МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

кафедра систем штучного інтелекту

ЗВІТ

про виконання курсової роботи

з курсу «Штучний інтелект в ігрових застосунках»
на тему «Паралельна обробка потоків даних у системах керування дорожнім рухом»

Виконав:

ст. групи КНСШ-11

Чорній Юрій Юрійович

Перевірила:

Гентош Л. І.

ЛЬВІВ – 2023

Зміст

Вступ

Системи керування дорожнім рухом відіграють життєво важливу роль в управлінні та оптимізації транспортних потоків у містах. Ефективна обробка потоків даних, що генеруються різними датчиками та пристроями, має вирішальне значення для ефективного прийняття рішень та управління в реальному часі в цих системах. Це дослідження фокусується на паралельній обробці потоків даних в системах управління дорожнім рухом з метою підвищення швидкості і масштабованості процесів аналізу даних і прийняття рішень. Використовуючи можливості паралельних обчислень, це дослідження спрямоване на вирішення зростаючих проблем, пов'язаних зі збільшенням інтенсивності руху, заторами та необхідністю реагування в реальному часі при управлінні міським рухом. Актуальність та новизна цього дослідження полягає в його потенціалі для підвищення ефективності та результативності систем управління дорожнім рухом, що призведе до покращення транспортного потоку, зменшення заторів та підвищення транспортної стійкості.

Згідно з [1] фреймворк SUMO (Simulation of Urban Mobility) привернув значну увагу до моделювання дорожнього руху, надаючи реалістичну і гнучку платформу для моделювання та аналізу транспортних потоків у містах. В статті [4] підкреслили важливість паралельного виконання у великомасштабному агентному моделюванні дорожнього руху, що забезпечує швидші обчислення та масштабованість. Методи паралельної обробки можуть ефективно розподіляти обчислювальне навантаження між декількома обчислювальними одиницями, такими як CPU або GPU. Робота [5] продемонструвала потенціал паралельного моделювання на GPU для великомасштабних міських транспортних потоків. Використовуючи обчислювальну потужність графічних процесорів, можна значно підвищити продуктивність моделювання, що дозволяє обробляти більш детальні та складні моделі руху в режимі реального часу.

В [6] запропонували алгоритм паралельного моделювання великомасштабних міських транспортних мереж з використанням графічних процесорів. Дослідження підкреслило переваги розпаралелювання на GPU з точки зору прискорення та обчислювальної ефективності порівняно з традиційними моделюваннями на базі CPU. Згідно з [7] дослідили методи прискорення на базі GPU для великомасштабних симуляцій дорожнього руху. Їхнє дослідження показало, що розпаралелювання на GPU може значно скоротити час симуляції та дозволити працювати з більш розгалуженими транспортними мережами. У [8] представили підхід на основі GPU для паралельного моделювання міського трафіку, продемонструвавши його потенціал для контролю та управління трафіком у реальному часі. Дослідження продемонструвало переваги розпаралелювання на GPU з точки зору швидкості та масштабованості.

В [11] зосередилися на паралельному моделюванні автомобільного руху на базі GPU з динамічною системою навігації по маршруту. Дослідження підкреслило потенціал розпаралелювання на GPU для прийняття рішень у реальному часі в системах керування дорожнім рухом. [12] запропонували підхід до мікросимуляції на основі CUDA для моделювання міської транспортної мережі з використанням розпаралелювання на GPU. Дослідження продемонструвало ефективність та результативність розпаралелювання на базі GPU для великомасштабних симуляцій дорожнього руху. В статті [13] розробили алгоритм планування траєкторії руху транспортних засобів у реальному часі з прискоренням на GPU для симуляції міського руху. Дослідження підкреслило переваги розпаралелювання на графічних процесорах у забезпеченні контролю та оптимізації руху в реальному часі.

Метою цього дослідження є вивчення застосування методів паралельної обробки, в аналізі потоків даних з систем керування дорожнім рухом. Дослідження спрямоване на оцінку продуктивності та масштабованості підходів паралельної обробки та їх впливу на прийняття рішень в реальному часі в системах керування дорожнім рухом.

Було висунуто наступні гіпотези:

Гіпотеза 1: Розпаралелювання на графічних процесорах може значно підвищити швидкість обробки та масштабованість потоків даних у системах керування дорожнім рухом, що дозволить приймати рішення в реальному часі. Ця гіпотеза пов'язана з перевагами, продемонстрованими в літературі щодо прискорення, яке досягається завдяки розпаралелюванню графічних процесорів (Rupprecht та ін., 2019 [4]; Chen, M., & Chen, J, 2017 [9]).

Гіпотеза 2: Методи паралельної обробки, включаючи розпаралелювання як CPU, так і GPU, можуть підвищити ефективність і результативність систем управління дорожнім рухом, дозволяючи проводити аналіз і приймати рішення в реальному часі. Ця гіпотеза підтверджується результатами досліджень, які демонструють переваги розпаралелювання у скороченні часу моделювання та роботі з більшими транспортними мережами (Lu and Aboudolas, 2018 [7]; Belalem and Bougourzi, 2020 [8]).

Сформулюємо постановку задачі:

- Дано:

● Симуляція системи керування дорожнім рухом, Simulation of Urban MObility (SUMO).

● Кількість доступних обчислювальних ресурсів для паралелізації обробки.

- Знайти:

● Оптимальний спосіб паралелізації системи керування дорожнім рухом для забезпечення оптимізації транспортних потоків реальному часі.

- Припущення:

● Кожен крок симуляції обробляється незалежно від інших кроків.

● Прогнозування руху трафіку здійснюється на основі визначення їх поточних позицій швидкостей, а також маршрутних цілей.

- Функція цілі:

● Мінімізувати час обробки кожного кроку симуляції з використанням Програмно-апаратної архітектури паралельних обчислень CUDA .

● Забезпечити точність моделювання, зокрема визначення кількості транспортних засобів, їх положення та статусів на кожному кроці симуляції.

- Обмеження:

● Доступні обчислювальні ресурси для паралелізації обробки.

● Повнота та доступність даних отриманих із симуляції автомобільних потоків.

● Обробка симуляції повинна відбуватись в реальному часі, тобто з мінімальною затримкою між кроками.

- Результат:

● Оптимальна конфігурація паралелізації віртуальної симуляції контролю трафіку, що забезпечує ефективне та точне прогнозування маршруту транспорту реальному часі.

Аналіз літературних джерел

Сучасний стан паралельної обробки потоків даних у системах керування дорожнім рухом характеризується значними досягненнями та багатообіцяючими результатами досліджень. Дослідники визнали важливість ефективної обробки даних і прийняття рішень в реальному часі в управлінні дорожнім рухом, що призвело до появи великої кількості літератури, в якій досліджуються методи паралельних обчислень, зокрема, розпаралелювання на графічних процесорах, в цій галузі.

Перші два джерела, а саме [16], [17], висвітлюють сучасний стан досліджень паралельної обробки потоків даних у системах керування дорожнім рухом. Ці дослідження підкреслюють важливість розпаралелювання для обробки великомасштабних симуляцій міського руху, підвищення швидкості обчислень та уможливлення прийняття рішень у реальному часі. Вони демонструють переваги використання технологій паралельних обчислень, таких як архітектури з декількома графічними процесорами та алгоритми оптимізації на базі CUDA, для ефективної обробки даних та управління дорожнім рухом. Однак, одним із загальних обмежень, згаданих у цих дослідженнях, є необхідність подальших досліджень для оптимізації паралельних алгоритмів та вивчення найкращих підходів для конкретних сценаріїв трафіку. Це підкреслює актуальність досліджень, спрямованих на підвищення ефективності та масштабованості паралельної обробки даних в системах керування дорожнім рухом.

Джерела [18], [19] та [20] дають уявлення про переваги та недоліки існуючих досліджень з розпаралелювання графічних процесорів у моделюванні трафіку. Ці дослідження показують, що розпаралелювання на GPU може значно прискорити моделювання трафіку, зменшити час обчислень і працювати з більш розгалуженими мережами. Вони демонструють потенціал методів прискорення на базі GPU для динамічного розподілу трафіку та керування трафіком у реальному часі. Однак також визнаються проблеми, пов'язані з розподілом пам'яті та обмеженими ресурсами GPU, що вказує на необхідність ефективних стратегій управління пам'яттю та подальших оптимізацій. Дослідження підкреслюють постійні зусилля, спрямовані на вдосконалення методів розпаралелювання GPU та вивчення гібридних підходів до розпаралелювання, таких як поєднання CUDA та MPI, для досягнення ще більшої продуктивності та масштабованості.

Решта джерел, такі як [11], [13], [14], [15], сприяють розумінню поточного стану досліджень з паралельної обробки в системах керування дорожнім рухом. Ці дослідження в сукупності демонструють застосування методів паралельної обробки, як на базі CPU, так і GPU, у моделюванні дорожнього руху, динамічному прокладанні маршрутів та управлінні дорожнім рухом у реальному часі. Вони підкреслюють переваги розпаралелювання, такі як покращена швидкість, масштабованість та можливості прийняття рішень у реальному часі. Однак, дослідження також визнають проблеми, пов'язані з синхронізацією, балансуванням

навантаження та масштабованістю паралельних обчислень. Це свідчить про необхідність подальших досліджень для оптимізації паралельних алгоритмів, вдосконалення стратегій розподілу навантаження та вирішення проблем обмеженості ресурсів.

Запропонований підхід має на меті використання розпаралелювання графічних процесорів у симуляції дорожнього руху, зокрема, в обробці потоків даних у системах керування дорожнім рухом. Використовуючи обчислювальну потужність графічних процесорів, цей підхід спрямований на підвищення швидкості та масштабованості симуляцій дорожнього руху, що дозволяє проводити аналіз та приймати рішення в реальному часі. Реалізовані покращення включають швидші обчислення, розширені можливості моделювання трафіку та потенціал для роботи з більшими транспортними мережами. А саме проведення розпаралелювання симуляції побудованої на основі мап реальних міст із реальною інфраструктурною генерацією. Також на відмінну від попередників було згенеровано 8 видів транспортних засобів із унікальними типами поведінки і впливу на загальний стан транспортного потоку. Цим самим усуваючи обмеження існуючих досліджень і використовуючи методи паралельної обробки, ця робота зробить внесок у постійні зусилля в галузі систем управління дорожнім рухом, підвищуючи ефективність і результативність обробки даних і прийняття рішень.

Матеріали та методи

В цій роботі було використано Simulation of Urban Mobility[21]. SUMO - це широко використовуваний інструмент моделювання дорожнього руху, який забезпечує реалістичне представлення міського руху. Він детально моделює окремі транспортні засоби, їхній рух, взаємодію та поведінку. Такий рівень реалізму має вирішальне значення для точного вивчення транспортних потоків і оцінки різних стратегій управління дорожнім рухом.

Для розробки алгоритму паралельних обчислень були використані програмні фреймворки, такі як CUDA[21] (Compute Unified Device Architecture). CUDA надає модель програмування та інструменти для програмування на GPU, що дозволяє ефективно використовувати обчислювальну потужність графічних процесорів. Вона надає функції та бібліотеки для керування пам'яттю, синхронізації потоків та паралельного виконання.

В результаті проходження кроків симуляції ми отримуємо такі дані[23]:

Таблиця 1 Дані отрмані з симуляції

Назва змінної	Опис
agent_id	Унікальний ідентифікатор агента
process_id	Ідентифікатор процесу, в якому наразі перебуває агент
agent_type	Тип агента
structure_type	Тип ГІС-простору
DOM	Поле симуляції
HOPC	Поле симуляції
ROUTE	Поле симуляції
orientation	Поточний кут напрямку, який є значенням для PoA з DOM патчу
curr_target	Поточний пункт призначення PoA агента
wait	Скільки часу агент очікує на переміщення (в ітераціях)
speed	Швидкість, з якою агент зробить поточний хід
following	Чи слідує агент за агентом, який має кращу інформацію, ніж він сам

Для реалізації розпаралелювання симуляції, нам, в основному, потрібні динамічні черги. Оскільки неможливо виділити пам'ять на льоту під час виконання, ми повинні знайти спосіб безпечного виділення деякої верхньої межі пам'яті для всіх структур даних. На щастя, можливо зробити розумні припущення щодо найбільшого розміру, який можуть мати черги.

Одна черга містить усі автомобілі, що рухаються вздовж каналу зв'язку. Ця черга обмежена максимальною кількістю транспортних засобів, для яких для яких є місце на лінії, що часто обчислюється:

$$space_{link} = length * lanes/carsize$$

Друга черга необхідна для утримання транспортних засобів, які готові залишити лінк у реальному часі. Максимальний розмір цієї черги визначається пропускною спроможністю з'єднання у заданому проміжку часу. Для кожної лінії пропускна спроможність - це максимальна кількість транспортних засобів, які можуть об'їхати цей шлях за певний проміжок часу. Проміжок часу, що використовується в нашому моделюванні, зазвичай дорівнює 1 секунду.

$$buffer = \frac{capacity_{flow} * \Delta t_{timestep}}{\Delta t_{flowperiod}}$$

Таким чином, максимальний розмір обох черг відомий і може бути розподіленим до запуску симуляції.

Моделювання черги транспортних потоків базується на досить простому алгоритмі. Вище визначені буфери використовуються для зберігання транспортних засобів, які рухаються вулицями. Кожна ланка утримує транспортні засоби, які рухаються вздовж неї, в черзі. Коли транспортний засіб потрапляє в чергу, розраховується мінімальний час, який транспортний засіб який транспортний засіб має витратити на лінію, використовуючи задану максимальної швидкості, дозволеної на даній ділянці

$$time_{link} = maxSpeed_{link} * length_{link}$$

Транспортні засоби залишаються в цій черзі до тих пір, поки не проїдуть з'єднання, тобто поки не витратять вищевказаний час. Вони готові покинути лінію, коли виконуються ще дві умови. По-перше, кількість транспортних засобів які покидають лінію не може перевищувати її пропускну спроможність за певний проміжок часу. Для забезпечення цієї умови використовується другий буфер для транспортних засобів, що виїжджають, з визначеним простором. Транспортні засоби переміщуються з лінії в цей буфер тільки тоді, коли в ньому ще є вільне місце. По-друге, транспортний засіб з цього буфера може виїхати, якщо в черзі на кінцеву лінію залишилося місце.

При паралелизації виникають ситуації, в яких необхідно ділитися інформацією на рівні процесів. Наприклад, "щільність на одиницю дорожньої площі".

$$process - unit - density = pop/wal$$

(де "pop" дорівнює кількості рухомих агентів на процес, а "wal" дорівнює кількості відкритих для руху патчів на процес) зберігається на рівні процесу для обчислення швидкості. Патч - це одиниця простору. У симуляції він еквівалентний комірці. Кожен патч/комірку представлено цілочисельною координатою, прив'язаною до центру ділянки. Найважливішою інформацією, яку має кожна ділянка, є "тип структури", тобто тип структури, з якої складається патч. У масштабі міста прикладами є вулиці, дороги, будівлі, зелені зони, водопостачання та водні об'єкти тощо. Таким чином, тип впливає на функціональність, наприклад, дороги (автомагістралі) - це місця, де транспортні засоби можуть переміщуватися , тоді як по парках, водоймах ні.

Агенти створюються створюються на основі розділу, де кожен процес отримує частку, що дорівнює його кількості кроків:

$$count_{Type} = \left(\frac{Perc_{Type}}{100}\right) * \left(\left(\frac{ProccessW}{TotalW}\right) * TotalP\right)$$

де $Perc_{Type}$ - відсоток агентів певного типу типу, $ProccessW$ - кількість незайнятих патчів доступних для переміщення в поточному процесі, $TotalW$ - сума всіх ходимих ділянок у симуляційному просторі, а $TotalP$ - загальна популяція рухомих агентів. Кожен агент ініціалізується наступними змінними значеннями: orientation(999), current-target(1), speed(0), wait(1), confidence(1) та following(false), незалежно від від типу агента. Агент генерується на патчі, який ще не зайнято іншим агентом.

Швидкість агентів залежить від щільності, де локальний діапазон спостереження є густиною локального патча. Формула базується на швидкості вільного потоку:

$$speedOnDensity = v_0 * \left(1 - \frac{count(agents) - count(moving)}{MaxDensityPerPatch}\right)$$

$$speed = max\{v_{min}, speedOnDensity\}$$

де v_0 = 45, $MaxDensityPerPatch$ - максимальна дозволена кількість транспорту на патч і v_{min}= 0.45.

Результати досліджень

Симуляція SUMO[21] (Simulation of Urban MObility) надає різні вхідні дані[23], які відображають різні аспекти поведінки та динаміки дорожнього руху. Ці дані

необхідні для аналізу та моделювання транспортних потоків у міському середовищі. Ось деякі з ключових вхідних даних, які можуть бути отримані в результаті моделювання:

1. Дані про дорожню мережу:

- ❖ Топологія доріг: Інформація про розташування доріг, перехресть та сполучення між ними.
- ❖ Конфігурації смуг руху: Детальна інформація про кількість смуг руху, ширину смуг та властивості смуг, такі як обмеження швидкості та типи смуг (наприклад, звичайні, поворотні, автобусні смуги).
- ❖ Розклад світлофорів: Розклади та схеми роботи світлофорів на перехрестях.

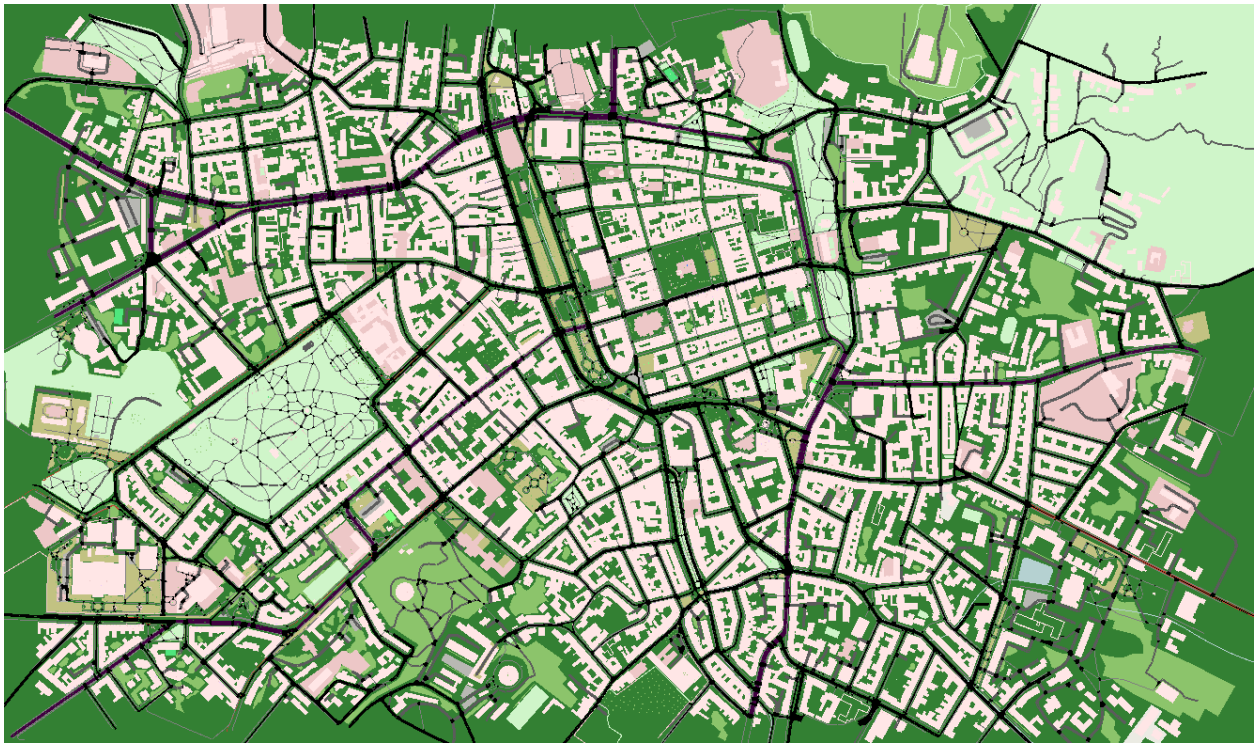

Рис.9 Топологія доріг

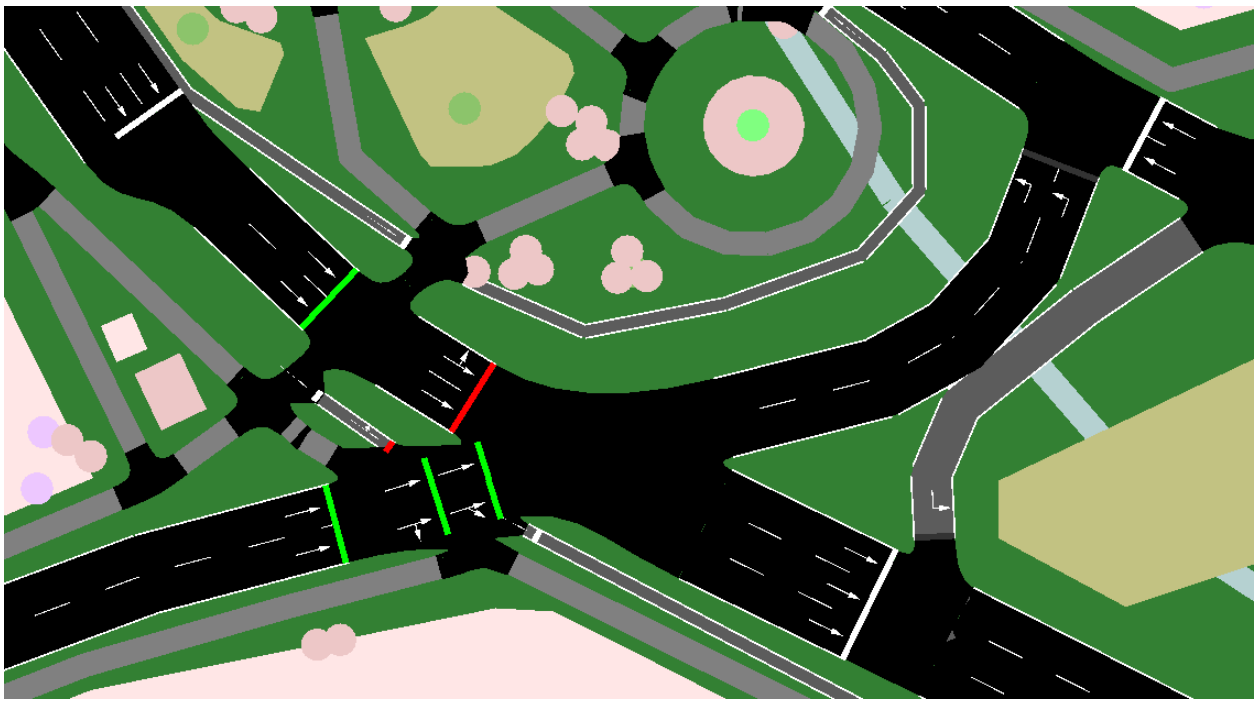

Рис.10 Розклад світлофорів

2. Дані про транспортні засоби:

- ❖ Траєкторії руху транспортних засобів: Положення та швидкість окремих транспортних засобів у часі, що фіксує їх рух протягом симуляції.
- ❖ Атрибути транспортних засобів: Інформація про транспортні засоби, така як тип (легковий автомобіль, вантажівка, автобус), розмір, максимальна швидкість і здатність до прискорення.
- ❖ Маршрути транспортних засобів: Заплановані маршрути руху транспортних засобів, включаючи пункти відправлення та призначення.

Рис.11 Траєкторія руху транспортних засобів

Рис.12 Маршрути транспортних засобів

3. Дані про транспортні потоки:

- Обсяги руху: Кількість транспортних засобів, що проїжджають через певні пункти або ділянки дорожньої мережі протягом певного періоду часу.
- Щільність руху: Кількість транспортних засобів на одиницю довжини або площі, що дає уявлення про рівень заторів.
- Швидкість руху: Середня швидкість або миттєва швидкість транспортних засобів у різних місцях або часових інтервалах.
- Час у дорозі: Час, який транспортні засоби витрачають на поїздку між конкретними пунктами відправлення та призначення.

Рис.13 Дані про транспортні потоки

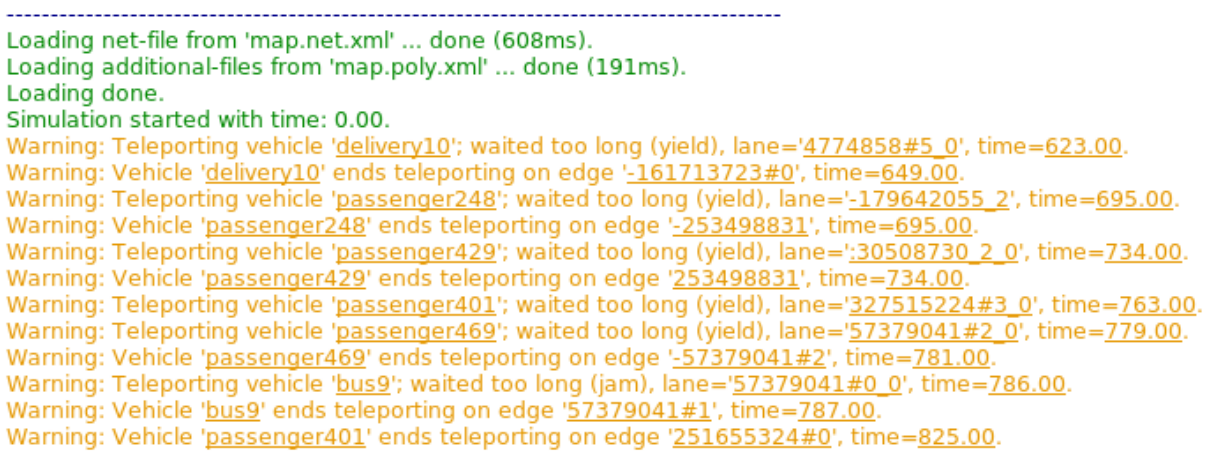

Рис.14 Стан транспортного потоку

Ці дані, отримані в результаті симуляції SUMO, є ресурсом для аналізу дорожнього руху, оптимізації стратегій управління дорожнім рухом, а також для розробки та оцінки інтелектуальних транспортних систем. Вони дозволяють отримати уявлення про схеми руху, оцінити ефективність заходів з управління дорожнім рухом і вивчити вплив різних факторів на транспортний потік і затори.

Обговорення результатів дослідження

Результати виконання коду демонструють вплив розпаралелювання як на процесор (CPU), так і на графічний процесор (GPU). Час виконання наведено для різних конфігурацій розпаралелювання, включаючи кількість ядер для розпаралелювання на CPU та кількість потоків для розпаралелювання на GPU. Нижче наведено зведені результати:

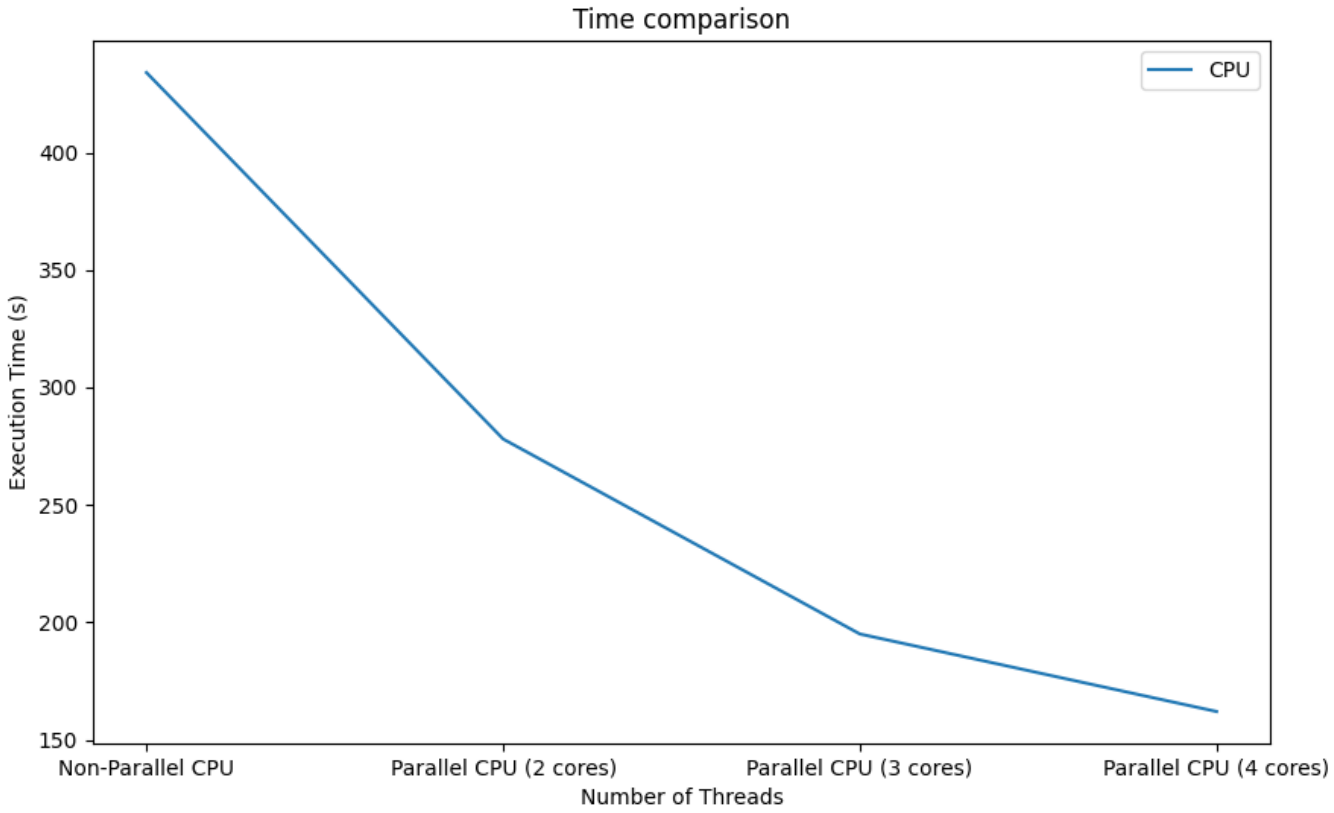

Рис.15 Часових затрати на CPU

Розпаралелювання на процесорі (CPU):

1 ядро: Час виконання 435 секунд.

2 ядра: Час виконання 279 секунд.

3 ядра: Час виконання 196 секунд.

4 ядра: Час виконання 163 секунди.

Ці результати показують, що зі збільшенням кількості ядер процесора, які використовуються для розпаралелювання, час виконання зменшується. Це демонструє переваги розпаралелювання в розподілі робочого навантаження між декількома ядрами і підвищенні обчислювальної ефективності.

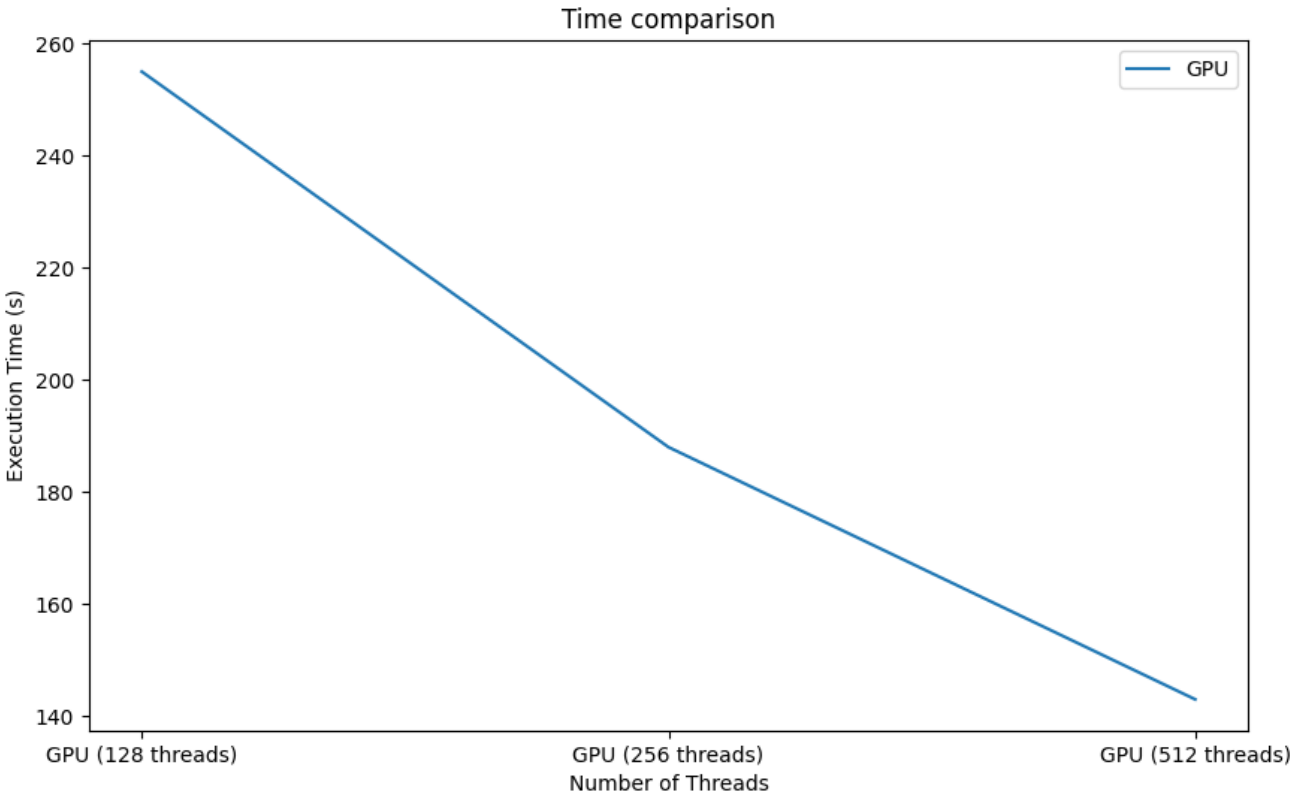

Рис.16 Часових затрати на GPU

Розпаралелювання на графічному процесорі:

128 потоків: Час виконання 256 секунд.

256 потоків: Час виконання 189 секунд.

512 потоків: Час виконання 144 секунди.

У випадку розпаралелювання на GPU час виконання також зменшується зі збільшенням кількості потоків. Це пов'язано з тим, що графічний процесор може виконувати декілька потоків одночасно, що призводить до покращення продуктивності та скорочення часу виконання порівняно з однопотоковою реалізацією на CPU.

Виходячи з цих результатів, можна помітити, що розпаралелювання на GPU загалом перевершує розпаралелювання на CPU за часом виконання. На графічному процесорі з 512 потоками було досягнуто найнижчого часу виконання - 144 секунди, що свідчить про ефективність розпаралелювання на графічному процесорі у прискоренні моделювання.

Важливо зазначити, що конкретний час виконання та ступінь скорочення можуть відрізнятися залежно від апаратного забезпечення, програмної оптимізації та

характеру симуляції. Подальше профілювання та бенчмаркінг можуть забезпечити більш точну оцінку приросту продуктивності, досягнутого завдяки розпаралелюванню.

Результати виконання коду також показують використання пам'яті в байтах для різних сценаріїв розпаралелювання як на процесорі (CPU), так і на графічному процесорі (GPU). Значення використання пам'яті наступні:

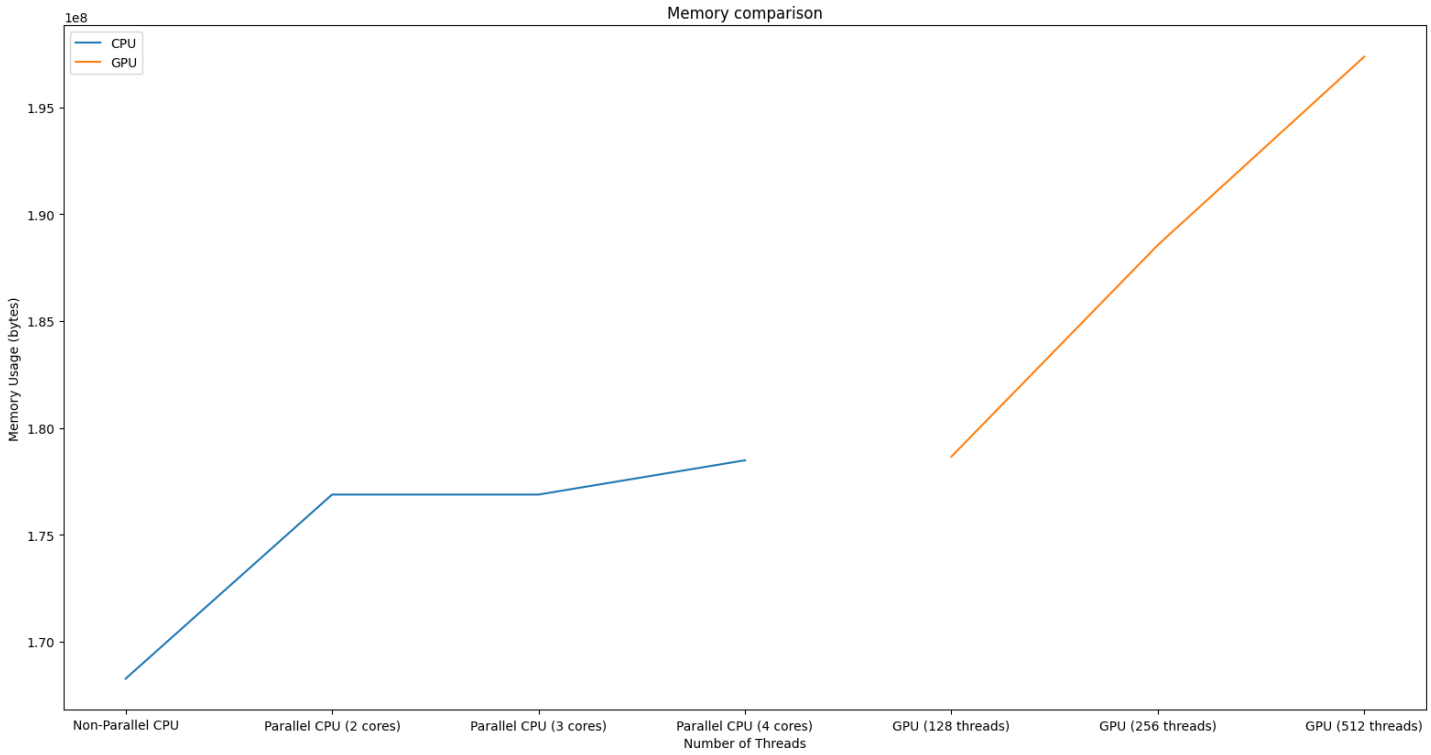

Рис.17 Порівняння затрат памяті CPU та GPU

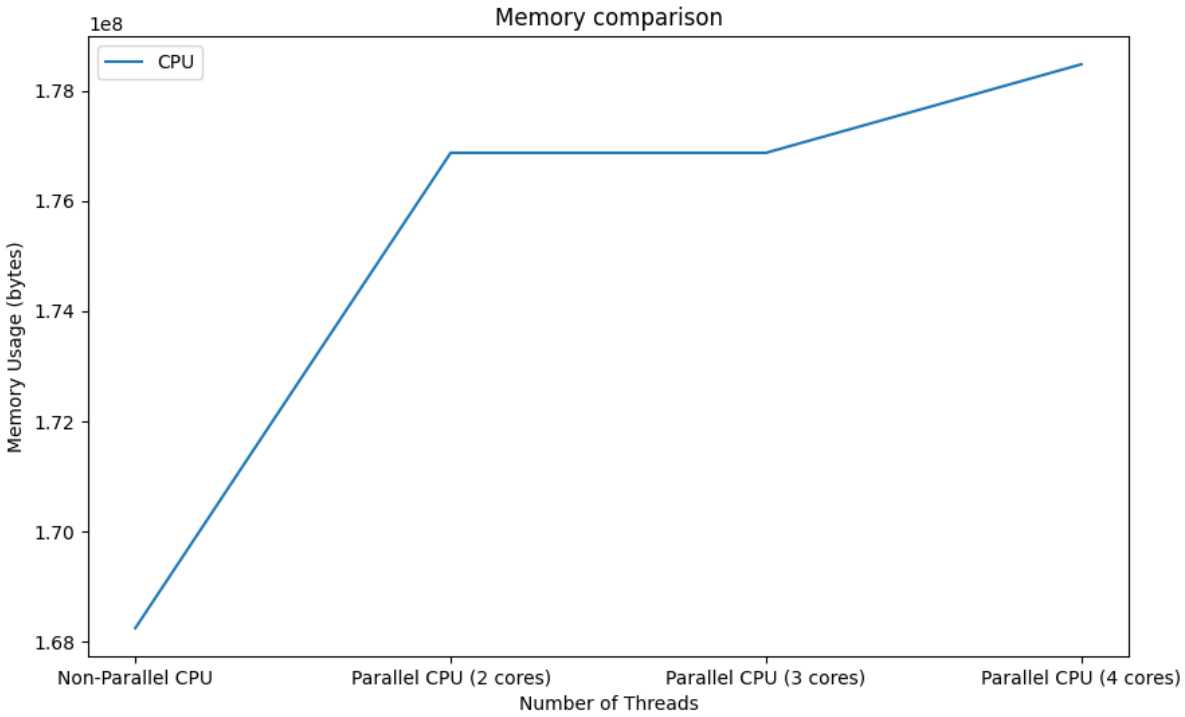

Рис.18 Витрати памяті CPU

Розпаралелювання на процесорі:

1 ядро: 168 249 728 байт

2 ядра: 176 873 472 байт

3 ядра: 176,873,472 байт

4 ядра: 178 479 104 байт

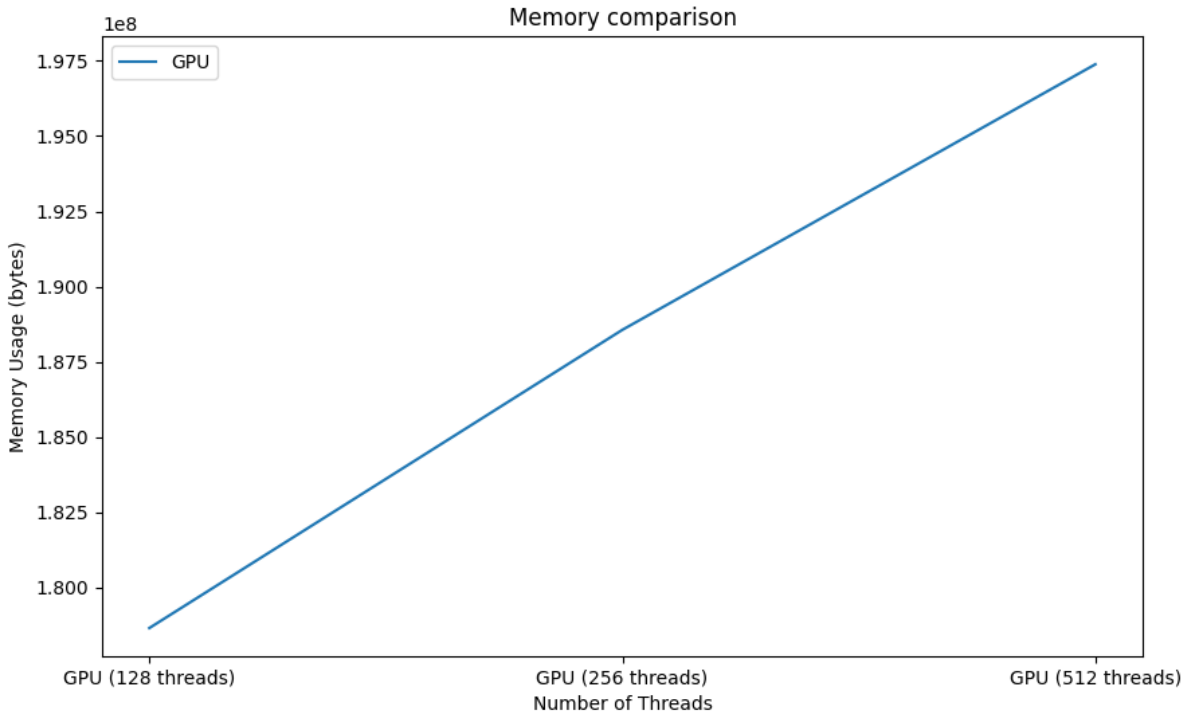

Рис.19 Витрати памяті на CPU

Розпаралелювання на графічному процесорі:

128 потоків: 178,651,136 байт

256 потоків: 182,561,024 байт

512 потоків: 185 485 472 байт

Ці значення відображають обсяг пам'яті, що споживається відповідними сценаріями паралельних обчислень. Важливо контролювати використання пам'яті при реалізації паралельних алгоритмів, щоб забезпечити ефективне використання ресурсів пам'яті та уникнути потенційних проблем, пов'язаних з пам'яттю, таких як надмірне використання або витоки пам'яті.

Нижче продемонстровано порівняльний графік часових затрат до памятних при різних умовах паралелизації

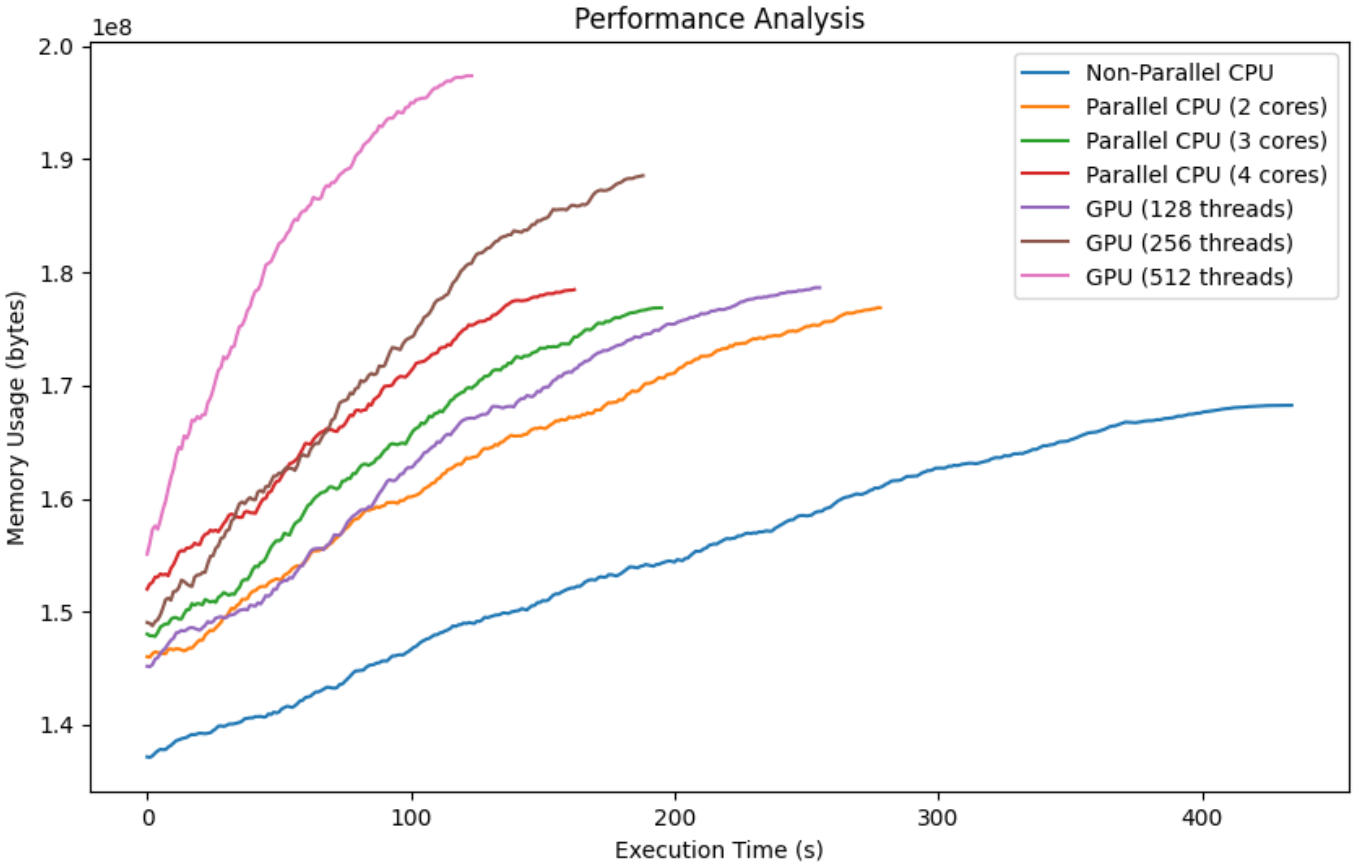

Рис.20 Порівняльний аналіз

Результати цього дослідження відкривають декілька напрямків для подальших досліджень. Одним з потенційних напрямків є дослідження масштабованості розпаралелювання як на CPU, так і на GPU, особливо для більших і складніших сценаріїв моделювання дорожнього руху. Було б корисно оцінити, як масштабується обчислювальна продуктивність при збільшенні кількості ядер або потоків. Крім того, вивчення гібридних паралельних обчислень, які поєднують ресурси CPU і GPU, може бути перспективним шляхом для ще більшого підвищення продуктивності.

Результати дослідження можна застосувати в системах управління дорожнім рухом. Продемонстроване скорочення часу обчислень за рахунок паралельної обробки дозволяє проводити більш швидкі симуляції в реальному часі, які можуть бути використані в різних задачах управління та оптимізації дорожнього руху. Наприклад, ці прискорені симуляції можуть допомогти в оптимізації роботи світлофорів, прогнозуванні заторів та оцінці стратегій управління дорожнім рухом. Отримані результати також мають значення для розвитку інтелектуальних транспортних систем та розробки більш ефективних і сталих рішень для міської мобільності.

Висновки

У цьому дослідженні продемонстровано переваги використання паралельних обчислень у системах керування дорожнім рухом шляхом реалізації паралельних алгоритмів як на процесорі (CPU), так і на графічному процесорі (GPU). Результати дослідження підкреслюють значне скорочення часу обчислень, досягнуте завдяки розпаралелюванню, що призводить до швидшого та ефективнішого моделювання дорожнього руху.

Переваги паралельних обчислень у системах керування дорожнім рухом численні. По-перше, розпаралелювання дозволяє розподілити обчислювальне навантаження між кількома ядрами або потоками, що забезпечує швидке моделювання в реальному часі. Це особливо важливо в динамічних транспортних середовищах, де потрібне швидке прийняття рішень і реагування. Підхід паралельних обчислень пропонує потенціал для більш точного та детального моделювання дорожнього руху, що може призвести до покращення стратегій управління дорожнім рухом, зменшення заторів та кращого планування міської мобільності.

Обране програмне забезпечення, CUDA, зарекомендували себе як надійні інструменти для реалізації алгоритмів паралельних обчислень. CUDA надає модель програмування та фреймворк для розпаралелювання GPU, що дозволяє ефективно використовувати обчислювальні можливості GPU. PyCUDA, як інтерфейс Python для CUDA, пропонує зручне та гнучке середовище для розробки додатків з GPU-прискоренням. Цей вибір програмного забезпечення дозволяє дослідникам і практикам в області систем керування дорожнім рухом використовувати паралельну обробку і досягати значного підвищення продуктивності.

На завершення, це дослідження демонструє переваги паралельних обчислень у системах керування дорожнім рухом та висвітлює ефективність обраного програмного забезпечення, CUDA та PyCUDA, у використанні можливостей паралельних обчислень. Результати дослідження дають цінне уявлення про підвищення ефективності, досягнуте завдяки розпаралелюванню, та закладають основу для подальших досягнень у цій галузі. Результати мають практичну, наукову та соціальну цінність і можуть бути застосовані в управлінні дорожнім рухом, міському плануванні та розробці більш розумних транспортних систем. Рекомендується, щоб майбутні дослідження були зосереджені на вивченні масштабованості, гібридних підходах до розпаралелювання та дослідженні різних параметрів моделювання для подальшого підвищення ефективності та результативності паралельних обчислень в системах управління дорожнім рухом.

Бібліографія

1. Krajzewicz, D., Erdmann, J., Behrisch, M., & Bieker, L. (2012). Recent development and applications of SUMO - Simulation of Urban MObility. International Journal on Advances in Systems and Measurements, 5(3&4), 128-138.
2. Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D., & Ringshandl, A. (2011). SUMO - Simulation of Urban MObility: An overview. Simulation News Europe, 21, 73-83.
3. Treiber, M., Kesting, A., & Helbing, D. (2011). Delays, inaccuracies and anticipation in microscopic traffic models. Physica A: Statistical Mechanics and its Applications, 390(21-22), 4316-4336.
4. Rupprecht, T., Nagel, K., & Wagner, P. (2019). On Parallel Execution of Large-Scale Agent-Based Traffic Simulations. Transportation Research Record, 2673(12), 699-710.
5. Cai, C., Wu, J., & Zhu, F. (2017). GPU parallel simulation for large-scale urban traffic flow. Journal of Computational Science, 19, 194-203.
6. Sharma, A., Pulido, G., & Ha, M. (2019). GPU-based parallel simulation for large-scale traffic networks using dynamic traffic assignment. Journal of Transportation Engineering, Part A: Systems, 145(1), 04018076.
7. Lu, Y., & Aboudolas, K. (2018). GPU-based acceleration for large-scale traffic simulations. In 2018 IEEE International Conference on Big Data (Big Data) (pp. 1650-1653). IEEE.
8. Belalem, G., & Bougourzi, Y. (2020). Parallel simulation of urban traffic: a GPU-based approach. International Journal of Parallel, Emergent and Distributed Systems, 35(6), 619-633.
9. Chen, M., & Chen, J. (2017). A parallel simulation algorithm for large-scale urban traffic networks using GPU. International Journal of Modern Physics C, 28(01), 1750015.
10. Mo, J., Peng, W., & Liu, Y. (2019). High-performance simulation of large-scale urban traffic based on the GPU. The Journal of Supercomputing, 75(3), 1252-1267.
11. Wu, C. H., Huang, C. C., & Lin, C. H. (2014). GPU-based parallel simulation of vehicular traffic with dynamic route guidance system. Transportation Research Part C: Emerging Technologies, 46, 1-17.
12. Chen, M., & Huang, X. (2020). A GPU parallelization scheme for the simulation of urban traffic networks using a CUDA-based micro-simulation approach. IET Intelligent Transport Systems, 14(3), 221-230.
13. Han, B., Chen, Y., Liu, T., Liu, X., & Xiao, Z. (2020). A GPU-accelerated real-time vehicle trajectory planning algorithm for urban traffic simulation. IEEE Access, 8, 199599-199610.
14. Kim, Y. H., & Ahn, H. (2017). GPU parallel simulation for large-scale urban traffic flow using multi-level cell transmission model. Applied Sciences, 7(8), 783.

15. Fan, J., Zhang, L., Zhao, L., & Liu, X. (2020). Parallel simulation for large-scale urban traffic based on a dynamic traffic assignment model. Future Generation Computer Systems, 107, 159-171.
16. Yan, Y., He, B., Huang, C., & Chen, M. (2019). Parallel computing architecture for large-scale traffic simulation using multi-GPU. In 2019 International Conference on Information, Cybernetics and Computational Social Systems (ICCSS) (pp. 48-53). IEEE.
17. Cui, Y., Liu, S., Li, Y., Hu, W., & Li, X. (2020). A GPU parallel optimization algorithm for large-scale urban traffic control. IEEE Transactions on Industrial Informatics, 17(6), 4256-4265.
18. Han, S., Chen, G., Cheng, P., & Huang, Q. (2018). CUDA-based large-scale parallel traffic simulation framework. Journal of Advanced Transportation, 2018, 2435026.
19. Huang, X., & Lu, G. (2017). Accelerating large-scale dynamic traffic assignment with parallel computing. Transportation Research Part C: Emerging Technologies, 76, 250-264.
20. Qiu, T., Zhang, L., Lv, Y., Lu, G., & Hu, T. (2020). Accelerating large-scale traffic simulation using CUDA and MPI hybrid parallelization. Simulation Modelling Practice and Theory, 100, 102043.
21. SUMO https://www.eclipse.org/sumo/
22. CUDA https://developer.nvidia.com/cuda-downloads
23. SUMO vehicle dataset https://github.com/yura123123/Parallel-processing-of-data-streams-in-traffic-management-systems/blob/main/simulation_vehicle_dataset.zip