***Чередюк Роман Русланович,*** магістр\*

***Яковлєв Ігор Сергійович,*** магістр\*

***Яковлєва Інна Дмитрівна***, кандидат технічних наук, доцент\*

*\*Чернівецький національний університет*

*імені Юрія Федьковича, Чернівці*

*ORCID 0000-0002-8917-8686*

**БІБЛІОТЕКИ PYTHON ДЛЯ ОЦІНКИ ПРОДУКТИВНОСТІ ПАРАЛЕЛЬНОГО ВИКОНАННЯ ПЕРЕТВОРЕННЯ ФУР'Є НА БАГАТОЯДЕРНИХ ПРОЦЕСОРАХ**

Час виконання алгоритмів завжди є дуже важливим критерієм роботи програми. На даний час є багато методів розпаралелювання, що пришвидшують час виконання, але залишається питання, який метод розпаралелювання найкращий для того, чи іншого алгоритму. Тому актуальним є визначення найкращого методу за рахунок аналізу характеристик роботи програми.

Розпаралелювання задач – це одна із форм виконання коду, в ході якого створюються нові потоки його виконання. Такий підхід розподіляє процеси, які будуть орієнтовані на одну і ту саму задачу. Під час розпаралелювання задач, кожна із ниток виконує одні і ті самі вказівки для однакових або різних даних [1].

Для виконання даного типу обрахунків використовується багато технологій: кластерні обчислення, багатоядерні, багатопроцесорні, за допомогою відеокарт (GPGPU), віддалені розрахунки та інші [2-4].

Оскільки, розпаралелювання задач за допомогою ядер процесора та за допомогою відеокарти можна проводити на комп’ютері, в якому є одна відеокарта, процесор та інші комплектуючі, які потрібні для роботи самої системи.

Швидке перетворення Фур’є (ШПФ) – швидкий алгоритм дискретного перетворення Фур’є, який використовується для цифрової обробки сигналів для перетворення дискретних даних з часового діапазону в частотний. Алгоритм Кулі-Тьюкі – найпоширеніший алгоритм швидкого перетворення Фур’є у якому обчислення дискретного перетворення Фур’є із складністю O(N2) можна виконати за O(Log(N)) операцій. Саме мінімальна складність алгоритму стала визначником його популярності. Сфера використання ШПФ визначається скрізь, де передбачається робота з сигналами: аудіо обробка, радари, радіокомунікації та вс інші дані, які можуть передаватися за допомогою аналогового сигналу. Також це перетворення використовується для обробки, фільтрації та стиснення зображень [1,5].

Якщо ШПФ виконувати паралельно, то можна передавати, отримувати та обраховувати набагато більше інформації за один і той самий період часу.

Для мови програмування Python 3.8 розглянуто три найбільш популярні бібліотеки: pyFFTW, SciPy, NumPy.

pyFFTW – є обгорткою бібліотеки FFTW для мови програмування С++ та використовує інтерфейси бібліотек SciPy та NumPy, та має декілька переваг над ними, до прикладу підтримка типу clongdouble, якої немає у NumPy. Дана обгортка поки не підтримує Python версії 3.12 [6].

SciPy – бібліотека яка включає в собі алгебраїчні алгоритми, оптимізацію, інтегрування, інтерполяцію та, також, працює на базі бібліотеки NumPy, розширюючи та доповнюючи її функціонал [7].

NumPy – фундаментальна бібліотека для наукових вираховувань, яка використовує обгортки таких швидкісних мов як С++ та FORTRAN, має велику підтримку різних обчислень та роботи з векторами та масивами. Використовується також як для CPU так і для GPGPU обчислень [8].

Ініціалізація виконання алгоритму ШПФ для всіх трьох методів є схожею. Для SciPy код ініціалізації є scipy.fft.fft(x), для NumPy – numpy.fft.fft(x), а для pyFFTW – в залежності від того, який інтерфейс потрібно застосувати, використовується pyfftw.interfaces.numpy\_fft.fft(x), у випадку застосування інтерфейсу NumPy та pyfftw.interfaces.scipy\_fft.fft(x) – для SciPy.

Для вхідного масиву N = 220  найкращий середній час виконання у SciPy дорівнює 0.44309 c, на другому місці бібліотека pyFFTW з використанням інтерфейсу NumPy - 0.77423 c, третє – NumPy 0.77423 c та четверте pyFFTW з використанням інтерфейсу SciPy - 0.78791 c відповідно (рис 1).

Використання оперативної пам’яті після виконання алгоритму є однаковим - 8.388608 Мбайт.



Рисунок 1 – Середній час виконання

 На основі проведеного дослідження зроблено висновок, що для ефективного виконання ШПФ за допомогою ядер процесора краще використовувати бібліотеку SciPy.

**Література**

1. Introduction to algorithms fourth edition / Thomas H. Cormen et al. Cambridge, Massachusetts London, England: The MIT Press, 2022. 312p.
2. Методи кластерних обчислень / С. Д. Погорілий та ін. Київ : Київ. ун-т, 2013. 415с.
3. Matt Haig 3 Books Collection Set / Peter Pacheco та ін. Penguin Life/HarperAvenue Ltd, 2021.
4. Eberly D. H. GPGPU Programming for Games and Science. A K Peters/CRC Press, 2014.
5. Мельник А. О., Яковлєва І. Д. Подання та структурний аналіз паралельних алгоритмів : навчальний посіб. Львів: Магнолія 2024. 100 c.
6. pyFFTW. *PyPI*. URL: <https://pypi.org/project/pyFFTW/> (дата звернення: 04.07.2024).
7. SciPy -. *SciPy -*. URL: <https://scipy.org/> (дата звернення: 04.07.2024).
8. NumPy -. *NumPy -*. URL: <https://numpy.org/> (дата звернення: 04.07.2024).