Курс «Паралельні алгоритми: побудова та аналіз».

Лекція 1. ***Вступ до курсу «Паралельні алгоритми: побудова та аналіз».***

I. Огляд курсу, що пропонується.

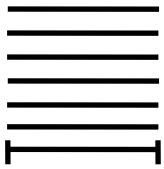
II. План лекції.

1. Поняття паралельного алгоритму.
2. Автономні та зв'язані гілки паралельного алгоритму.
3. Синхронна та асинхронна схеми обчислень.
4. Оцінювання ступеня паралелізму алгоритму. Середній ступінь паралелізму.
5. Прискорення та ефективність паралельного алгоритму.
6. Паралелізм і конвеєризація.
7. Рівні розпаралелювання.

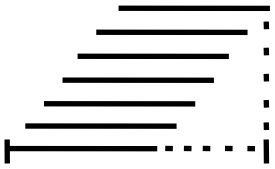
1. ***Паралельний алгоритм*** розглядаємо, як деяку сукупність гілок, які виконуються одночасно. Кожна гілка – це сукупність певних кроків (операторів, дій тощо).

Розглянемо деякі різновиди паралельних гілок:

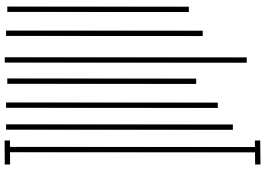
а) гілки приблизно одинакової довжини (найкращий варіант);



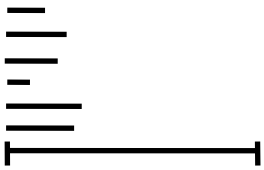
б) гілки приблизно однакової довжини, але зсунуті між собою; тут присутні елементи синхронізації, яка може здійснюватися програмно або апаратно;



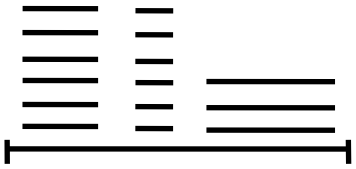
в) гілки різної довжини, але паралельність їх є суттєвою; час роботи алгоритму дорівнює часу виконання найдовшої гілки;



г) гілки різної довжини, але паралельність є несуттєвою;



д) паралельно-послідовна реалізація алгоритму;



е) отже, можливі різні варіанти зображення паралельних гілок алгоритму обчислень.

2. ***Автономні гілки*** паралельного алгоритму не обмінюються інформацією між собою. Такі алгоритми можуть бути реалізовані як на системах зі спільною (багатоядерні комп'ютери), так і розподіленою (кластерні обчислювальні системи) пам'яттю.

Зв'язані гілки паралельного алгоритму обмінюються між собою інформацією. Таких обмінів може бути мало та багато. Тому можливі різні варіанти реалізації таких алгоритмів. Якщо обмінів між гілками є багато, тобто гілки є сильнозв'язаними, то такі алгоритми найкраще реалізувати на обчислювальних системах зі спільною пам'яттю. Якщо ж між гілками є поодинокі обміни, тобто гілки є слабозв'язаними, то в багатьох випадках такі алгоритми можна реалізувати як на системах зі спільною, так і розподіленою пам'яттю (в цьому випадку виникає потреба в дослідженні прискорення паралельного алгоритму).

3. ***Синхронна схема обчислень*** – після виконання деякої кількості операцій в кожній з гілок їх робота синхронізується. Наприклад, синхронізація в кожній гілці може здійснюватися після виконання кожної операції.

Асинхронна схема обчислень – відсутні будь-які засоби синхронізації, а під час обчислень використовуються наявні в пам'яті поточні значення змінних.

Зазвичай у природі, суспільстві всі процеси відбуваються асинхронно…

4. Для оцінки паралелізму (його кількості, тобто міри) алгоритму використовують такі величини, як ***ступінь паралелізму*** та ***середній ступінь паралелізму***.

Ступінь паралелізму алгоритму – це кількість операцій алгоритму, які можна виконати паралельно (цю величину застосовують зазвичай у разі ідеального паралельного алгоритму, наприклад, алгоритму додавання двох векторів; якщо кількість компонент векторів дорівнює m, то ступінь паралелізму такого алгоритму дорівнює теж m).

Середній ступінь паралелізму числового алгоритму – це відношення кількості операцій цього алгоритму до висоти його ярусно-паралельної форми

(ЯПФ). Ця величина застосовується стосовно паралельних алгоритмів, в яких для різних фрагментів обчислень можна паралельно виконати різну кількість операцій.

5. Для оцінювання паралельного алгоритму використовують такі величини, як

- ***прискорення***

та

- ***ефективність***

паралельного алгоритму.

Прискорення паралельного алгоритму обчислюється за формулою:

$$S_p = \frac{T_1}{T_p},$$

де T_1 – час виконання алгоритму на одному процесорі, тобто послідовно;
T_p – час виконання паралельного алгоритму на p процесорах.

Стосовно прискорення S_p паралельного алгоритму справджується нерівність:

$$1 < S_p \le p.$$

У більшості випадків на практиці $S_p < p$, до чого спонукає низка факторів:

- наявність значної кількості обмінів інформацією;
- робота з пам'яттю;
- наявність (використання) засобів синхронізації.

Однак, зустрічаються і випадки, коли $S_p > p$, тоді маємо так званий парадокс паралелізму, що насправді не є парадоксом, оскільки така ситуація має своє пояснення.

Ефективність E_p паралельного алгоритму обчислюється за формулою:

$$E_p = \frac{S_p}{p}.$$

Ця величина вказує на питоме прискорення, що припадає на кожен процесор і при цьому справджується нерівність:

$$0 < E_p \le 1.$$

Отже, E_p показує, наскільки завантажений корисною роботою кожен процесор під час реалізації паралельного алгоритму.

6. ***Паралелізм та конвеєризація.***

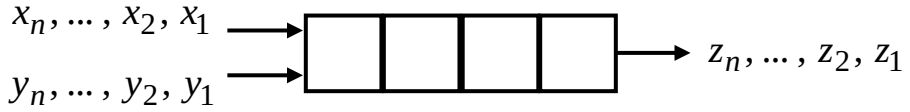
Ідея ***конвеєризації*** обчислень полягає у виділенні окремих етапів виконання деякої операції так, щоб кожен етап, виконавши свою роботу, передавав би

результат наступному етапу, одночасно приймаючи нову порцію вхідних даних.

Загальну кількість етапів конвеєра називатимемо ***довжиною конвеєра***.

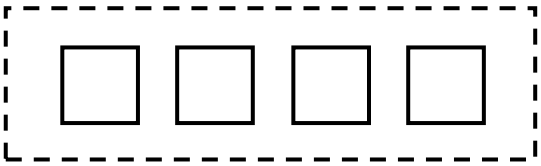
Для чого потрібна конвеєризація і чому її не можна замінити паралелізмом?

Конвеєр: $\vec{z} = \vec{x} + \vec{y}$.



Вектор $\vec{z}$ отримуємо на конвеєрі довжиною 4 за (n+3) такти.

Паралельна обчислювальна система:



Вектор $\vec{z}$ отримуємо на паралельній обчислювальній системі із чотирьох процесорів за n тактів. Кожен із процесорів повинен реалізувати всі етапи конвеєра, тобто у цьому випадку використовується більше обладнання і вартість обчислювального процесу зростає. Тому суто з економічного погляду більш прийнятним є використання конвеєрного пристрою.

7. Розглянемо деякі основні ***рівні розпаралелювання*** обчислень:

- рівень задач;
- рівень процедур;
- рівень виразів;
- рівень команд;
- рівень операцій;
- бітовий рівень (реалізований у кожному сучасному мікропроцесорі).

Наведені рівні розпаралелювання є умовними, для них не існує чіткої класифікації. Однак, вони дозволяють краще зрозуміти саму суть проблеми розпаралелювання обчислень та побудови паралельних алгоритмів.

Деякі висновки.

Отже, курс буде спрямовано на вивчення та дослідження паралельних алгоритмів розв'язання окремих задач, зокрема задач цифрової фільтрації даних, лінійної алгебри, а також деяких паралельних обчислювальних систем нетради-

ційної архітектури, наприклад, систолічних, квазісистолічних, нейромережевих, зі структурно-процедурною організацією обчислень, потоків даних тощо.

Список рекомендованих джерел для вивчення курсу.

1. Вальковський В. О., Яджак М. С. Проблеми подальшого розвитку та модифікації методу пірамід для розпаралелювання циклів // Математичні методи та фізико-механічні поля. – 2000. – **43**, № 1. – С. 68–75.
2. Жуков І.А., Курочкін О.В. Паралельні та розподілені обчислення. Навч. посіб. – К.: «Корнійчук», 2005. – 226 с.
3. Новотарський М.А., Нестеренко Б.Б. Штучні нейронні мережі: обчислення. – Київ: Інститут математики НАН України, 2004. – 408 с.
4. Рейтинговий список найпотужніших обчислювальних систем світу. – Режим доступу: www.top500.org.
5. Штейнберг Б.Я., Штейнберг О.Б. Преобразование программ – фундаментальная основа создания оптимизирующих распараллеливающих компиляторов // Программные системы: теория и приложения.– 2021.– **12**, № 1. – С. 21–113.
6. Яджак М.С. Квазісистолічні обчислювальні структури та їх застосування // Академический вестник. – 2007. – № 20. – С. 53–57.
7. http://eprints.library.odeku.edu.ua/695/1/RolshchikovVB_Distributed_Systems_Technology_And_Parallel_Computing_Module_1_KL_2018.pdf.