Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра Обчислювальної Техніки

Лабораторна робота №3

з дисципліни "Проектування складних систем"

Тема: "Дослідження технології апаратної реалізації транзакційної пам'яті"

Виконав: Перевірив:

студент групи ІП-93 Долголенко О. М.

Домінський В.О.

Київ 2023

**Зміст**

[Мета: 3](#_Toc129728561)

[Вихідні дані: 3](#_Toc129728562)

[Хід роботи 4](#_Toc129728563)

[Висновок: 8](#_Toc129728564)

[Посилання: 9](#_Toc129728565)

# Мета:

Дослідження технології апаратної реалізації транзакційної пам'яті. Для виконання цієї роботи потрібен доступ до мікропроцесора Intel, що підтримує технологію TSX-NI. Розгорніть на ньому якусь СУБД. Організуйте доступ до СУБД за допомогою спеціального API, наприклад, javax.sql. Розробіть тестову програму на Java для роботи з СУБД і дослідіть продуктивність своєї СУБД при використанні hardware transactional memory. Після цього відключіть на своєму мікропроцесорі підримку технології TSX-NI (див. виконання другої лабораторної роботи”) і на цій же апаратурі розгорніть ту ж саму СУБД і Java (деякі бібліотеки при цьому будуть іншими). Дослідіть продуктивність цього варіанту СУБД на тій же тестовій програмі при використанні software transactional memory. Приведіть порівняльні графіки й зробіть висновки по роботі.

# Вихідні дані:

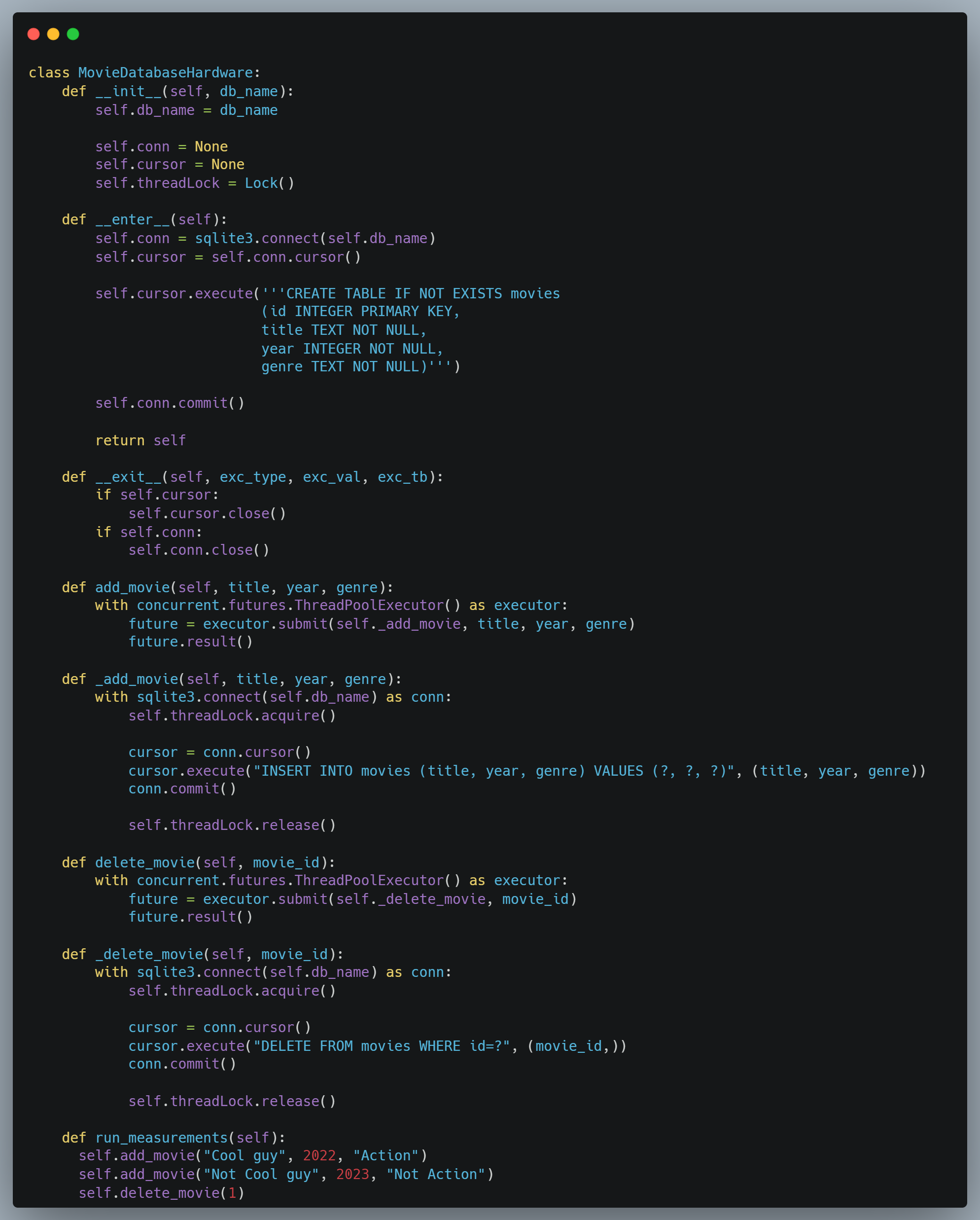
Процесор з технологією Intel® TSX-NI - Intel(R) Core(TM) i7-9700 CPU @ 3.00GHz

## Хід роботи

Для початку потрібно створити базу даних та підключитись до неї за допомогою субд. Мій вибір пав на Sqlite, так як вона є однією з найпростіших у використанні та зберіганні.

Для реалізації обох видів пам’яті у Java для Hardware є функціонал локів, а для software – atomic операції. Тому треба знайти подібну реалізацію на Python. Для першого варіанту усе просто – використовуємо все те, що було у минулій лабораторній, а от для другого розкажу трішки згодом.

Першим кроком буде створення самої БД та робота з нею:



Як видно з картинки Я маю клас для бази даних фільмів, де є такі операції:

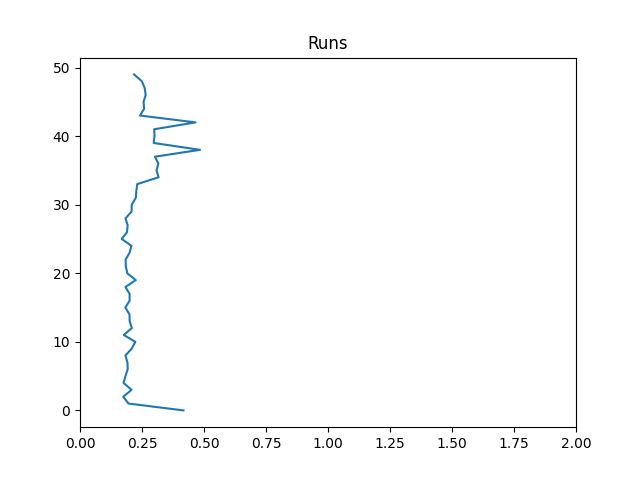
1. Створення власне самої БД
2. Додавання фільму
3. Видалення фільму
4. Запуск невеличкого тестування

У публічних методах до future записуються операції з приватних методів, котрі вже в Свою чергу напряму взаємодіять з базою даних – саме в них і буде прописано функціонал для обох варіантів роботи. У даному випадку – це локи.

Тепер давайте проведемо тестування Нашого коду з увімкненою технологією TSX-NI:



І отримаємо такий результат:



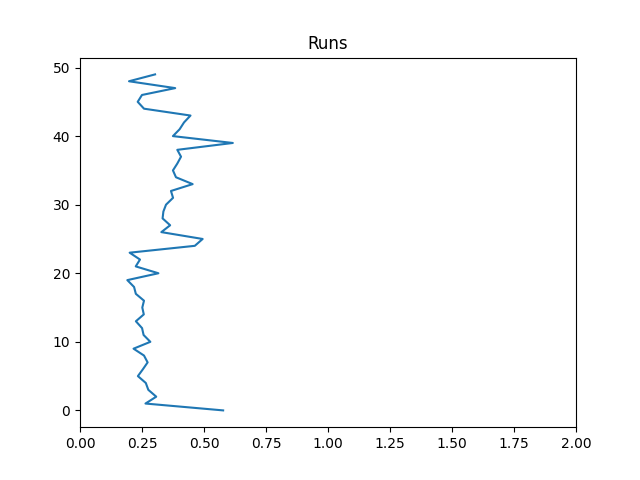
Наступне завдання – переробити код на використання atomic та запустити на тій же машині, але без використання TSX-NI:



Для цього варіанту Ми замінили локи на декоратор @atomically та через специфіку даного функціоналу потрібно зробити додатково обгортку у вигляді методу.

Проте, на жаль, бібліотека stm, котра і має необхідну для Нас фічу, трішки застаріла, тому треба було лізти в її код та замінити одне ключове слово на більш сучасне, з яким не виникає проблем на нових версіях Python.

Нумо глянемо на результати запуску:



# Висновок:

Створивши та протестувавши два варіанти лабораторної Я отримав результати, котрі дуже схожі на ті, що були отримані в минулих роботах - при використанні технології TSX-NI графіки мають гірші тенденції руху.

Раніше Я писав, що подібна ситуація можлива через те, що технологія доволі стара та перестала розроблятись, проте, у коментарях до минулої роботи Я почув іншу теорію – такі результати виникають саме через python, що є доволі резонною причиною, так як у даній мові програмування не так просто працювати з паралелізмом й схожими поняттями.

У ході виконання роботи Я краще засвоїв роботу з Lock, ThreadPoolExecutor та future’ами, за допомогою яких створив варіант роботи з hardware transaction memory. Нагадав Собі, як працювати з базами даних: створювати, додавати та видаляти поля, тощо. Також дізнався про декоратор @atomically, за допомогою якого можна реалізовувати software transaction memory

# Посилання:

1. Робота на GitHub (разом з результатами роботи коду при увімкненій та вимкненій технології TSX-NI) – [посилання](https://github.com/VsIG-official/Designing-Complex-Systems/tree/master/Lab3)