Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра Обчислювальної Техніки

Лабораторна робота №2

з дисципліни "Проектування складних систем"

Тема: "Дослідження технології апаратного пропуску замків

(Intel’s Hardware Lock Elision)"

Виконав: Перевірив:

студент групи ІП-93 Долголенко О. М.

Домінський В.О.

Київ 2023

**Зміст**

[Мета: 3](#_Toc128516692)

[Вихідні дані: 3](#_Toc128516693)

[Хід роботи 4](#_Toc128516694)

[Висновок: 11](#_Toc128516695)

[Посилання: 12](#_Toc128516696)

# Мета:

Для виконання цієї роботи потрібен доступ до мікропроцесора Intel, що підтримує технологію TSX-NI. Створіть потоки з використанням інтерфейсу Callable з пакету java.util.concurrent (синхронізація потоків за допомогою інтерфейсу Lock із пакета java.util.concurrent.lock). Дослідіть швидкість виконання Вашої паралельної програми на цьому мікропроцесорі. Після цього відключіть Intel® TSX-NI за допомогою наступного налагоджування реєстру (для Windows 10): reg add "HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Kernel" /v DisableTsx /t REG\_DWORD /d 1 /f. Перегрузіть комп’ютер, щоб ці зміни мали ефект та знову дослідіть швидкість виконання варіантів Вашої паралельної програми. Побудуйте порівняльні графіки й зробіть висновки по роботі. Для наступного включення технології Intel® TSX-NI, анологічно виконайте: reg add "HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Kernel" /v DisableTsx /t REG\_DWORD /d 0 /f.

# Вихідні дані:

Варіант – 12

Функції:

* А=В\*МС+D\*MZ+E\*MM
* MG=min(D+E)\*MM\*MT-MZ\*ME

Процесор з технологією Intel® TSX-NI - Intel(R) Core(TM) i7-9700 CPU @ 3.00GHz

## Хід роботи

За основу для даної лабораторної роботи візьмемо минулу, але зробимо такі зміни:

1. Зробимо потоки з інтерфейсом callable. У java це java.util.concurrent, а в python - concurrent.futures. Додатково використаємо ThreadPoolExecutor, котрий може об’єднувати потоки в групи:

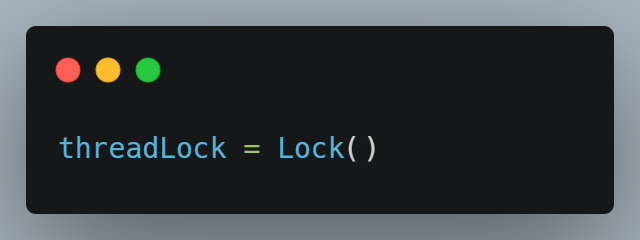


У даному випадку Ми створюємо ThreadPoolExecutor, де робимо 4 future, кожен з яких представляє кінцевий результат асинхронної роботи та відповідає за власну частину виразу.

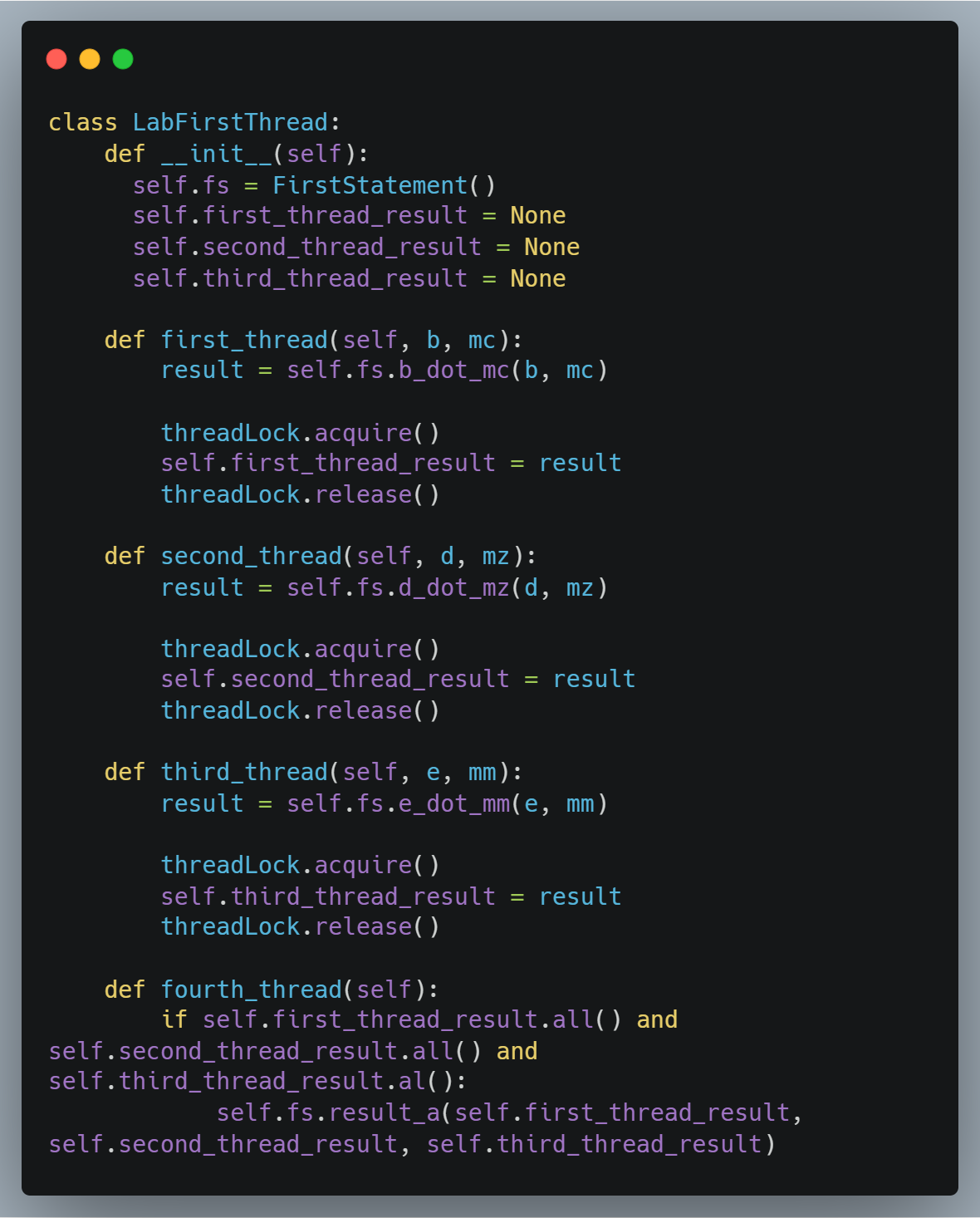
За допомогою методу submit Ми плануємо виклик future передаючи лямбда вирази методів, котрі обчислюють функцію відповідно до варіанту роботи.

Як видно з малюнку, то спочатку Ми запускаємо перші 3, а вже потім – останній – 4-ий future, котрий фіналізує Наш результат, так як він залежить від усіх інших.

1. Зробити записи до глобальних змінних синхронними. Для цього використаємо Lock:

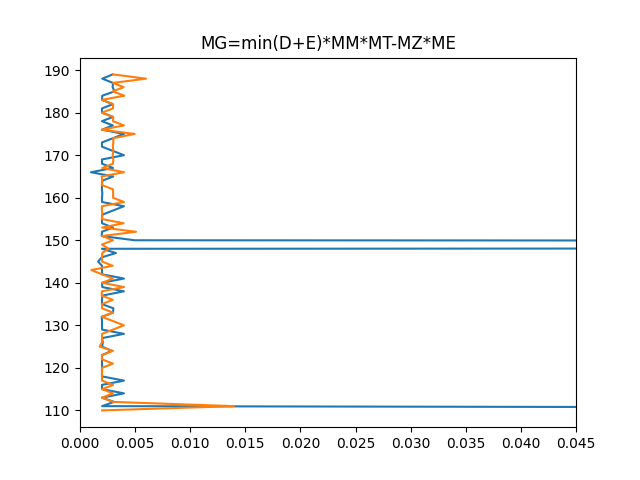
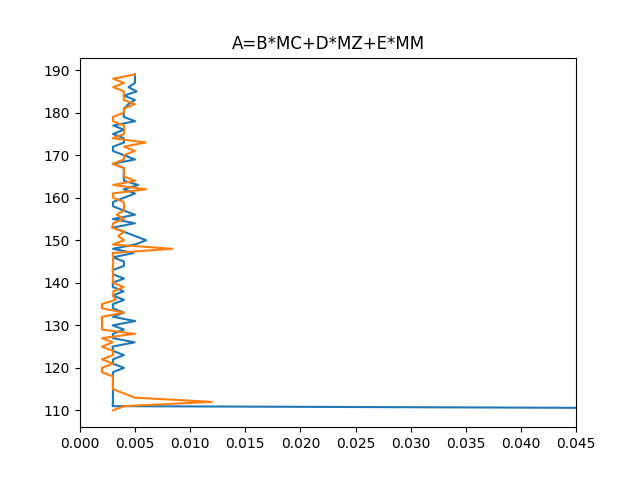


Та будемо працювати з ним при кожному записі:



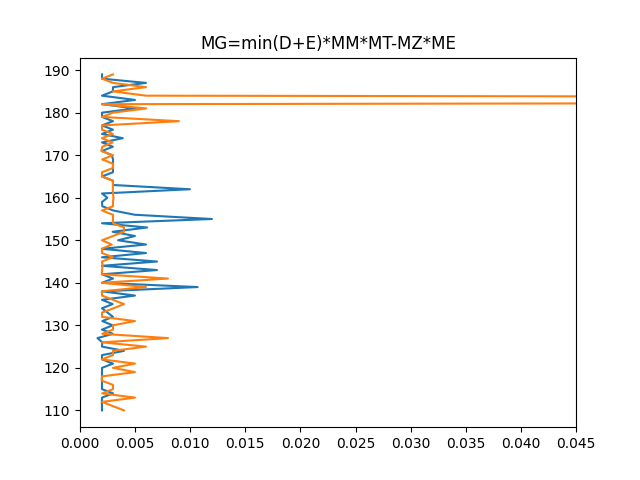
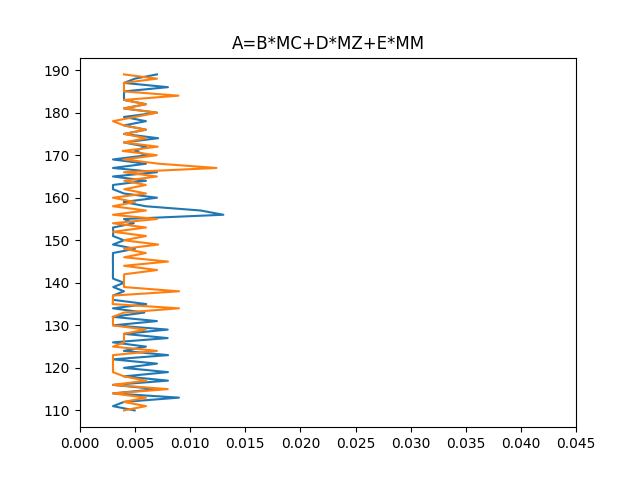
Метод acquire потрібен для закриття статусу Нашого локу, а release – навпаки – для відкриття. Таким чином робота зі змінними не буде відбуватись одночасно.

Тепер давайте запустимо цей код на персональному комп’ютері, де є процесор з підтримкою технології Intel® TSX-NI, але поки що активовувати її не будемо:



Тобто значення тримається в районі 0.0025 - 0.0037 і, в основному, має різницю приблизно 0.001

А зараз запустимо Лабораторну роботу з увімкненою технологією Intel® TSX-NI:



З графіків можна дійти до висновку, що коливання значень стало значно більшим, ніж було: з 0.0034 до 0.007

# Висновок:

На жаль пророблена робота не дала бажаних результатів, а саме пришвидшення обрахування всіх виразів, а навпаки – зробила тільки гірше. Можливо було не зовсім коректно проведено синхронізацію; можливо це пов’язано з несправедливим розподілом задач (деякі потоки роблять більш складні операції, ніж інші) - точно Я сказати в даному випадку не можу.

У ході виконання роботи Я створив 2 варіанти багатопотокової програми:

1. Перший – мав в собі принципи локалізації - при цьому як скалярні так і векторні змінні не повинні розділюватися між потоками – механізм когерентності кешів MESIF при цьому працювати не буде
2. У другий ж навпаки – було використано глобальні змінні для скалярів та матриць за допомогою механізмів синхронізації - пошук актуального значення змінної при цьому буде відбуватися, відповідно до механізму когерентності кешів MESIF, у кешах L1D всіх ядер мікропроцесора

Побудував кілька графіків, котрі порівнюють обидві варіації та зробив невтішні висновки щодо швидкодії багатопотокового методу.

Додатково створив запис значень в окремі файли – excel та txt

# Посилання:

1. Робота на Google Colab – [посилання](https://colab.research.google.com/drive/1rwjOYMOAeb3BiBhhC4iHz5a8dASqY9Gk?usp=sharing)
2. Робота на GitHub – [посилання](https://github.com/VsIG-official/Designing-Complex-Systems/tree/master/Lab1)