УДК 681.3.06

Демчик Валерій,
Корочкін Олександр

ЗАСТОСУВАННЯ ДРІБНОЗЕРНИСТОГО ПАРАЛЕЛІЗМУ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ПАРАЛЕЛЬНИХ ТА РОЗПОДІЛЕНИХ ОБЧИСЛЕНЬ

APPLICATION OF FINE-GRAINED PARALLELISM FOR INCREASING THE EFFICIENCY OF PARALLEL AND DISTRIBUTED COMPUTING

В статті наводяться результати досліджень можливості підвищення ефективності обчислень в багатоядерних комп'ютерних системах за рахунок застосування дрібнозернистого паралелізму як окремо, так і в складі середньозернистого паралелізму.

Ключові слова: ядро, потік, паралелізм, зернистість, fork-join.
Рис.: 4. Табл.: 2. Бібл.: 5.

The paper deals with the issues of increasing the efficiency of computations in multi-core computer systems due to the application of fine-grained parallelism, separately or as part of middle-grained parallelism.

Key words: core, thread, parallelism, granularity, fork-join.
Fig.: 4. Tabl.: 2. Bibl.: 5.

Актуальність теми дослідження. На даному етапі розвитку обчислювальної техніки, коли питання жорсткої обмеженості доступних апаратних ресурсів не стоїть настільки гостро, одним з найважливіших показників, який визначає ефективність роботи конкретної обчислювальної системи, є час, який буде витрачено даною системою на рішення нею конкретної задачі. Мінімізація часових затрат на обчислення є важливим питанням, до якого спрямований значний інтерес як наукового товариства, так і бізнесу. Одним з шляхів, який дозволяє досягти значних скорочень часових затрат є організація паралельних та розподілених обчислень. І одними з ключових питань, які вирішуються при цьому, є вибір зернистості паралелізму, оскільки застосування паралелізму тієї чи іншої зернистості може виказати значний вплив на ефективність різноманітних обчислень, та вибір засобів організації паралелізму.

Постановка проблеми. В найпопулярніших засобах організації паралельних та розподілених обчислень, наряду з традиційними механізмами організації потоків, тільки відносно нещодавно з'явились надійні засоби підтримки дрібнозернистого паралелізму. Тому відсутні порівняльні дослідження проблеми вибору паралелізму того чи іншого ступеню зернистості для вирішення конкретних математичних задач, які потребують значних обчислень, а також засобів, які реалізують паралелізм.

Аналіз останніх досліджень і публікацій. Протягом останніх років з'являється все більше статей присвячених паралелізму різного ступеню зернистості. Проте підходи для застосування дрібнозернистого паралелізму все ще недостатньо вивчені.

Виділення недосліджених частин загальної проблеми. Дана стаття присвячена вивченню, порівнянню та аналізу питань ефективності застосування

паралелізму різного ступеню зернистості, організованого різними засобами при застосуванні його в багатоядерних комп'ютерних системах. Також проводиться дослідження ефективності власного запропонованого підходу комбінованого паралелізму для організації паралельних та розподілених обчислень, який ґрунтується на одночасному застосуванні (комбінуванні) середньо- та дрібнозернистого паралелізму.

Постановка завдання. Завданням є розробка пакету програм для матричних операцій з використанням бібліотеки OpenMP, мов C# та Java. Кожну операцію реалізувати у вигляді чотирьох програм П1-П4. В програмі П1 використати традиційний підхід, який базується на програмуванні декількох потоків та низькорівневих примітивів їх взаємодії (середньозернистий паралелізм), в програмі П2 – застосувати лише розпаралелювання обчислювальних циклів (дрібнозернистий паралелізм), в програмі П3 – організувати розпаралелювання на декілька потоків, в кожному з яких провести розпаралелювання обчислювальних циклів (комбінований паралелізм). Програма П4 повинна являти собою послідовну реалізацію, необхідну для розрахунку коефіцієнтів прискорення програм П1-П3. Провести тестування розроблених програм.

Аналіз засобів реалізації. Засоби реалізації дрібнозернистого паралелізму реалізовано в бібліотеці OpenMP та мовах Java і C# [1- 4].

Бібліотека OpenMP. Дрібнозернистий паралелізм представлений спеціальною директивою препроцесору #pragma omp parallel for**,** яка відноситься до директив розділення роботи (work-sharing directive). Такі директиви застосовуються не для паралельного виконання коду, а для логічного розподілу групи потоків, щоб реалізувати вказані конструкції керуючої логіки. Директива #pragma omp for повідомляє, що при виконанні циклу в паралельному режимі ітерації циклу повинні бути розподілені між групою потоків. Взаємодія потоків здійснюється через неявну бар'єрну синхронізацію в кінці кожного блоку #pragma omp for.

Мова Java. Розробникам пропонується надзвичайно гнучкі та потужні засоби реалізації дрібнозернистого паралелізму, в основу яких покладено так механізм «Fork-Join». За фактичну реалізацію цього механізму в Java обрано рекурсивний алгоритм, робота якого полягає в наступному:

1. Здійснюється перевірка можливості розподілу дій даного потоку на дві менші задачі;

2. Якщо перевірка успішна, то виконується розподіл (Fork), шляхом створення нових потоків для кожної нової задачі. В кожному новому потоку алгоритм починається заново. Потік, який виконав розподіл блокується до тих пір, поки обидва створені ним потоки не закінчать свою роботу, а після виконує остаточний збір результату;

3. Якщо перевірка не успішна (досягнуто межі так званого «зерна паралелізму»), то виконуються задані обчислення в цьому потоці, по закінченню яких відбувається з'єднання з породившим потоком (Join).

Таку реалізацію містять два класи: RecursiveAction та RecursiveTask. Головна різниця між ними полягає в тому, що коли необхідно порахувати якесь конкретне числове значення певної великої функції (наприклад суму елементів вектора) то зручно використовувати RecursiveTask, а у випадку загальних операцій, результатом роботи яких не є конкретне число, краще використати RecursiveAction.

Наслідуючи дані класи та описавши власне перевантаження методу compute розробник може налаштувати роботу під вирішення конкретної задачі, що є надзвичайно ефективним.

Мова C#. В мові C# також з'явилась можливість реалізації дрібнозернистого паралелізму. Весь необхідний для цього функціонал вмістив в себе статичний клас System.Threading.Tasks.Parallel, а саме три його основні методи Parallel.For(), Parallel.ForEach(), Parallel.Invoke() та різні їх перевантаження. Parallel.For та Parallel.ForEach забезпечують паралельне виконання циклів for та foreach відповідно. В основу кожного зі згаданих методів покладено механізм, аналогічний тому, який застосовується в мові Java. Але при цьому його значно спрощено шляхом застосування механізму делегування та використанням анонімних методів.

Комбінований паралелізм. Запропонований в роботі підхід базується на використанні в паралельної програмі двох видів паралелізму: середньозернистого і дрібнозернистого. При цьому паралельна програма включає набір традиційних потоків по кількості ядер БКС. Кожен з цих потоків додатково реалізує внутрішній (дрібнозернистий) паралелізм шляхом створення підпотоків за допомогою відповідних засобів типу Fork-Join. Початкова кількість традиційних потоків може зменшуватися з метою забезпечення дрібнозернистого паралелізму вільними процесорними ресурсами.

Результати тестування. Тестування програм проводилось на КС, оснащеною шестиядерним процесором AMD Phenom II та модулем ОЗУ об'ємом 4 ГБ типу DDR3. Програмне забезпечення: операційна система Windows 7, бібліотека OpenMP 3.1, віртульна машина JVM 1.8, .NET Framework 4.7.

В таблиці 1 наведені отримані результати тестування паралельних програм для операції множення матриць для різних значень N (розмірність матриць).

Таблиця 1

Результати тестування паралелізму різного ступеню зернистості

N	Час виконання (сек.)								
	Середня зернистість			Дрібна зернистість			Змішана зернистість		
	OpenMP	Java	C#	OpenMP	Java	C#	OpenMP	Java	C#
516	1.7	0.2	1.2	1.6	0.2	1.1	1.5	0.2	1.1
1032	13.3	3.3	9.8	12.8	2.9	9.8	12.4	3.1	9.5
1548	46.7	14.0	35.6	45.9	12.1	34.1	44.9	13.0	33.8
2064	111.7	38.5	79.8	110.8	33.5	78.1	108.1	35.5	74.8

На рисунках 1-3 зображено графіки, які демонструють виявлену залежність коефіцієнтів прискорення (Кп) від значень N, для всіх трьох рівнів паралелізму.

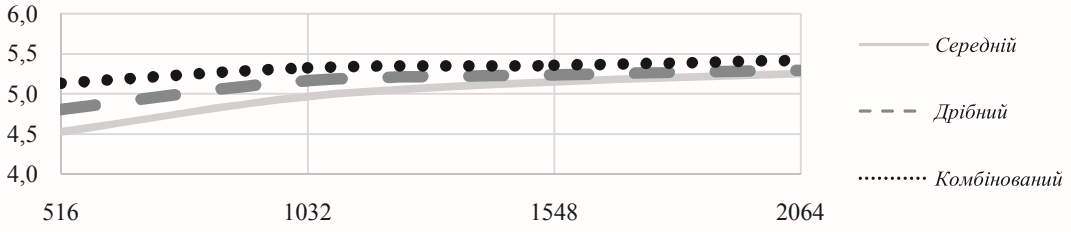

Рис.1. Графік залежності Кп від N. Бібліотека OpenMP

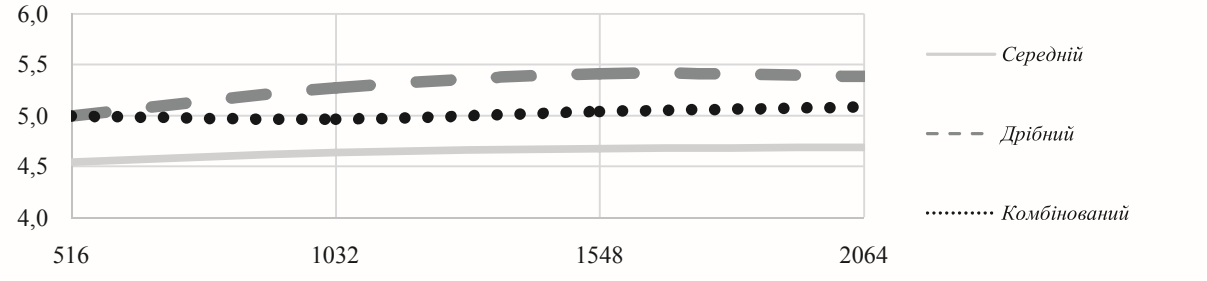

Рис.2. Графік залежності Кп від N. Мова Java

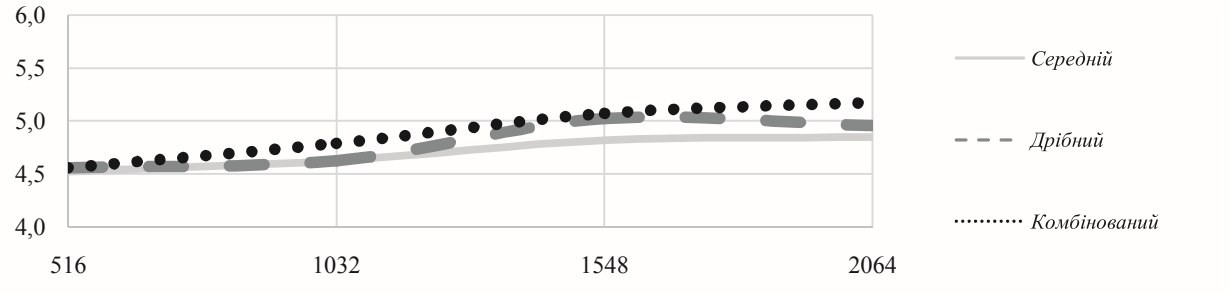

Рис.3. Графік залежності Кп від N. Мова C#

Окрім того, додатково було проведено окреме, більш детальне тестування дрібнозернистого паралелізму, з метою виявлення шляхів підвищення його ефективності. На цьому етапі тестувалися ті програми із розробленого пакету, які були написані з використанням лише дрібнозернистого паралелізму. Проводилися багаторазові заміри часу виконання операції множення двох матриць розмірністю 1500*1500 елементів. Перевірялася залежність часу виконання від кількості програмно-реалізованих потоків. На рисунку 4 наведено графік, який демонструє виявлену залежність часу роботи від кількості потоків (P). Детальні результати наведено в таблиці 2.

Таблиця 2

Результати тестування дрібнозернистого паралелізму

Кількість потоків	**Час виконання (сек.)**		
	C#	**OpenMP**	**Java**
2	71.309	133.944	29.67
3	55.1	101.41	24.12
4	42.099	77.393	16.894
5	37.202	60.942	14.167
6	34.222	47.532	13.101
10	34.113	45.551	12.24
12	32.841	45.117	12.485
15	31.405	44.37	11.195
20	30.404	44.33	9.899
25	30.511	43.101	9.636
30	28.329	41.9	9.211
50	27.683	40.88	9.5

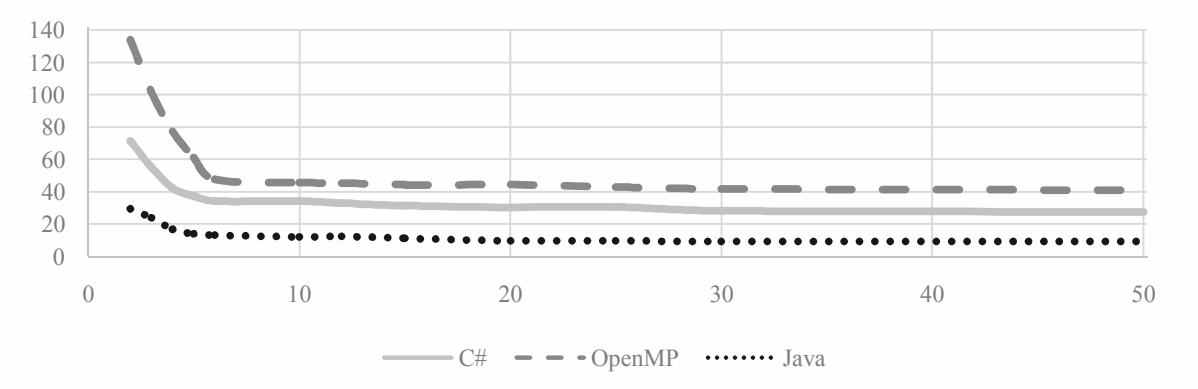

Рис. 4. Дрібнозернистий паралелізм.
Графік залежності часу роботи від кількості потоків

Висновки. Результати тестування показали ефективність багатоядерних комп'ютерних систем при реалізації рішення розглянутої математичної задачі засобами мов JavaiC# та бібліотеки OpenMP. При цьому можливе скорочення часу виконання програм при застосуванні паралелізму будь-якого ступеню зернистості (значення Кп лежать в межах 4.5-5.5). Найкращий результат по часу виконання програми отримано для мови Java.

Середньозернистий паралелізм показав достатню ефективність, але мав найгірший результат. При цьому прискорення, досягнуте засобами OpenMP постійно зростає зі збільшенням обсягу оброблюваних даних, а засобами Java та C# залишається приблизно на одному і тому ж рівні.

Застосування дрібнозернистого паралелізму також виявилося ефективним, в цьому випадку коефіцієнт прискорення стабільно зростає зі збільшенням обсягу оброблюваних даних. Найбільш ефективним цей тип паралелізму виявився в мові Java, завдяки потужному механізму підтримки fork-join.

Окрім того, проведене додаткове тестування дрібнозернистого паралелізму виявило спадаючу експоненціальну залежність між кількістю потоків, які дозволено створити програмі, та часом її роботи. Головною причиною цього є те, що чим більша кількість потоків, тим більше від них звернень до спільних ресурсів, що призводить до значних простоїв внаслідок вирішення задачі взаємного виключення. Оптимальна кількість потоків дрібнозернистого паралелізму, незалежно від того, якими засобами його було організовано, лежить в межах від 5 до 20.

Запропонований в роботі підхід, який ґрунтується на комбінованому паралелізмі, показав свою ефективність і дозволів збільшити Кп при використанні в мові C# та бібліотеці OpenMP. При цьому спостерігається зростання Кп при збільшенні обсягу оброблюваних даних, що є одним з найважливіших аргументів доцільності застосування даного підходу в БКС.

Можна припустити, що ефективність використання комбінованого паралелізму буде збільшуватися при зростанні кількості ядер в БКС, де :

- з'являться додаткові процесорні ресурси для його реалізації;
- можна зменшити розмір дрібної зернистості
- можна знайти оптимальне співвідношення між кількістю потоків і підпотоків.

Крім того, ефективність реалізації комбінованого паралелізму в OpenMP можна покращити шляхом більш ефективної реалізації потужного механізму управління підпотоками, що закладений в бібліотеці, аналогічно тому, як це зроблено в моделі Fork-Join.

Таким чином, можна стверджувати що застосування комбінованого паралелізму в більшості випадків є ефективним підходом до реалізації об'ємних паралельних обчислень на багатоядерних комп'ютерних системах.

Список посилань

1. Lea, Doug. A Java Fork/Join Framework, In Proceedings of ACM Java Grande 2000 Conference (San Francisco, California, June 3-5, 2000)

2. Цилькер Б.Я. Организация ЭВМ и систем: учебник для вузов / С.А. Орлов, Б.Я. Цилькер. - СПб.: Питер, 2011. - 688 с.

3. Жуков І.А., Корочкін О.В. Паралельні та розподілені обчислення: Навч. посібник [Текст]. – К.: Корнійчук, 2005. – 226 с. – ISBN 996-7599-36-1.

4. Канг Су Гэтлин, Пит Айсенси. OpenMP и C++. [Електронний ресурс]. – Режим доступу: https://msdn.microsoft.com/ru-ru/library/dd335940.aspx

5. Julien Ponge.Fork and Join: Java Can Excel at Painless Parallel Programming Too![Електронний ресурс]. — Режим доступу: http://www.oracle.com/technetwork/articles/java/fork-join-422606.html— Загл. з екрану.

ДОВІДКА ПРО АВТОРІВ

Демчик Валерій Валентинович – студент, кафедра обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».

Demchyk Valerii –student, Department of Computer Engineering, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute".

E-mail: kirintor830@gmail.com

Корочкін Олександр Володимирович – доцент, кандидат технічних наук, кафедра обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».

KorochkinAleksandr– docent, candidate of Technical Sciences, Department of Computer Engineering, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute".

E-mail: avcora@kpi.ua

Demchyk Valerii, Korochkin Aleksandr

APPLICATION OF FINE-GRAINED PARALLELISM FOR INCREASING THE EFFICIENCY OF PARALLEL AND DISTRIBUTED COMPUTING

Relevance of the research. Solving the issue of organizing efficient parallel and distributed computing is one of the key stages in designing modern software for multi-core computer systems. At the same time, one of the key points in the solution of this issue is the choice of the grains of the applied parallelism, which can have a significant impact on the time of work.

Target setting. The lack of comparative studies on selection of a parallelism degree of granularity to solve specific mathematical problems that require significant computation, and tools that implement parallelism.

Actual scientific researches and issues analysis. In recent years, there are more articles devoted to the parallelism of varying degrees of graininess. However, approaches to apply fine-grained parallelism is still not well understood.

Uninvestigated parts of general matters defining. This article is devoted to the comparison and analysis of the issues of the efficiency of the application of parallelism of different degrees of grains, organized by various means when applied to multicore computer systems. The research of the effectiveness of the proposed proposed approach of combined parallelism is also being conducted.

The research objective. The task is to develop a package of programs for matrix operations using the OpenMP library, the C # languages and Java, to implement each operation in several variants, using a parallelism of varying degrees of grainy. Conduct testing of developed programs.

The statement of basic materials. The comparative characteristic of the main means of organizing parallelism of different degree of grains is carried out. The description of its own proposed type of parallelism is given. Testing of programs for parallel computer systems with application of different granularity of parallelism is carried out.

Conclusions. The application of fine-grained parallelism proved to be the most effective approach to the organization of parallel computing. The approach suggested in the work has shown its efficiency and permissions to increase the acceleration factor. At the same time there is an increase in speedup coefficient with an increase in the volume of processed data, which is one of the most important arguments of the expediency of this approach in multi-core systems.

Key words: core, thread, parallelism, granularity, fork-join.