially true for microprocessor control systems, IoT systems, mobile devices, Industry 4.0 systems, and so on. To build an automated system for parallelizing programs, the authors proposed to use the discrete particle swarm method as an optimization method that allows the one to find a local or global minimum of program execution time with a different nature of the relationship between block sizes and execution time. The article proposes an approach to optimizing the process of partitioning the iterative space of loop operators using the methods of Swarm Intellect. The developed method for optimizing the partitioning of the iterative space of program loop operators, which is tested in the 2-dimensional case, partitioning into rectangular parts, has no fundamental restrictions on its use for other types of partitioning (triangles, parallelograms, rhombuses, etc.).

***Keywords**: parallel programs, program parallelization, loop parallelization, particle swarm optimization, tiling.*