ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: **«ДОСЛІДЖЕННЯ ШЛЯХІВ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ПРОЦЕСУ ЗБИРАННЯ ТА ОБРОБКИ ДАНИХ У СИСТЕМАХ УПРАВЛІННЯ ТРАНСПОРТНОЇ ІНФРАСТРУКТУРИ»**

на здобуття освітнього ступеня магістр
за спеціальності 123 Комп'ютерна інженерія
(код,найменування спеціальності)
освітньо-професійної програми Комп'ютерні системи та мережі
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Євгеній БОНДАРЕНКО
(підпис) *(ім'я, ПРІЗВИЩЕ здобувача)*

Виконав: здобувач вищої освіти гр.КСДМ-61
Євгеній БОНДАРЕНКО
(ім'я, ПРІЗВИЩЕ)

Керівник: Артем АНТОНЕНКО
к.н.т,доцент *(ім'я, ПРІЗВИЩЕ)*

Рецензент: _____
(ім'я, ПРІЗВИЩЕ)

Київ 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерної інженерії
Ступінь вищої освіти «Магістр»
Спеціальність 123 Комп'ютерна інженерія
Освітньо-професійна прорама Комп'ютерні системи та мережі

ЗАТВЕРДЖУЮ
Завідувач кафедрою Комп'ютерної інженерії
Наталія ЛАЩЕВСЬКА
(ім'я, ПРІЗВИЩЕ)
"____" _____2023 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Бондаренку Євгенію Сергійовичу
(прізвище, ім'я, по батькові здобувача)

1.Тема кваліфікаційної роботи: Дослідження шляхів підвищення ефективності процесу збирання та обробки даних у системах управління транспортної інфраструктури
керівник роботи Артем АНТОНЕНКО, к.т.н., доцент
(ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від "19" 10 2023 р. №145
2. Строк подання кваліфікаційної роботи 28.12.2023 р.
3. Вихідні дані кваліфікаційної роботи:
3.1. Різнорідні дані;
3.2. Машинне навчання;
3.3. Транспортна інфраструктура;
3.4. Науково-технічна література по темі магістрської роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
4.1. Аналіз процесів збору та злиття різнорідних даних у системах управління транспортною інфраструктурою.
4.2. Дослідження методів ефективного збору та обробки різнорідних даних у системах управління транспортної інфраструктурою.
4.3. Практична реалізаціїя архітектури системи збору та обробки даних, обгрунтування ефективності досліджуваних підходів.

5.Перелік ілюстраційного матеріалу: *презентація*

6. Дата видачі завдання "19" жовтня 2023р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Підбір технічної літератури	21.10.2023р. 28.10.2023р.	Виконано
2.	Аналіз особливостей збору даних з різнорідних джерел для управління транспортної інфраструктурою	28.10.2023р. 04.11.2023р.	Виконано
3.	Огляд методів ефективного збору та обробки різнорідних даних у системах управління транспортної інфраструктурою	04.11.2023р. 11.11.2023р.	Виконано
4.	Реалізація архітектури системи збору та обробки даних	11.11.2023р. 19.11.2023р.	Виконано
5.	Висновки по роботі, розробка демонстраційних матеріалів та доповіді.	19.11.2023р. 03.12.2023р.	Виконано
6.	Розробка демонстраційних матеріалів, доповідь.	03.12.2023р. 13.12.2023р.	Виконано
7.	Оформлення магістерської кваліфікаційної роботи	13.12.2023р. 20.12.2023р.	Виконано

Здобувач вищої освіти _____ Євгеній БОНДАРЕНКО
(підпис) *(ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Артем АНТОНЕНКО
(підпис) *(ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття ступня магістр: 71 стор., 43 рис., 7 табл., 23 джерел.

Мета роботи - підвищення швидкість доступу до даних у процесі прийняття рішень управління транспортної інфраструктури.

Об'єкт дослідження - процес збирання та обробки даних.

Предмет дослідження – методи збирання та обробки різнорідних даних у системах керування транспортною інфраструктурою.

Короткий зміст роботи: Проведено аналіз систем підтримки прийняття рішень управління транспортною інфраструктурою, їх архітектури та особливості. Одним з найважливіших завдань у системах є завдання збору та попередньої обробки даних для подальшого прийняття рішень. Представлено класифікацію даних за різними критеріями. Виконано огляд методів злиття різнорідних даних, що застосовуються в системах управління та підтримки прийняття рішень.

Досліджено фреймворк для генерації подій у реалізації систем, званій EVGEN, представлено архітектуру системи збору та попередньої обробки даних, проведено аналіз випробування та обґрунтовано ефективність досліджуваних моделей та методів.

КЛЮЧОВІ СЛОВА: РІЗНОРІДНІ ДАНІ, ІНФРАСТРУКТУРА, МАШИННЕ НАВЧАННЯ, АРХІТЕКТУРА, ПОТІК ДАНИХ, FRAMEWORK, МОДЕЛЬ, МЕТОД, КЛАСТЕР.

ABSTRACT

The text part of the qualification work for obtaining a master's degree: 71 pages, 43 figures, 7 tables, 23 sources.

The purpose of the work is to increase the speed of access to data in the decision-making process of transport infrastructure management.

The object of research is the process of data collection and processing.

The subject of research is methods of collecting and processing heterogeneous data in transport infrastructure management systems.

Summary of the work: An analysis of transport infrastructure management decision-making support systems, their architecture and features was carried out. A critical task in systems is the task of collecting and pre-processing data for further decision-making. Classification of data according to various criteria is presented. An overview of the methods of merging disparate data used in management and decision support systems was performed.

The framework for event generation in systems implementation, called EVGEN, was studied, the architecture of the data collection and preprocessing system was presented, the test analysis was performed, and the effectiveness of the studied models and methods was substantiated.

KEY WORDS: DIVERSE DATA, INFRASTRUCTURE, MACHINE LEARNING, ARCHITECTURE, DATA FLOW, FRAMEWORK, MODEL, METHOD, CLUSTER.

ЗМІСТ

ВСТУП……..10
РОЗДІЛ1 ОСОБЛИВОСТІ ЗБОРУ ДАНИХ З РІЗНОРІДНИХ ДЖЕРЕЛ ДЛЯ УПРАВЛІННЯ ТРАНСПОРТНОЇ ІНФРАСТРУКТУРОЮ……12
1.1 Системи підтримки прийняття рішень в управлінні транспортною інфраструктурою…….12
1.2 Аналіз класифікації даних……19
1.3 Проблеми збору та злиття різнорідних даних у системах підтримки прийняття управлінських рішень……...23
1.4 Дослідження моделей зберігання різнорідних джерел даних у системах з пакетною та потоковою обробкою даних…….....27
1.5 Огляд технологічних платформ збору, обробки та зберігання даних.....31
РОЗДІЛ 2 МЕТОДИ ЕФЕКТИВНОГО ЗБОРУ ТА ОБРОБКИ РІЗНОРІДНИХ ДАНИХ У СИСТЕМАХ УПРАВЛІННЯ ТРАНСПОРТН ІНФРАСТРУКТУРОЮ……...34
2.1 Огляд моделі розподіленого зберігання даних……...34
2.2 Аналіз методів збирання та попередньої обробки різнорідних даних для керування транспортною інфраструктурою…….....39
2.3 Метод реалізації запитів до різнорідних даних…….....45
2.4 Метод з елементами доповненої реальності……...51
РОЗДІЛ 3 РЕАЛІЗАЦІЯ АРХІТЕКТУРИ СИСТЕМИ ЗБОРУ ТА ОБРОБКИ ДАНИХ…….55
3.1 Реалізація фреймворку генерації подій транспортної інфраструктури для оцінки ефективності досліджуваних методів……55
3.2 Архітектура технологічного стека для реалізації досліджуваних методів…….....63
3.3 Обґрунтування ефективності досліджуваних методів у системі керування транспортною інфраструктурою……64
3.4 Обґрунтування ефективності методу у режимі реального часу із

застосуванням елементів доповненої реальності……76
ВИСНОВКИ…….80
ПЕРЕЛІК ПОСИЛАНЬ……...81
ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)……..84

ВСТУП

Актуальність теми. Системи підтримки прийняття рішень є розвитком комп'ютерних систем підтримки прийняття управлінських рішень (СППР) з допомогою застосування принципу проактивних обчислень, тобто перенесення дій людини на більш високий рівень управління та використання алгоритмів інтелектуальної обробки даних та машинного навчання.

При реалізації управління транспортною інфраструктурою критичним завданням є завдання ефективної обробки різнорідних даних, що отримуються з різних джерел. Якість та своєчасність даних впливає на оперативність та результативність прийняття рішень. Зростання обсягу даних і збільшення мережевої смуги пропускання, що надається для передачі даних, відкриває нові можливості управління транспортною інфраструктурою, але при цьому виникають проблеми ефективної обробки даних.

Тому для вирішення завдань управління транспортною інфраструктурою необхідно вирішити питання збору, зберігання та забезпечення ефективного доступу до різнорідних даних великого обсягу. Найважливішим фактором тут є час доступу до різноманітних даних у процедурах прийняття рішень.

Мета роботи - підвищення швидкість доступу до даних у процесі прийняття рішень управління транспортної інфраструктури.

Для виконання поставленої мети, у магістерській роботі розроблено та виконано наступні завдання:

- аналіз процесів збору та злиття різнорідних даних у системах управління транспортною інфраструктурою.

- дослідження методів ефективного збору та обробки різнорідних даних у системах управління транспортною інфраструктурою;

- практична реалізаціїя архітектури системи збору та обробки даних, обгрунтування ефективності досліджуваних підходів.

Об'єкт дослідження – процес збирання та обробки даних.

Предмет дослідження – методи збирання та обробки різнорідних даних у системах керування транспортною інфраструктурою.

Методи дослідження. У процесі виконання поставлених завдань використовувалися методи системного аналізу, теорії прийняття рішень, методи розподілених та паралельних обчислень, методи машинного навчання та інтелектуальної обробки даних.

Джерела дослідження:

- https://www.it.ua/knowledge-base/technology-innovation/big-data;
- https://www.questionpro.com/blog/data-collection-methods;
- https://www.vedantu.com/commerce/sources-of-data;
- https://hal.science/hal-03741847/document;
- https://www.scribbr.com/methodology/data-collection.

Наукова новизна одержаних результатів. Новизна полягає в сукупності моделі та методів збору та попередньої обробки різнорідних даних у системах управління транспортною інфраструктурою, що включає: модель зберігання різнорідних даних відповідно до концепції «озеро даних»; методу збору та попередньої обробки даних; метод аналізу різноманітних даних у режимі реального часу в системах управління транспортною інфраструктурою з елементами доповненої реальності.

Практична значущість одержаних результатів. Практична значущість одержаних результатів полягає у дослідженому програмному забезпеченні, що реалізує розглянуті методи.

Апробація результатів магістерської роботи. Основні положення і результати магістерської роботи доповідались і обговорювались на двох науково-практичних конференціях.

Публікації. За матеріалами роботи опубліковано одну статтю у науковому журналі.

РОЗДІЛ 1 ОСОБЛИВОСТІ ЗБОРУ ДАНИХ З РІЗНОРОДНИХ ДЖЕРЕЛ ДЛЯ УПРАВЛІННЯ ТРАНСПОРТНОЮ ІНФРАСТРУКТУРОЮ

1.1 Системи підтримки прийняття рішень в управлінні транспортною інфраструктурою

На сьогоднішній день для вирішення проблем, що обмежують розвиток сучасних інформаційних технологій, використовуються адаптивні та проактивні системи [1,2]. Аналіз показав, що в останні два десятиліття ємність дисків щорічно збільшувалася приблизно вдвічі, обчислювальна продуктивність – у 1,6 раза (за законом Мура), а ефективність персональних мережевих можливостей – у 1,3 раза. Це призводить до зростання складності обробки великих даних.

Проактивні системи розширюють уявлення про застосування комп'ютерів, підтверджуючи необхідність моніторингу фізичного світу та впливу на нього, орієнтації на процеси, що передбачають складні взаємодії з реальним світом, але зараз обмежені нині ступенем необхідного втручання з боку людини. Деякі дослідження виходять за рамки вже створених систем повсюдних обчислень і розраховані на майбутні конфігурації, що передбачають, що людина працює з тисячами об'єднаних у мережу комп'ютерів.

Зазначимо, що загальною метою проактивних комп'ютерних систем є орієнтація на системи, у яких людина виконує спостережну (супервізорну) функцію, яка частково чи повністю реалізується автоматичними системами. Іншими словами, проактивні комп'ютерні системи зосереджуються на забезпеченні спостережної діяльності людини, коли користувач не втручається в управління системою - доти, доки не буде потрібна його участь у прийнятті критично важливих рішень. Зміна відносин між користувачем та комп'ютером за часом призводить до необхідності розробки та вдосконалення проактивних систем ефективної підтримки прийняття рішень. У цих випадках користувач грає роль спостерігача та приймає важливі рішення у системах. Проте, дослідження в галузі адаптивних та

проактивних систем багато в чому взаємопов'язані. На рис. 1.1 подано відношення між парадигмами обчислень.

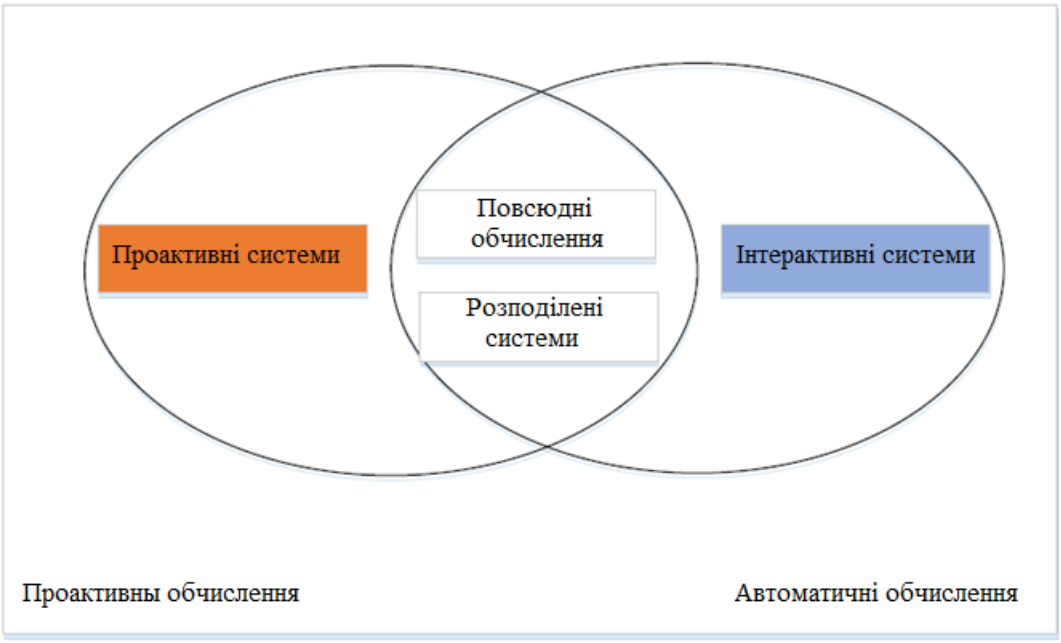


Рисунок 1.1 – Відношення між парадигмами обчислень

Системи підтримки прийняття управлінських рішень, що використовуються в технічних, економічних, соціальних системах, дозволяють покращити якість управління процесами прийняття рішень. Особливо у випадках ситуації, що постійно змінюється, і у відсутності у користувача можливості досить інтенсивно взаємодіяти з пристроєм, ефективна підтримка прийняття рішень не забезпечується традиційними підходами, такими, як адаптивними, реактивними.

Відповідно до роботи [2], особі, яка приймає рішення (ЛПР), пред'являються для вибору варіанти, представлені своїми векторними оцінками, окремі координати яких повинні відповідати його понятійним уявленням про предметну область. Тільки в цьому випадку ЛПР зможе сформулювати свої переваги за кожним показником, що дозволяє застосовувати вже певні формалізовані процедури порівняння варіантів для вибору єдиного варіанту або принаймні звуження безлічі альтернативних. Таким чином, можна визначити, що система підтримки прийняття рішення (СППР) повинна включати два основних модулі: прогнозування оцінок

показників аналізованих варіантів; формалізації переваг ЛПР та впорядкування аналізованих варіантів у порядку переваги.

Нині проактивні СППР застосовуються у технічних, економічних, соціальних системах, а й під час управління транспортною інфраструктурою міста.

Розробка проактивних систем спирається на сім принципів, розглянемо три основні принципи проактивних систем підтримки прийняття рішень.

Зв'язок із фізичним світом. Значна частина існуючої нині обчислювальної інфраструктури пов'язує персональні комп'ютери з масивом серверів. Створене віртуальне середовище дозволяє породжувати, обробляти та зберігати інформацію, яка, через людей, може опосередковано впливати на фізичний світ. Щоб сформувати оточення, в якому комп'ютерні системи допомагають у вирішенні повсякденних завдань, фізичний світ має бути оснащений такими комп'ютерними системами, які здатні детально вивчати дійсність, використовуючи зібрану інформацію для ефективних впливів на неї. Прикладом цього можуть бути мікрокліматичні прогнози погоди, моніторинг дорожнього трафіку та визначення можливого місцезнаходження людей у будівлі, що постраждала від землетрусу.

Немає необхідності говорити, що створення подібних систем пов'язані з певними проблемами. По-перше, постають цілком прагматичні питання, такі як техпідтримка, комунікації та наявність відповідних джерел електроживлення. По-друге, питання координації та управління піднімаються на новий рівень складності, необхідно створювати нові протоколи, що дозволяють підтримувати відповідні потоки даних, а управління енергоспоживанням перетворюється на критично важливий параметр для датчиків, які мають працювати від незалежних джерел енергії. Застосування фізичних датчиків у глобальному масштабі – завдання справді грандіозна, але суспільство стає дедалі складнішим, тому результат того вартий.

При масштабуванні систем настільки, щоб вони були здатні вести моніторинг фізичного світу, відразу ж виникають проблеми, які стосуються адміністрування та використання, тобто саме ті проблеми, на вирішення яких націлена концепція адаптивних комп'ютерних систем. Однак за рахунок використання простих вузлів,

які можна описати окремо, ми можемо більше дізнатися про методики, потрібні для підтримки більших мереж, зібраних з традиційних комп'ютерів. Складні бездротові мережі датчиків, подібні до тих, які розроблені вченими з Каліфорнійського університету в Берклі [4,5], мають саме такі характеристики.

Функціонування в масштабі реального часу та у замкнутому циклі. Якщо припустити, що комп'ютери будуть більш тісно інтегровані з фізичним світом, зворотний зв'язок реального часу стане критично важливим чинником. У 60-ті роки комп'ютерні системи були або повністю інтерактивними, що включали людину в цикл управління, або абсолютно негнучкими, створеними на основі спеціалізованої системи управління. Щоб повністю інтегруватися у завдання реального світу, необхідно реагувати швидше, ніж це можливо у разі участі людини у циклі управління: комп'ютерні системи мають «відгукуватися» на події фізичного світу у реальному часі.

Якби обчислювальні системи загального призначення були перепроектовані таким чином, щоб могли гарантувати зворотний зв'язок у реальному часі, з'явилося б безліч нових проактивних додатків. Однак більшість програмних систем не гарантують зворотний зв'язок у реальному часі, приховуючи складність за рівнями абстракції та не враховуючи час відповіді, що визначається різноманітними факторами. У світі вбудованих систем, як правило, вдаються до допомоги спеціалізованого програмного забезпечення інструментарію, що використовує для критично важливих керуючих впливів можливості операційної системи реального часу, які не підтримуються більшістю звичайних платформ.

Прогнозування. Прогнозування – основа проактивних комп'ютерних систем. Щоб системи були справді проактивні, вони, у певному сенсі, мають передбачати майбутнє. Застосування низки перспективних методик дозволить системам швидко опрацьовувати ситуації реального світу, надаючи необхідний рівень взаємодії. У тому числі існують операції, що враховують контекст та статистичні міркування. Далі розглянемо конкретно перелічені методики:

1.Операції, які враховують контекст. Системи, що підтримують мобільні та бездротові комунікації, дають можливість використовувати контекстну

інформацію, наприклад, про фізичне місцезнаходження та про готовність навколишньої інфраструктури, щоб змінити поведінку додатків. Адаптивні та проактивні системи для прийняття найважливіших рішень можуть використовувати контекст, враховуючи стан середовища, в якому вони працюють. Адаптивні комп'ютерні системи можуть бути корисні безпосередньо для підтримки нових конфігурацій, наприклад, дозволяючи визначати локальні ресурси та налаштування операцій. Проактивні системи, що діють на вищому рівні, можуть фільтрувати інформацію для відображення та настроювати результати виконання команд. Місцезнаходження – один з найбільш корисних параметрів для визначення контексту, і надання мобільним пристроям і системам високоточної інформації про місцезнаходження, що підтримують їх, – одна з першочергових цілей дослідників.

2. Статистичні міркування. За останнє десятиліття значно удосконалено підходи, у яких для вирішення важливих проблем використовуються статистичні методи. Вони перевершили і навіть замінили собою деякі з традиційніших підходів, що використовують детерміновані методи. Застосування подібних методик до управління та аналізу складних систем дає значні переваги як у IT, так і для управління процесами довільної природи.

У деяких веб-додатках (наприклад, у машині пошуку Google, Yandex) такі методики вже застосовуються для видобутку даних. Крім того, статистичні методи з успіхом використовуються і в інших галузях інформатики, таких як розпізнавання мови, візуальна обробка і навіть алгоритми маршрутизації, реалізовані в деяких засобах автоматизованого проектування. У перспективі планують використати статистичні методи для обробки подій фізичного світу в режимі реального часу.

При розробці та управлінні інфраструктурою великих міст доцільно реалізовувати проактивні системи з метою підтримки та управління ризиками безпеки через специфіку транспортної інфраструктури: розкиданого характеру виконуваних робіт, технічних труднощів та численних учасників [6]. У роботі [6] показано, що традиційна діагностика безпеки на місці не може повністю вирішити всі проблеми, особливо ті, які пов'язані з ризиками, що виникають.

Проактивні системи можуть бути реалізовані для управління міською транспортною системою: моніторинг трафіку та його прогнозування. Під час розробки інтелектуальної системи управління транспортним рухом великих міст виникає завдання проактивного управління транспортними потоками. Деякі великі міста впровадили синхронізовану систему світлофорів на основі історичних даних про трафік з метою збільшення транспортних потоків на основних перехрестях, що призвело до скорочення часу в дорозі. Проактивне керування також застосовується у транспортних системах для вирішення задачі дорожніх пробок. В роботі [7] запропоновані проактивні алгоритми для об'єднання різних транспортних потоків з метою покращення пропускної спроможності трафіку, зменшуючи або усуваючи вузькі місця на автомагістралях, зокрема, для ситуацій злиття, таких як перехрестя, де з'їзд веде на шосе. Проактивні підходи можна застосовувати в управлінні автострадами. Дослідження у галузі управління автострадами були зосереджені на автоматичному виявленні інцидентів. Ідея виявлення інцидентів включає аналіз закономірностей в даних спостереження за дорожнім рухом, що спостерігаються відразу після інцидентів. Оскільки дані про трафік для автострад збираються безперервно, можна розробляти моделі, використовуючи історичні дані про події, та застосовувати їх у режимі реального часу для вивчення даних про трафік та виявлення будь-яких подій, які могли статися на автостраді.

Проактивні системи, які виконують проактивне управління, засноване на короткостроковому прогнозуванні стану руху, мають великі перспективи у боротьбі з пробками транспортних мереж у великих містах.

На сьогодні в деяких містах впроваджено проактивні системи управління рухом - бездротові мережі, камери спостереження та підключені вуличні світлофори. Завдання управління інфраструктурою набагато важче, ніж окрема технологія, оскільки потрібно об'єднувати технології підключення, апаратного та програмного забезпечення для спільної роботи в одній системі.

Ухвалення попереджувальних рішень може призвести до колосальної економії коштів під час управління міськими процесами. У концепції проактивних систем лежить схема: виявити – спрогнозувати – вирішити – діяти та принципи

побудови СППР на основі обробки подій. У нових СППР інтелектуальна обробка подій дозволяє передбачати потенційні проблеми під час виконання процесу та, таким чином, забезпечує проактивне управління процесом. Однією з галузей промисловості, яка може очікувати на відповідні переваги від застосування проактивної обробки подій, є транспортування вантажів. На сьогоднішній день транспортні компанії стикаються з численними стохастичними проблемами при управлінні відвантаженням товарів [8]. Однією з таких проблем є точні обсяг, вага і кількість одиниць, які відправник вантажу хоче відправити. Одним із можливих підходів до вирішення цієї проблеми є використання моніторингу в режимі реального часу та проактивне попередження, щоб допомогти транспортним компаніям прогнозувати фактичну вагу, обсяги та кількість відправлених вантажів.

Проактивні системи також застосовні для вирішення завдань оптимізації витрат ресурсів мешканців міста за рахунок застосування методів прогнозування транспортного стану з використанням історичних даних, зібраних з транспортних об'єктів. Прикладом цього завдання є пошук оптимального шляху залежно від стану транспорту та історичних даних за інтервалами часу.

Для реалізації проактивної підтримки прийняття рішень критичною задачею є завдання ефективної обробки різнорідних даних, одержуваних з різних джерел. Якість та своєчасність даних впливає на прийняття рішень. У сучасній науковій дискусії є питання про розробку підходів ефективного збору та обробки даних, пошуку нових способів зберігання та попереднього аналізу. Експонентне зростання щільності зберігання даних та збільшення мережної смуги пропускання, що надається для передачі даних, дозволяє проактивним обчислювальним системам швидко надавати дані користувачам без їхньої безпосередньої участі. Проактивні системи можуть використовувати мобільну пам'ять високої щільності, завдяки чому системи здатні заздалегідь завантажувати дані, які можуть бути потрібними користувачам у майбутньому, не обтяжуючи їх громіздкими мобільними пристроями. Так само мережі з широкою смугою пропускання можуть за короткий період передавати великі обсяги даних на сервер, фізично розташований неподалік користувача. Однак адаптивні методики повинні довести, що користувачі можуть

довіряти подібним системам, гарантуючи, що вони працюватимуть у різних умовах.

Локальне кешування даних та переміщення даних можуть відігравати важливу роль у підтримці мобільності користувачів. З іншого боку, локальне кешування даних може бути корисно користувачеві, якщо інформація для кешування обрана розумним чином, хоча дані не завжди можуть виявитися актуальними. За рахунок використання обох цих методик проактивні комп'ютерні системи покликані надавати дані комп'ютерам, що переміщаються у фізичному світі в реальному часі, тим самим формуючи єдине уявлення.

1.2 Аналіз класифікації даних

За ISO/IEC/IEEE 24765-2010 дані – зареєстрована інформація, подання фактів, понять чи інструкцій у формі, прийнятній для спілкування, інтерпретації чи обробки людиною або за допомогою автоматичних засобів. В інформатиці та інформаційних технологіях дані визначаються як:

- подання багаторазової інтерпретації подання інформації у формалізованому вигляді, придатному для передачі, зв'язку або обробки;

- форми подання інформації, з якими мають справу інформаційні системи та їх користувачі.

За визначенням оксфордського словника, дані – величини, знаки чи символи, якими оперує комп'ютер і які можуть зберігатися і передаватися у вигляді електричних сигналів, записуватися на магнітні, оптичні чи механічні носії.

У комп'ютерних системах будь-яка інформація, подана у допустимій для комп'ютера формі – тексти, малюнки, музика та ін – вважаються даними.

Дані можуть бути різнорідними, які зібрані з різних джерел, таких як веб-додатки, датчики, сенсори, банківські транзакції і т.д.

Отже, у системах підтримки прийняття рішення завдання ускладнює різнорідність даних. На основі аналізу літератури та реальних бізнес-процесів виділено наступну класифікацію типів даних, що використовуються в обробці за

проактивної підтримки прийняття рішень. Машинні дані сенсорного типу (sensors data), як правило, кількісні дані, одержувані внаслідок вимірювання характеристик цільової системи. Подієві дані (log data), дані, що фіксують дію, подію або стан об'єкта в певний момент часу. В окремому випадку подієві дані є інтерпретацією людиною інших типів даних та результатом формування запису в журнал. Візуальні дані - зображення або відео, на яких фіксується поведінка цільової системи або систем в операційному оточенні, що прямо або непрямо характеризують стан системи. Текстові дані (textual data) – неструктуровані дані, які є повідомлення природною мовою і є, як правило, інтерпретацією проблемних ситуацій чи переваг користувачів. Дані соціальних мереж (social data) – неструктуровані дані, що являють собою сукупність текстових даних, виразів («лайки», «репости»), що характеризують паттерни поведінки особи, яка приймає рішення. Георозподілені дані (geospatial data) – дані про події, які як основні атрибути та атрибути тимчасової мітки мають дві географічні координати (широту і довготу). У роботі [9] було представлено георозподілені дані, що обробляються у створенні системи управління містами. На рис.1.2 наведено приклад візуалізації георозподілених даних.

За способом розташування джерел даних слід говорити про зосереджені дані (одне централізоване джерело), територіально-розподілені джерела. Напрямок досліджень, що займається збором та обробкою даних, отримав назву злиття даних (data fusion).

В результаті аналізу робіт [9,10] було побудовано класифікацію даних за різними критеріями, що використовуються в обробці при проактивній підтримка прийняття рішень. Розроблена класифікація представлена на рис. 1.3.

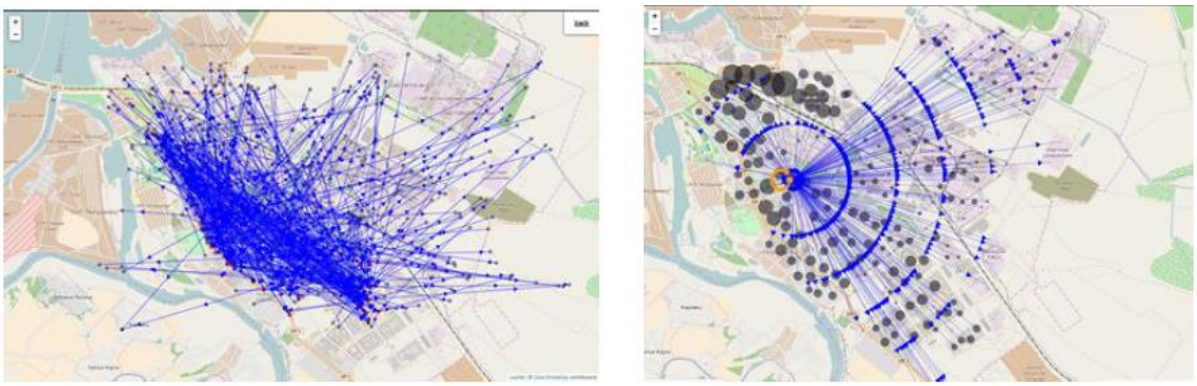


Рисунок 1.2 – Приклад візуалізації георозподілених даних. Вузли точки відправлення – призначення всередині міста, стрілки – напрямок необхідного руху

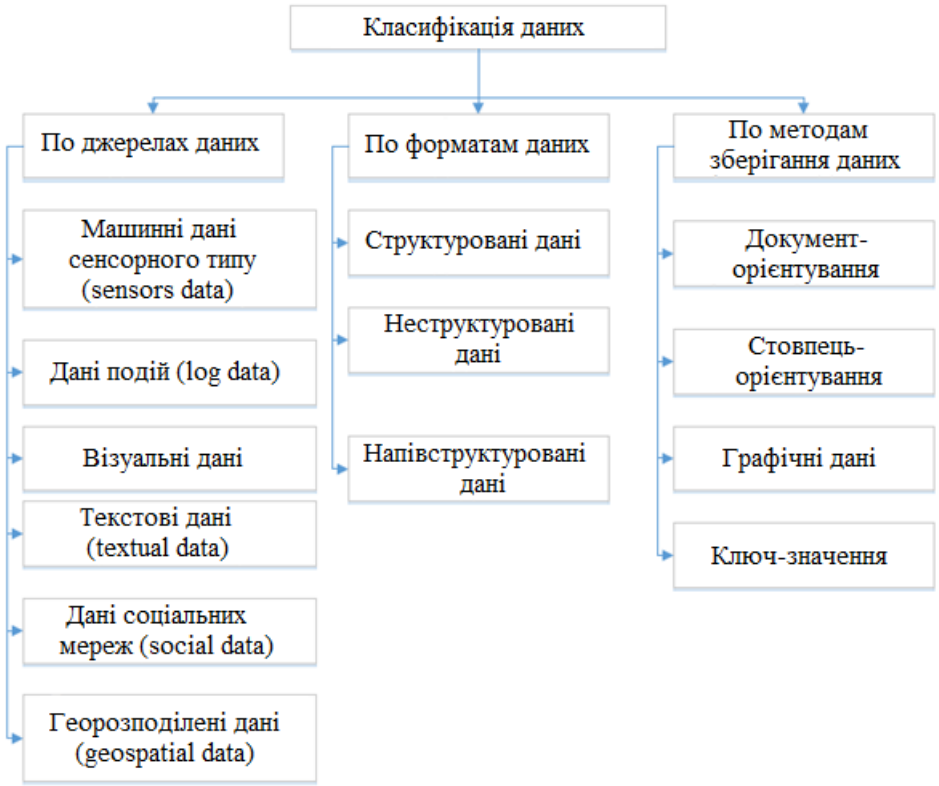


Рисунок 1.3 – Класифікація даних, що використовуються в обробці за проактивної підтримки прийняття рішень

Залежно від структури даних, дані можуть бути поділені на 3 основні класи: структуровані, неструктуровані та напівструктуровані. Структуровані дані часто зберігаються у базах даних, у яких вони вже доступи та оброблені у формі з фіксованим форматом. Цей формат зазвичай відомий заздалегідь. В іншому

випадку, неструктуровані дані є даними невідомої структури. Ця форма даних характеризується рядом складнощів для обробки та вилучення корисної інформації. Типові приклади неструктурованих даних - гетерогенне джерело, що містить комбінацію простих текстових файлів, картинок, або відео, отримані з камер на дорогах для відстеження транспортних засобів. Напівструктуровані дані містять описані властивості структурованих та неструктурованих. Ці дані не визначаються у базах даних, а у файлах формату XML, JSON.

Як правило, збільшення кількості джерел даних призводить до збільшення обсягу даних. У зв'язку з цим виникло поняття «великі дані». Відповідно до Gartner, великі дані характеризуються обсягом, швидкість надходження, різноманітності та мінливості.

1. Об `єм. Розмір даних — найважливіший показник щодо можливої видобутої цінності. Щодня 6 мільйонів людей використовують цифрові медіа, що, за попередніми оцінками, генерує 2.5 квінтильйона байтів даних. Тому обсяг – перша до розгляду характеристика.

2. Різноманітність. Цей аспект великих даних пов'язаний із гетерогенними джерелами та природою даних, які можуть бути як структурованими, так і неструктурованими. Дані можуть мати різний формат: електронні листи, фото, відео, файли PDF, текстові файли, аудіофайли і т.д. Така різноманітність неструктурованих даних призводить до проблем у зберіганні, обробці та аналізі: за статистикою, 27% підприємств не впевнені, що працює з відповідними даними.

3. Швидкість створення. Те, наскільки швидко дані накопичуються та обробляються задоволення вимог, визначає потенціал рішення, працюючого з даними з різних джерел: дані бізнес-процесів, лог-файлів, дані соціальних мереж і медіа, сенсорів та інших.

4. Мінливість описує (структурну та семантичну) мінливість даних у часі, що ускладнює обробку та управління. Так, наприклад, більша частина даних неструктурована за своєю природою.

Крім цього, у роботі [11] авторами було виділено ще кілька характеристик великих даних:

1. Коректність. Під коректністю розуміється міра, що характеризує наскільки дані можна інтерпретувати і загалом довіряти даним.

2. Волатильність у великих даних може ставитись до того, наскільки дані актуальні і як довго вони повинні зберігатися в системах. Організації повинні планувати та визначати, у який момент дані більше не можуть бути використані для аналізу.

3. Цінність відноситься до знань, отриманих у результаті аналізу даних [12]. Цінність зазвичай розглядається в контексті неодноразового використання даних, знання також можна використовувати повторно для будь-якого майбутнього аналізу мети шляхом об'єднання з іншими наборами даних.

1.3 Проблеми збору та злиття різнорідних даних у системах підтримки прийняття управлінських рішень

Актуальність проблем збору та злиття різнорідних даних обумовлена обов'язковими вимогами, що висуваються під час реалізації будь-якої системи прогнозної аналітики або проактивної системи підтримки прийняття рішень. Підходи, що вирішують подібні завдання, виділені в область досліджень під назвою «змішування та інтеграція даних» (неусталений термін від англійської Information Fusion), в рамках якої пропонуються методи консолідації даних для подальшого зберігання та обробки.

Особливості та проблеми збору даних з різнорідних джерел. При реалізації систем прогнозної аналітики чи проактивної системи ППР має бути розроблена система збирання та консолідації даних. У системах ППР дані можуть бути різнорідними, такими як текстові, лог-файли, медіа-дані і т.д. Ці дані можуть бути зібрані з різних джерел: зі зв'язаних баз даних і джерел для досягнення вищої точності і конкретніших висновків, ніж це може бути досягнуто за рахунок використання тільки однієї системи збору даних [1,2]. На сьогоднішній день концепція інтеграції різнорідних даних є досить новою, тому існує низка проблем, вирішення яких необхідне. Завдання збору та консолідації різнорідних даних є

актуальним при реалізації системи проактивної підтримки прийняття рішень. Для вирішення завдання збору та консолідації різнорідних даних у проактивних системах мають бути реалізовані адаптери, які одержують дані з різних джерел. Адаптери працюють як системи фільтрації важливих даних для подальшої обробки. Сформулюємо такі вимоги розробки адаптерів збору даних у проактивних системах ППР:

1. Необхідно визначити структури форматів даних для перетворення з вихідних даних;

2. Необхідно забезпечувати якість отриманих даних у режимі реального часу, використовуючи методи оцінки та забезпечення якості даних, також як методи обробки викидів даних;

3. Слід мінімізувати участь людини у процесі збирання різнорідних даних;

4. Необхідно забезпечувати збирання даних у реальному часі з різним ступенем інтенсивності надходження різнорідних даних.

В результаті аналізу процесу збирання різнорідних даних відзначено низку проблем. Дані, одержані адаптерами, можуть бути неструктурованими. Тому для отримання важливих даних з отриманих даних необхідно реалізувати механізми аналізу даних (парсери). Це значно ускладнює реалізацію систем, що забезпечують збирання даних з безлічі різнорідних джерел. Ситуація ускладнюється, коли структура вихідних даних змінюється у часі. Це призводить до постійного перегляду реалізації парсера даних. Це безпосередньо впливає якість збору даних зокрема, і якість прийняття управлінські рішення загалом.

Отримані дані, такі, як звуки, медіа можуть зберігатися в бінарному вигляді і займати великий обсяг пам'яті і час на обробку. Збір таких даних традиційними походами не є ефективним. У сучасних комп'ютерних системах використовується розподілений підхід, де існують різні адаптери для збору та передобробки кожних типів даних. Після цього різноманітні дані об'єднуються в реальному часі. На практиці обробка медіа даних виконується тривалий час і вимагає реалізації спеціальних методів (наприклад, методи обробки сигналу звуку, методи обробки зображень), що впливає на тривалість процесу збору даних в цілому.

Як правило, при проектуванні СППР першим виникає питання про те, на основі яких даних ці системи працюватимуть [5]. Ухвалення рішень має ґрунтуватися на реальних даних про об'єкт управління. Тому визначення, які дані мають бути зібрані, є актуальним завданням. Така інформація повинна зберігатися в оперативних базах даних систем оперативної обробки транзакції (OnLine Transaction Processing, OLTP), які забезпечують введення, структуроване зберігання та обробку інформації в режимі реального часу. Насправді для зберігання зібраних даних часто використовуються бази даних, основі яких виконуються аналітичні функції і складні операції об'єднання, звані «сховище даних». Ще одна концепція, нерозривно пов'язана з поняттями СППР і ХД, - концепція оперативної аналітичної обробки даних - OLAP (On-Line Analytical Processing, Інтерактивна Аналітична Обробка, IAO), що використовує методи та засоби для збору, зберігання та аналізу багатовимірних даних з метою підтримки процесів прийняття рішень. Традиційні рішення не підходять під час проектування сучасних OLAP. Однак на основі ряду переваг NoSQL на практиці спеціалістами спробують використовувати бази даних NoSQL для проектування OLAP, які найважче застосовуються, ніж реляційні бази даних. Бази даних NoSQL забезпечують дуже високу пропускну спроможність читання та запису великих даних. За винятком недавніх випусків бази даних Mongo, вони також не надають жодних додаткових базових перетворень даних у базах даних. Наприклад, в IE, Neo, Cassandra або HBase немає функції SUM(). Тим не менш, Neo, Cassandra і Mongo мають дуже високу продуктивність при спробі прочитати відразу кілька записів. OLAP вимагає математичної обробки великої кількості записів у режимі реального часу. Інший недолік при застосуванні бази даних NoSQL для OLAP – підготовка та використання NoSQL у середовищі OLAP буде трудомісткою. Користувач має сам писати методи збору даних. У роботі [3] проведено та аналізовано бенчмарк для OLAP з використанням технологій NoSQL, що порівнює рішення для багатовимірних сховищ. У роботі також показано можливість застосування NoSQL у розробці OLAP.

Особливості та проблеми злиття різнорідних даних у проактивних системах. Приватною проблемою проактивних обчисленнях є проблема управління (або контролю) з використанням інтелектуального аналізу даних. Теоретично управління цей підхід добре відомий як управління з використанням прогнозуючих моделей (Model Predictive Control, MPC). В управлінні підходи називаються прогностичною аналітикою чи проактивним обслуговуванням. Узагальнена схема управління розширюється компонентами, що реалізують прогностичну аналітичну технологію або з використанням прогнозованих характеристик.

Як предметну область у дисертації розглядається область управління транспортною інфраструктурою. Було вивчено можливості реалізації рішень, заснованих на даних для покращення управління міськими процесами, наприклад, аналізу мережі громадського транспорту. Впровадження рішень, що ґрунтуються на даних з використанням прогнозної аналітики, може призвести до зниження витрат в управлінні містами за рахунок мінімізації ризиків виникнення негативних наслідків. У цьому випадку розглядаються дві проблеми: збір різнорідних даних для подальшого аналізу та прогнозування та злиття різнорідних даних.

У принципі, збір даних та злиття даних є основними процесами у рішеннях, керованих даними. Найважливішим фактором тут є час доступу до даних процедур прийняття рішень. Якщо дані, що зберігаються у розподіленому сховищі даних, мають різний формат, необхідно обробити дані під час виконання запиту користувача. Наприклад, якщо користувач запитує інформацію про певну ситуацію на дорозі, що описується відеопотоками та зареєстрованими даними з транспортних засобів, ці дані повинні оброблятися та виводитися як результат запиту. Для вирішення проблеми прогнозування використовуються методи прогнозування на основі експертних знань та методів прогнозування.

В останньому випадку потрібні вибірки даних та інформація про об'єкт для побудови моделей. Якщо пошук оптимальних гіперпараметрів може бути зведений до проблеми комбінаторного пошуку на сітці або описаний як завдання оптимізації,

то збирання та попередня обробка різноманітних даних з різних джерел є неструктурованим завданням і потребує значного часу.

1.4 Дослідження моделей зберігання різнорідних джерел даних у системах з пакетною та потоковою обробкою даних

При розробці системи зберігання та обробки різнорідних джерел даних у проактивних системах з процесами пакетної та потокової обробки даних застосовуються такі ідеї: лямбда-архітектура (λ-архітектура, Lambda Architecture) та архітектура Kappa.

λ -архітектура забезпечує доступ до «великих і швидких» даних і являє собою універсальну, масштабовану та стійку до відмови архітектуру систему обробки даних. Ця архітектура з'явилася та розвивалася у розподілених системах обробки даних на основі рішень компаній Backtype та Twitter. У [10] представлено використання λ -архітектури для розробки системи аналізу настроїв користувачів Twitter. На відміну від традиційних сховищ даних та систем бізнес-аналізу, архітектура яких розроблена для структурованих даних, λ -архітектура дозволяє обробляти напівструктуровані дані, або так звані «сирі дані», тобто без певної λструктури. Слід зазначити, що системи, побудовані з урахуванням λ -архітектури, мають аналізувати дані у пакетному режимі, а й у режимі реального часу. λ -архітектура представлена трьома рівнями: пакетний рівень (Batch Layer), сервісний рівень (Serving Layer), рівень реального часу (Real-Time Layer).

Для реалізації системи обробки потоків даних у роботі застосовується λ -архітектура, яка складається з п'яти рівнів: Message Bus, Batch Layer, Serving Layer, Real-Time Layer та Visualization Layer . На рис.1.4 представлено λ-архітектура для реалізації системи обробки потоків даних у реальному часі.

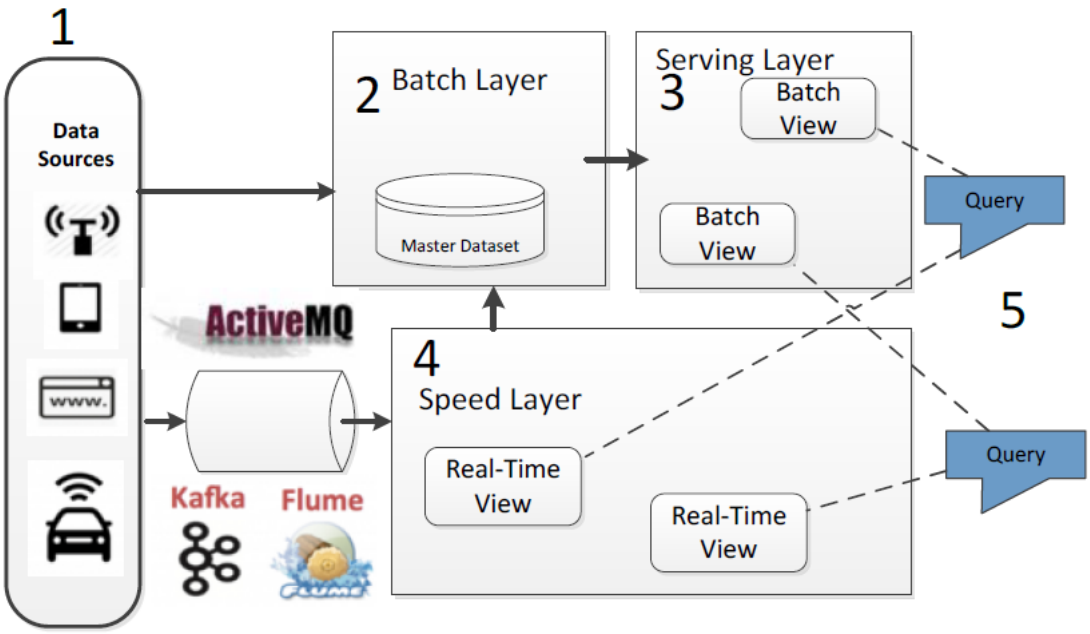


Рисунок 1.4 – λ-архітектура для реалізації системи обробки потоків даних у реальному часі

Вихідні дані (Data Source - GPS data, Log file, Message та інші). Усі дані надходять у систему, пакетний рівень і рівень прискорення.

Пакетний рівень (Batch Layer – Hadoop for Storge, *.Json, *.csv) виконує дві функції, а саме: управління незмінним «сирим» набором даних (Master-набором) та передобробка пакетної передачі набору даних. На цьому рівні можна використовувати Hadoop HDFS для зберігання вхідних даних. Такі дані потім використовуються програмами сервісного рівня для подальшої обробки. Також на цьому рівні дані можуть зберігатись у файлах у форматах *.Json, *.csv, *.xml.

Сервісний рівень (Serving Layer) індексує пакети та обробляє результати обчислень, що виконуються на пакетному рівні. За рахунок індексації та обробки інформації, що надходить, результати вимагають часу. На сьогоднішній день на цьому рівні найкращим вибором є Spark Streaming та Storm.

Рівень прискорення (Speed Layer) відповідає за обробку даних у реальному часі. На цьому рівні використовуються швидкі та інкрементальні алгоритми для обробки потоків поточних вхідних даних у реальному часі. На цьому рівні також можуть використовуватись алгоритми, реалізовані у пакеті Spark Streaming. Потоки вхідних даних повинні зберігатися у сховищі даних, враховуючи високу

інтенсивність надходження. Один із найкращих виборів для вирішення цієї проблеми – Apache HBase.

Рівень візуалізації результатів (Visualization Layer) призначений для візуалізації результатів відповідно до запитів. Будь-який вхідний запит можна обробити шляхом об'єднання результатів від рівня прискорення та сервісним рівнем як реального часу.

Архітектура Kappa була запропонована ще у 2014 році Джеєм Крепсом з компанії LinkedIn. Ця архітектура намагається вирішувати проблеми, які має λ - архітектура. Наприклад, λ-архітектура може бути дорогою в обслуговуванні та розробці, оскільки існує необхідність підтримувати кодову базу для пакетного рівня та рівня швидкості окремо. Архітектура Kappa не є заміною лямбда-архітектури, оскільки основна ідея архітектури Kappa полягає в обробці даних як в режимі реального часу, так і безперервній обробці даних з використанням єдиного механізму обробки потоку. Порівняно з λ -архітектурою, архітектура Kappa складається лише з двох рівнів: швидкісного рівня та сервісного рівня рис.1.5.

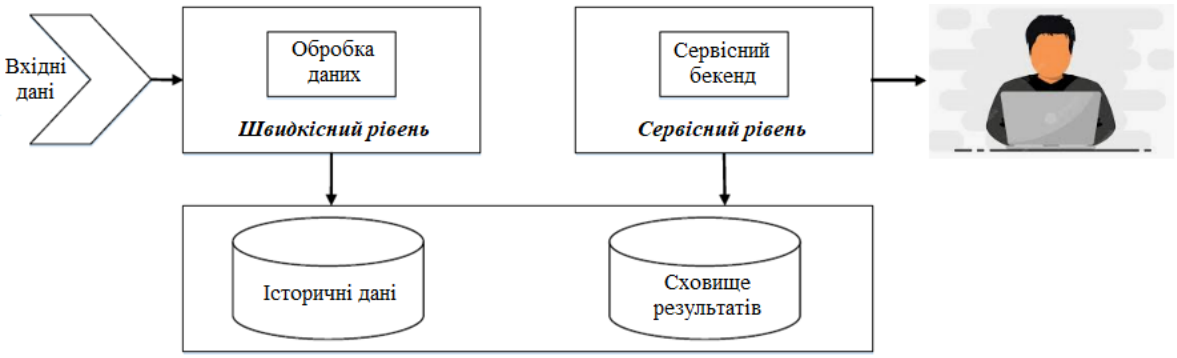


Рисунок 1.5 - Архітектура Kappa

Швидкісний рівень виконує всі функції обробки потоку даних. Один процес обробляє весь потік у реальному часі, тоді як повторна обробка даних виконується лише тоді, коли змінюється певний код механізму обробки потоку. Повторне оброблення виконується за допомогою іншого потокового процесора в реальному часі, який відтворює весь набір даних з усіма попередніми даними.

Сервісний рівень, як і при пакетній обробці, є інтерфейсом для запиту результатів обробки, збережених в деякому сховищі результатів.

Коли одні й самі алгоритми даних застосовують як для обробки реальному часі, так пакетної обробки історичних даних, підтримка однієї й тієї ж кодової бази обох рівнів явно вигідно й економічно ефективно. Важлива перевага архітектури Kappa полягає в тому, що вона потребує менше часу на розробку та підтримку. Недоліком і те, що вона оптимізована для складних завдань пакетної обробки історичних даних [13].

Ці дві архітектури можуть бути реалізовані шляхом об'єднання різних технологій з відкритим вихідним кодом, таких як: Apache Kafka, Apache HBase, Apache Hadoop (HDFS, MapReduce), Apache Spark, Apache Drill, Spark Streaming, Apache Storm та Apache Samza.

Наприклад, дані можуть бути завантажені в архітектури Lambda та Kappa за допомогою системи обміну повідомленнями "публікація-підписка", такої, як Apache Kafka. Зберігання даних і моделей може бути реалізовано за допомогою постійного сховища, такого як HDFS. Пакетна система з високою затримкою, така як Hadoop MapReduce, може використовуватися на рівні пакетної обробки архітектури Lambda для навчання моделей з нуля. Системи з малою затримкою, наприклад, Apache Storm, Apache Samza та Spark Streaming можуть використовуватися для реалізації інкрементних оновлень моделей на рівні прискорення. Ці технології можна використовувати для реалізації рівня потокової обробки в архітектурі Kappa.

В якості альтернативи Apache Spark можна використовувати як спільну платформу для розробки пакетного та швидкісного рівнів у λ-архітектурі. Таким чином, більшість коду може бути розділена між пакетним і швидкісним рівнями. Сервісний рівень може бути реалізований з використанням бази даних NoSQL, такої як Apache HBase, Cassandra, та механізму SQL-запитів, такого, як Apache Drill.

1.5 Огляд технологічних платформ збору, обробки та зберігання даних

Зараз повсюдний розвиток технологій призводить до збільшення обсягів даних, що збираються (у тому числі і неструктурованих).

Обробка таких даних традиційними методами є трудомісткою. Для вирішення низки завдань потрібна обробка даних як реального часу. Наприклад, під час вирішення проблеми планування графіка роботи громадського транспорту місті з урахуванням результатів аналізу дорожніх пробок у часі. Для її вирішення необхідно здійснювати збір даних з транспортних засобів (їхнє положення, швидкість, напрямок) та стан транспортних вузлів. Тому актуальним завданням є розробка системи обробки потоків транспортних даних як реального часу [14].

До цих систем виставляються жорсткі вимоги щодо швидкості роботи, масштабованості, надійності та відмовостійкості. Це веде до потреби у вирішенні наукових завдань, пов'язаних із удосконаленням методів інтелектуальної обробки даних у режимі реального часу. У цьому роботі представлені архітектури технологічних платформ на вирішення завдань підтримки прийняття рішень під час аналізу міських проектів. На основі аналізу було обрано засоби реалізації, а також запропоновано архітектурну модель системи обробки потоків даних у режимі реального часу (ОПДРЧ).

Як говорилося раніше, найпоширенішими рішеннями обробки великих даних нині є Hadoop і Spark. Крім них можна виділити такі відомі платформи, як: S4, Storm, Disco та High-Performance Computing Cluster (HPCC). Hadoop включає такі компоненти: Hadoop Common, HDFS, Yet Another Resource Negotiator (YARN) і MapReduce. Зазначимо дві переваги Hadoop. По-перше, як було зазначено, у ньому використовується розподілена файлова система HDFS надання функцій зберігання даних. Вона дозволяє реплікувати дані між вузлами кластеру. По-друге, у ньому застосовується ієрархія YARN, коренем якої є планувальник ресурсів (ResourceManager). ResourceManager керує всім кластером та здійснює призначення додатків базовим обчислювальним ресурсам. Таким чином, рішення

на основі Apache Hadoop має безліч переваг у порівнянні з традиційними рішеннями, такими як:

- розподілене зберігання даних;

- можливість збирати кластери, що включають недорогі апаратні засоби (звичайні обчислювальні машини);

- забезпечення лінійної масштабованості від 1 до 4000 серверів;

- можливість використання безкоштовного програмного забезпечення з відкритим кодом.

До дефіциту Hadoop варто віднести специфіку моделі MapReduce. Вона забезпечує виконання завдань map (відображення) та reduce (скорочення) для даних, розміщених на кластері. При використанні цієї моделі всі результати як кінцеві, так і проміжні записуються на диск, що веде до суттєвих тимчасових витрат.

Фреймворк Spark може вирішувати ці завдання до 100 разів швидше, ніж Hadoop. Однак, Spark не підтримує розподілену систему зберігання, як Hadoop. Тому можна використовувати Spark як надбудову над платформою Hadoop, використовуючи HDFS для розподіленого зберігання даних. Важливим компонентом Spark є Spark Streaming, що підтримує обробку потокових даних у режимі реального часу. До них відносяться дані лог-файлів журналу сервера (наприклад, HDFS/S3), соціальних медіа (наприклад, Twitter) та різні черги повідомлень, такі як Kafka, Flume. Spark включає наступні компоненти: MLlib – бібліотеку машинного навчання, про яку йшлося вище, що включає різні алгоритми класифікації, регресії, кластеризації, спільної фільтрації, і т.д. для реалізації на кластері; GraphX – бібліотеку до роботи з графами; SparkSQL – компонент Spark, який підтримує запити до даних або через SQL або через Hive Query Language.

У табл. 1.1 представлено порівняння платформ розробки системи обробки потокових і пакетних даних.

Розгляд платформ обробки великих даних у реальному часі має бути виконано з урахуванням таких особливостей. Насамперед у системах підтримки

прийняття рішень має бути передбачена реалізація механізмів обробки даних у режимі реального часу та в пакетному режимі.

Таблиця 1.1 - Порівняння платформ обробки великих даних

Платформи	Розробник	Тип	Опис
Storm	Twitter	Потоковий	Рішення для аналізу великих даних в Twitter
S4	Yahoo	Потоковий	Платфома для обчислень з розподілених потоків даних
Hodoor	Apache	Пакетний	Перший відкритий фреймворк, який реалізує модель MapReduce
Spark	UC Berkeley AMPLab	Пакетний+ Потоковий	Аналітична платформа, яка підтримує в пам’яті різні набори даних, які володіють високою відмовостійкістю
Disco	Nokia	Пакетний	Фреймворк Nokia для розподілу моделі MapReduce
HPCC	LexisNexis	Пакетний	HPC кластер для великих даних
Apache Flink	Stratosphere	Пакетний+ Потоковий	Платформа розподілу оброботки для обчислення з урахуванням стану по необмеженим та обмеженим потокам даних. Платформа розроблена для виконання розрахунків з високою швидкістю роботи і в любому маштабі.

Крім цього, система повинна включати механізми обробки складних подій (англ. complex event processing, CEP), що об'єднує дані з безлічі джерел для максимально швидкого виявлення патернів в даних.

РОЗДІЛ 2 МЕТОДИ ЕФЕКТИВНОГО ЗБОРУ ТА ОБРОБКИ РІЗНОРІДНИХ ДАНИХ У СИСТЕМАХ УПРАВЛІННЯ ТРАНСПОРТНОЇ ІНФРАСТРУКТУРОЮ

2.1 Огляд моделі розподіленого зберігання даних

В результаті системного аналізу процесу збору та структури даних було розглянуто модель зберігання різнорідних даних відповідно до концепції озера даних (далі за текстом модель ХДОД):

$$S = \langle \{DT\}_{\mathrm{i=1}}^{n_{\mathrm{ST}}}, \{SS\}_{\mathrm{j=1}}^{m_{\mathrm{SS}}}, \{E\}_{\mathrm{k=1}}^{p_{\mathrm{E}}}, IS, DS \rangle, \quad (2.1)$$

де, $\{DT\}_{\mathrm{i=1}}^{n_{\mathrm{ST}}}$ – безліч шаблонів даних, n_{ST} – кількість шаблонів даних, $\{SS\}_{\mathrm{j=1}}^{m_{\mathrm{SS}}}$ – методи розбиття різнорідних даних, m_{SS}- кількість типів даних, $\{E\}_{\mathrm{k=1}}^{p_{\mathrm{E}}}$ - безліч виконавців завдань збору даних, IS – метод індексування даних у сховищі озера даних, DS – структура озера різнорідних даних. Таким чином, запропонована модель ХДОД відрізняється від наявних, наявністю шаблонів об'єктів і шаблонів параметрів об'єктів, які дозволяють розподілено зберігати як сирі різнорідні дані, так і структуровані дані відповідно до визначеної схеми, що дозволяє знизити тимчасові витрати на доступ до даних.

Розглянемо компоненти моделі ХДОД, схема якої представлена рис.2.1. Компонент Object Template призначений для шаблонизації деякого об'єкта O_i, де i = 1, ..., n. Кожен об'єкт може мати безліч джерел даних M_{ds}, які створені відповідно до шаблонів Data Source Template. Нехай є певний об'єкт, структура якого буде представлена таким чином:

$$\mathrm{O} = \langle ds_1, ds_2, .., ds_m \rangle \quad (2.2)$$

Кожне джерело даних може мати безліч параметрів з різними типами даних M_p відповідно до шаблонів Parameter Template. Структура такого довільного джерела даних ds представлена у вигляді:

$$ds = \langle p_1, p_2, .., p_k \rangle \quad (2.3)$$

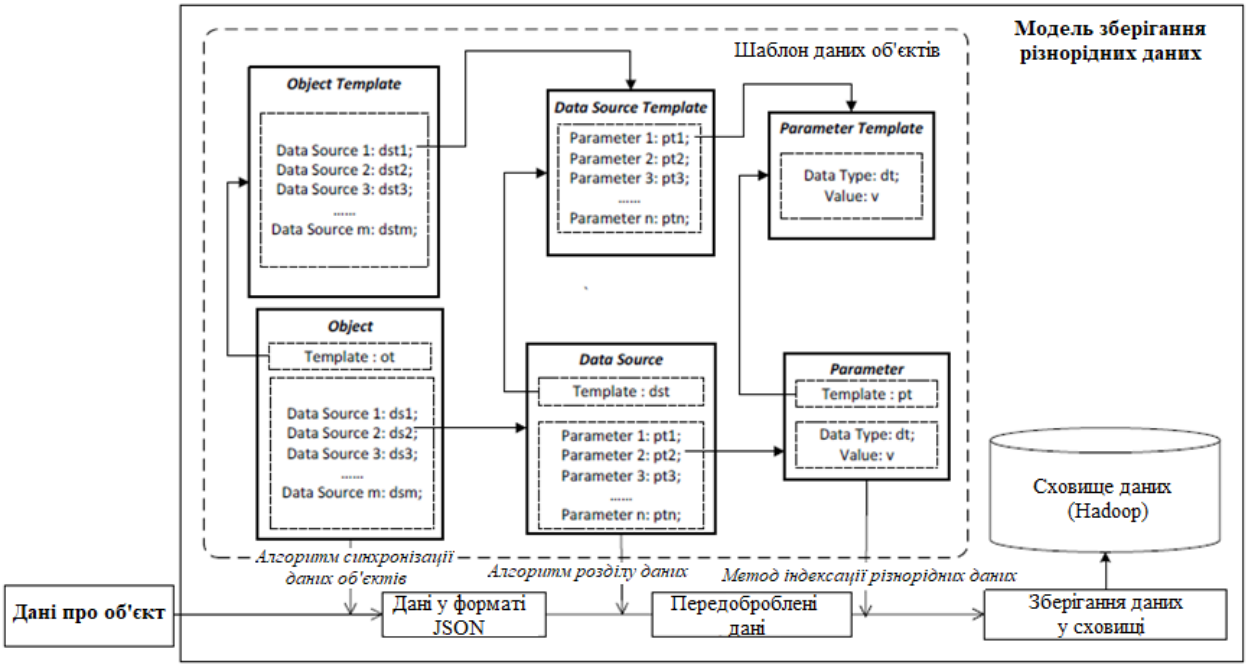


Рисунок 2.1 – Модель зберігання різнорідних даних

В рамках досліджуваної моделі ХДОД розроблено метод збирання та зберігання різнорідних джерел даних відповідно до лямбди-архітектури для потокової та пакетної обробки даних. Розроблено: (i) алгоритм синхронізації різнорідних даних, (ii) алгоритм поділу даних та (iii) метод індексування різнорідних даних.

Розглянемо детальніше запропоновані алгоритми.

Алгоритм синхронізації різнорідних даних описано наступним чином:

1. Отримати пакет даних P.

2. Створити порожні списки $\boldsymbol{Li}<\boldsymbol{t},\boldsymbol{d}>$ для зберігання об'єктів $\boldsymbol{Li}<\boldsymbol{t},\boldsymbol{d}>=\boldsymbol{\phi}$, i = 1, n де n – кількість шаблонів об'єктів, t – час генерації даних d (d як JSONObject).

3. Для кожного елемента пакета отриманих даних виконайте такі кроки:

3.1. Визначити об'єкт за ключовим параметром, що входить до отриманих даних.

3.2. Якщо об'єкт не знайдений у списку $\boldsymbol{Lk}$, який відповідає даному об'єкту за шаблоном, k=1, n то,

1. Створити новий об'єкт JSONObject.

2. Зберегти отримані дані у створений об'єкт JSONObject відповідно до шаблону об'єкта.

3. Додати створений об'єкт до списку L_k.

Інакше:

1. Сортувати всі дані про знайдений об'єкт за часом timestamp T генерації блоку даних об'єктом (або датчиком, встановленим на об'єкті).

2. Зберегти отримані дані у створений об'єкт JSONObject відповідно до шаблону об'єкта.

4. Вивести список L.

Кожен елемент отриманого списку L є блоком даних про об'єкт у форматі JSONObject. Дані про об'єкт можуть бути різнорідними, що в деяких ситуаціях призводить до значних труднощів при подальшому аналізі. Тому для простоти обробки різнорідних даних у цій роботі використано наступний алгоритм поділу різнорідних даних на основі шаблону параметрів:

Алгоритм поділу різнорідних даних.

1. Отримати список L (результат виконання)

2. Створити порожній список R<JSONObject>, який містить дані про об'єкт відповідно до шаблону «Parameter Template».

3. Для кожного елемента списку L виконати:

3.1. Розділити елементи відповідно до шаблону DataSourceTemplate і отримати список Ls.

3.2. Для кожного елемента списку Ls виконати:

3.1. Розділити елементи відповідно до шаблону «ParameterTemplate» та додати до списку Lp.

3.2. Для кожного елемента списку Lp виконати:

3.2.1. Створити новий об'єкт JSONObject.

3.2.2. Зберегти отримані дані у створений об'єкт JSONObject відповідно до шаблону параметрів.

2.2.3. Додати об'єкт до списку R<JSONObject>

4. Повернути список R<JSONObject>.

Для підвищення ефективності збирання та зберігання різнорідних даних запропоновано метод індексування різнорідних даних у ХДОД. Ефективна структура зберігання різнорідних даних ХДОД дозволяє зручно та швидко здійснювати доступ до сховища для пакетної обробки даних. В даний час існує велика кількість аналогічних систем зберігання великих даних, наприклад Hadoop HDFS, S3 Bucker, Amazon Cloud і т.д. Кожен файл, збережений у сховищі різнорідних даних, є парою <t,v>, де t – час генерації даних, v – значення даних.

Створений файл індексується у сховищі відповідно до принципу індексування різнорідних даних у сховищі даних, представленому на рис.2.2.

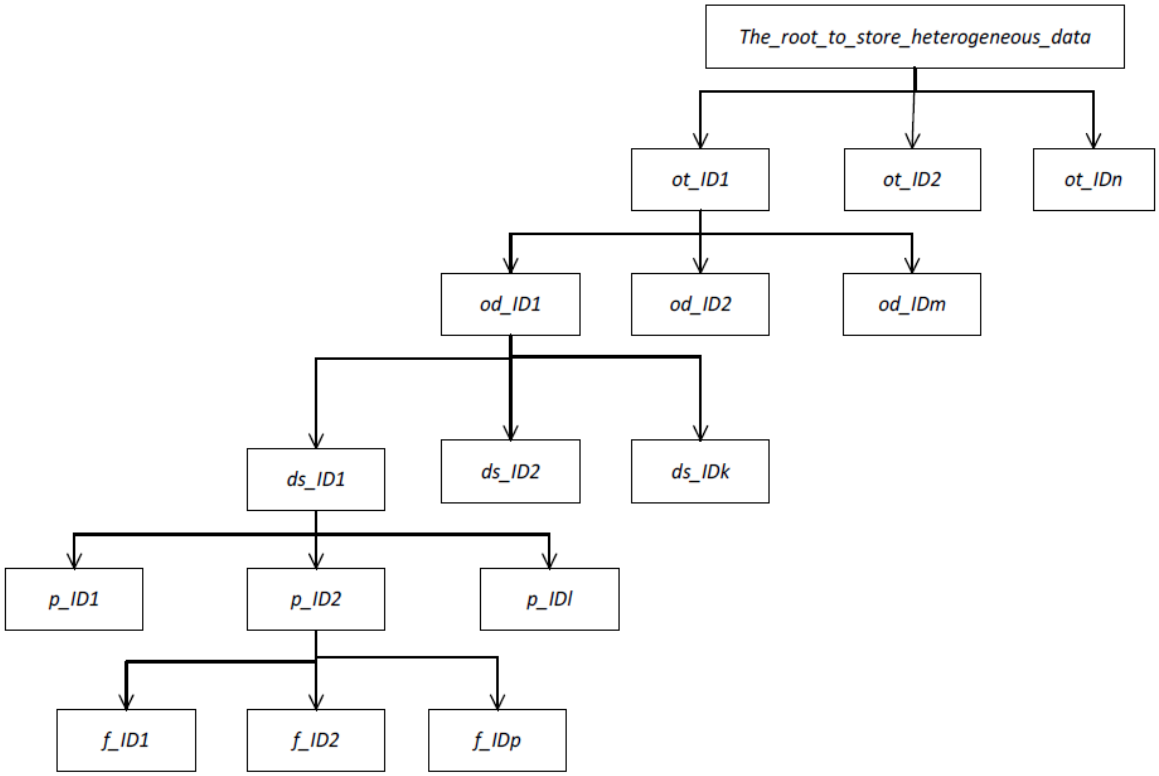


Рисунок 2.2 – Принцип індексування різнорідних даних у сховищі даних

Наведемо роз'яснення щодо досліджуваного способу індексації.

1. The_root_to_store_heterogeneous_data – кореневий каталог для зберігання різноманітних даних;

2. ot_IDn – ідентифікатор шаблону об'єкта n;

3. od_IDm – ідентифікатор унікального об'єкта m;

4. ds_IDk – ідентифікатор унікального джерела даного об'єкта k;

5. p_IDl - ідентифікатор унікального параметра джерела l;

6. f_IDp – ідентифікатор унікального файлу параметра p.

Зауваження. Для простоти та скорочення часу обробки даних файлу надається найменування, що містить час t генерації блоку даних.

Розглянемо застосування алгоритму на конкретній задачі.

Припустимо, що є сховище різнорідних даних, побудованих за принципом озера даних ХДОД, де зберігаються дані про стан подій транспортних засобів. Подієві дані можуть містити інформацію про швидкість, місцезнаходження транспортного об'єкта. Ці дані збережені відповідно до принципу індексування різнорідних даних у ХДОД, описаному вище. Нехай певний елемент транспортної інфраструктури (транспортний засіб, транспортний засіб) записаний у сховище за ідентифікатором od_ID1. Параметри швидкості об'єкта збережені у сховищі у каталозі ds_ID1/p_ID2 у вигляді файлу формату JSON. Імена файлів відповідають тимчасовим міткам timestamp. Потрібно вибрати всі значення швидкості p_ID2 джерела ds_ID1 ТС od_ID1, згенерованого відповідно до шаблону об'єкта ot_ID1, в інтервалі часу [t1, t2]. Для розв'язання цього завдання виконаємо алгоритм:

Алгоритм пошуку даних щодо атрибутів параметрів:

1. Отримати список I імені файлів даних у каталозі /The_root_to_store_heterogeneous_data/ot_ID1/od_ID1/ds_ID1/p_ID2/.

2. Конвертувати імена файлів даних отриманого списку I відповідно до часу timestamp.

3. Відфільтрувати значення, що задовольняють умові $t1 \leq timestamp\ namefilD$

4. Конвертувати імена файлів у списку R (у числовому поданні) у рядки.

5. Створити порожній список P<JSONObject>.

6. Для кожного елемента списку R виконайте такі кроки:

6.1. Створити новий об'єкт JSONObject.

6.2. Отримати дані в каталозі /The_ root_ to_store_ heterogeneous_data /ot_ID1/od_ID1/ds_ID1/p_ID2/R[i], де R[i] – файл номер i.

6.3. Додати отримані дані до списку P<JSONObject>.

7. Повернути список P<JSONObject>.

2.2 Аналіз методів збирання та попередньої обробки різнорідних даних для керування транспортною інфраструктурою

Опис методу збору та попередньої різнорідних даних у ХДОД. Досліджуваний в рамках даної роботи метод включає наступні етапи: визначення необхідної схеми даних для зберігання даних для проактивного управління транспортною інфраструктурою; опис схем джерел даних і параметрів налаштування для збору даних; розробка та впровадження програмного забезпечення, що реалізує алгоритми розподіленої обробки даних; запис даних у ХДОД. Під різнорідними даними розуміються дані, що мають різну структуру при зборі, наприклад сенсорні, подієві дані, відеопотоки про об'єкти транспортної інфраструктури.

У рамках цієї роботи розглядається проблема скорочення часу доступу до даних, отриманих із різних джерел у системах проактивного управління транспортною інфраструктурою. Пропонується метод збору та злиття різнотипних даних, що дозволяє знизити час запиту за рахунок попереднього розподіленого перетворення даних до необхідної схеми.

Розглянемо докладніше запропонований метод.

1. Визначення необхідної схеми даних. Схема для зберігання даних про об'єкти, за якими здійснюється спостереження, описується у форматі sD = <gId, timestamp, (lat, lon), attrD>, де gId – глобальний ідентифікатор об'єкта, унікальний для всіх об'єктів у системі; timestamp – мітка часу, фіксація спостережень; (lat, lon) – координати географічних широти та довготи, що характеризують місце розташування об'єкта в момент часу timestamp; attrD - список (словник) пар ключ-значення, що характеризують ознаки об'єкта, наприклад, "швидкість: 68".

2. Опис джерел даних та налаштування параметрів збору даних. Схема даних на верхньому рівні для довільного джерела даних визначається форматом sC = <sId, acs, attrS >, sId – унікальний ідентифікатор джерела (наприклад, адреса веб-сервісу

на який приходять дані з об'єктів, що рухаються), acs – атрибути у форматі списку «ключ» -значення», що визначають доступ до джерела, attrS - внутрішня схема даних, що отримуються з джерела. Вважаємо, що attrS містить обов'язковий параметр type, який приймає значення зі списку <type:JSON, type:VIDEO>.

3. Встановлення зв'язків між структурами. Для запису даних у необхідну схему їх множини вихідних, для кожного джерела даних задається безліч відповідностей R між атрибутами схеми attrD sd і атрибутами схеми attrS sc.

4. Реалізація алгоритмів перетворення даних. Відповідно до правил перетворення R визначити алгоритми перетворення даних їх однієї схеми в іншу. Слід зазначити, що перетворення можуть бути простими та складними. Якщо передбачені перетворення відеопотоку з механізмами розпізнавання об'єктів, то attrD можуть вказуватися вилучені об'єкти та їх характеристики (метадані). У цьому випадку доцільно включити в схему attrD міру, що характеризує якість розпізнавання.

5. Запис до бази даних. Отримані дані у новому форматі записати до бази даних зі схемою відповідно до формату sD. Таким чином, при виборі даних з бази даних користувачу повертаються всі дані, отримані з різних джерел.

Пропонований метод був реалізований для оцінки дорожньої ситуації на основі подійних даних про переміщення ТЗ (у вигляді лог-файлів) та відеопотоків. Користувач запитує дані про переміщення ТС за певний проміжок часу для певної геолокації з усіх доступних джерел даних, тобто з подієвих файлів та відеопотоків.

Для обґрунтування ефективності запропонованого методу здійснюється порівняння з методом P1, який виконує перетворення сирих даних при реалізації запиту (на вимогу). Показником ефективності є час виконання запиту.

Поліпшення методу збору та попередньої різнорідних даних на мікропотоки. В результаті апробації методу було виявлено напрями його покращення з метою мінімізації часу виконання запитів до ХДОД. Ідея модифікації полягає у поділі потоків даних на мікропотоки та можливістю реалізації у п'ятишарової розподіленої архітектури системи, заснованої на технології Apache Kafka та Spark Streaming.

Метод містить такі кроки:

1. Визначення необхідної схеми даних. Об'єкти, що спостерігаються, описуються набором гетерогенних даних. Схема даних для зберігання таких даних представлена відповідно до формату.

$$sD=\langle gId,timestamp,(lat,long)\rangle$$

де gId – глобальний ідентифікатор об'єкта, який є унікальним для кожного об'єкта, що спостерігається; timestamp – тимчасова мітка, що визначається при спостереженні; (lat, lon) – координати розташування об'єкта у часовій тимчасовій відмітці як пара широти та довготи; attrD – список (словник) пар ключ-значення, що описує функції об'єкта та його значення, наприклад, «speed: 68».

2. Опис джерел даних та налаштувань збирачів даних. Ми використовуємо збирачі даних термінів для програмного забезпечення, що має доступ до джерел даних, і збору даних про об'єкти, що спостерігаються. Високорівневе опис довільних джерел даних визначається форматом.

$$sC=\langle sId,acs,(lat,lon),attrS\rangle$$

де sId є унікальним ідентифікатором джерела даних (наприклад, посилання на веб-службу, звідки надходять дані), acs – це список значень ключа для джерела даних, позначений як sId, attrS – внутрішня схема даних, отриманих від джерела даних. Вважаємо, що attrS містить обов'язковий тип параметра, який набуває значення зі списку <type: JSON, type: VIDEO>.

3. Побудова схем прив'язування даних. На цьому етапі створюється зв'язок між вихідною схемою джерела даних та необхідною схемою. Це посилання представлене у вигляді набору R, що містить пари атрибутів з набору attrD схеми sD та атрибутів з набору attrS у схемі sC. $R=\{ri,j\};ri,j=\langle attrDi,attrSj\rangle$,

Відмітимо, що:

$$\exists rlon:\langle lon^{(sC)},lon^{sD}\rangle \text{ і, } \exists r_{lat}:\langle\ lat^{sD}\rangle.$$

4. Реалізація алгоритмів перетворення даних. Відповідно до налаштувань прив'язки R і алгоритмів $\{\alpha k\}k=1||R||$ реалізовано перетворення даних із вихідної схеми в бажану. $\forall ri,j\in R;\exists\alpha i,j:v(attrDi)\rightarrow v*(attr$

де вказує значення атрибута. Слід зазначити, що перетворення можуть бути простими та складними. У звичайному випадку, наприклад, для типу: JSON може бути встановлена однозначна відповідність двох схем даних. Якщо тип VIDEO, відеопотік перетворюється за допомогою кадрового перетворення з тимчасовою прив'язкою атрибута часу прив'язки та географічними координатами (lat, lon) для кожного кадру. Крім того, attrD задає параметри з ідентифікатором відео, ідентифікатором кадру, ім'ям кадру та вказівкою шляху до збереженого відеофайлу або збереженого зображення (кадру). Якщо для розпізнавання образів використовуються засоби комп'ютерного зору, attrD містить атрибути його використання. Крім того, необхідно включити атрибути attrD, що оголошують оцінку продуктивності розпізнавання.

5. Поділ даних. Схема DS для поділу потоків даних у мікро-потоки визначено.

$$DS\alpha_k = \langle df, \alpha k, \{mdfl\}_{l=1}^{L\alpha_k} \rangle,$$

де df – вихідний потік даних, mdf_l – l потік даних у пам'яті для певного алгоритму α_k, $L_{\alpha k}$ – кількість потоків.

На цьому етапі дані розбиваються на потоки даних, що підлягають обробці в розподіленій архітектурі, відповідно до визначених завдань, наприклад, обчислюючи кількість об'єктів з рівними атрибутами набору атрибутів attrD.

6. Вставка оброблених даних у базу даних. Коли дані перетворюються відповідно до певної схеми, вони вставляються в базу даних. Він дозволяє витягувати різноманітні дані з бази даних без додаткових маніпуляцій з даними.

Розглянемо опис завдання, у якому може бути застосований запропонований метод. Запропонуємо, що всі події, генеровані об'єктами, відповідають шаблону:

{

data : [

"uid" : { Integer } ,

"eventStart": {Long int } ,

"eventEnd": {Long int } ,

"long": {Double } ,

"l a t": {Double } ,

"velocity": {Double } ,

"status": { String }

]}

де uid – ідентифікатор ТС, eventStart – час початку події, eventEnd – час завершення події, lat – довгота розташування об'єкта, lon – широта розташування об'єкта, velocity – поточна швидкість ТС, status – статус ТС (додаткова інформація).

Зауваження. Шаблони можуть бути модифіковані відповідно до вимог конкретної задачі.

Приклад файлу наведено на наступних схемах. Перша схема є файл журналу, отриманий від транспортного засобу з ідентифікатором "f49a09f4-5e79-42cb-8f89-f0df33a08a02" і типом "Taxi".

{

"VehicleID" : "f49a09f4 −5e79−42cb−8f89−f0df33a08a02",

"VehicleType" : "taxi ” ,

"Position" :

{

"latitude":",46.583435",

"longitude ”:”44.892395"

} ,

"timestamp":1465471124373,

"speed": 65.0

}

Друга схема включає дані про транспортний засіб з типом «Bus» та ідентифікатором "42760523-7259-4b6d-b4ea-640a2c3e789c".

{

"VehicleID" : "42760523−7259−4b6d−b4ea−640a2c3e789c",

"VehicleType" : "bus",

"Position" :

{

"latitude" : "46.94521",

"longitude" : "44.93591"

} ,

"timestamp":"2017−06−10 07:13:55",

"speed" : 88.0

}

Для оцінки часу виконання використовувалися різні зміни розподіленої архітектури. Основний метод був реалізований для обробки даних про рух транспортних засобів, що включають подієві дані та відеопотоки. На рис. 2.3 представлений кадр з відеопотоку, що містить ситуацію транспортної інфраструктури, що обробляється.

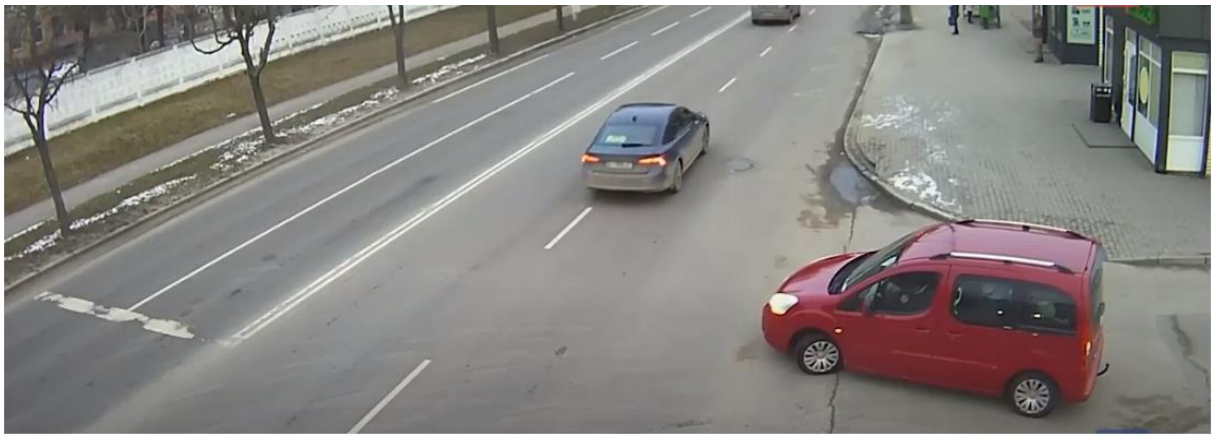


Рисунок 2.3 – Кадр, отриманий із відеопотоку: стан транспортної інфраструктури

Запропонуємо, що завдання полягає в тому, що користувачу необхідно отримати інформацію про кількість транспортних засобів, що проїжджають фіксовану позицію (елемент дорожньої інфраструктури), в заданий період часу [t_0, ,t_k]. Ця інформація має бути вилучена з лог-файлів (переданих із транспортних засобів) та відеопотоків, отриманих з камер відеоспостереження за транспортною інфраструктурою.

2.3 Метод реалізації запитів до різнорідних даних

Опис методу. Для вирішення проблеми уніфікації доступу до даних у цій роботі було спроектовано архітектуру системи обробки різнорідних даних з використанням уніфікованих запитів для ХДОД. На рис.2.4 представлено архітектуру досліджуваного рішення.

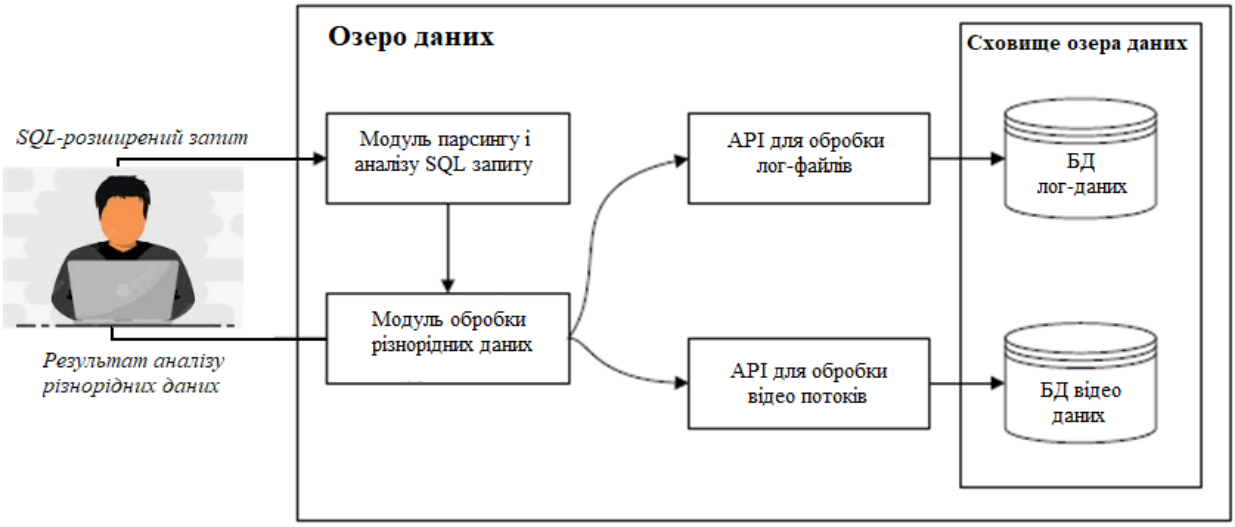


Рисунок 2.4 - Архітектура системи обробки різнорідних даних із використанням уніфікованих запитів

Так, користувач може сформувати запит до різнорідних даних ХДОД, приклад яких представлений у таблиці 3.1.

Лог-файли розміщуються в ХДОД у певних каталогах та мають структуру за аналогією з JSON структурою, приклад якої представлений на рис. 2.5.


```
{
  "VehicleID":"0a955f61-65c3-44f9-8893-f49163225c05",
  "Speed":78.0,
  "Timestamp":"1516661614478",
  "Latitude":"45.07256",
  "Longitude":"43.995274"
}
```


Рисунок 2.5 – Формат лог-файлу

Таблиця 2.1 – Приклад запитів до ХДОД

Запит	Значення запиту
select count from DATALOG, Camera1 where MODIFIED > 1516051435073 and MODIFIED < 1516661699999;	Повернути кількість ТЗ із джерел DATALOG (лог-файл) та CAMERA1 (фрейми відеофайлів), які надсилають дані на сервер в інтервалі $t_{modified}$ >1516051435073 та $t_{modified}$ < 1516661699999
select count from DATALOG, CAMERA2 where longitude > 44 and longitude < 45 and latitude > 45 and latitude < 46;	Повернути кількість ТЗ в певному розташуванні (longitude > 44 and longitude < 45 and latitude > 45 and latitude < 46) із джерел Datalog (лог- файл) та camera2 (фрейм відеофайлів).

Відеопотоки зберігаються як у вихідному форматі, так і у вигляді кадрів (зображень у форматі PNG) у заданому заздалегідь каталозі. Загальна структура зберігання даних представлена рис. 2.6.

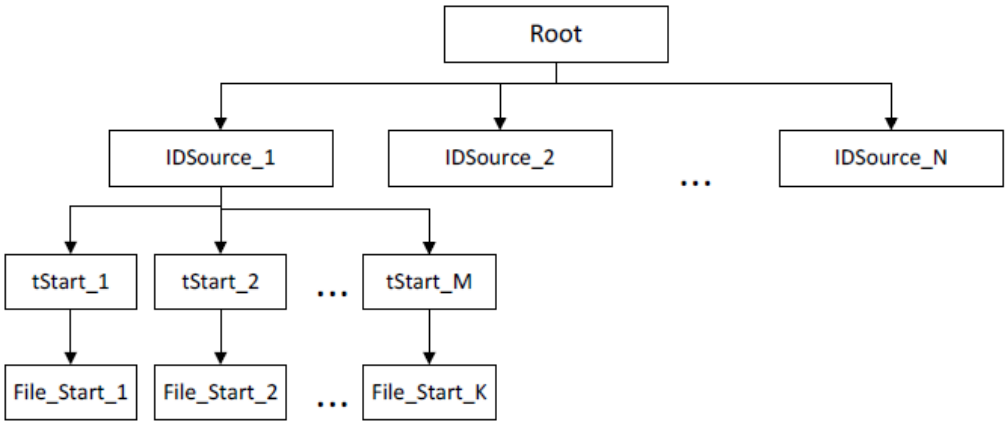


Рисунок 2.6 – Загальна структура зберігання даних

Для аналізу та аналізу SQL запиту необхідно розробити механізм аналізу відповідно до деякої уніфікованої граматики. В результаті функціонування такого механізму формуються блоки даних, які надалі обробляється методами (у тому числі методами інтелектуальної обробки даних). Наприкінці результати обробки агрегуються та виводяться користувачеві.

Пропонований метод реалізації запитів до ХДОД з уніфікованою граматикою SQL-подібних запитів включає наступні кроки.

1. Побудова уніфікованого запиту SQL-подібною мовою.

2. Розбір запиту, сформованого SQL-подібною мовою.

3. Формування запиту до ХДОД.

4. Агрегація та виведення користувачеві результатів на запит до сховища різнорідних даних.

Для реалізації методу необхідне вирішення наступних завдань.

Завдання 1. Створення граматики, що розширює мову SQL на формування запитів до ХДОД. На цьому кроці має бути побудована граматика, що відповідає структурам SQL-подібних запитів для вибірки даних, визначено необхідні лексеми та токени для побудованої граматики.

Завдання 2. Побудова ефективного парсера для розпізнавання SQL-подібних запитів відповідно до розробленої граматики. На даному етапі може бути використана бібліотека ANTLR 3.0 для створення коду парсера.

Результатом розв'язання задачі – програмне забезпечення для розбору SQL-подібних запитів.

Завдання 3. Реалзація програмних модулів для отримання та обробки різнорідних даних (відеофайли, лог-файли) з ХДОД.

Розглянемо пропоновану граматику на формування запитів до різнорідним даним в ХДОД.

Загальна граматика інструкції, що реалізує вибір даних має такий вигляд:

query : select_stmt where_stmt;

де, select_stmt - оператор вибору,

where_stmt – оператор умов фільтрації результатів;

Візуальна нотація інструкції SELECT представлена рис. 2.7.

Рисунок 2.7 – Схема інструкції SELECT

Інструкція SELECT представлена таким чином:

select_stmt

: 'select' 'count' from_stmt

де FROM_STMT – оператор вибору джерел даних

from_stmt

: ('from' directoryname (',' directoryname)) ?*

count – оператор підрахунку кількості об'єктів.

Схема інструкції SELECT із оператором COUNT представлена на рис.2.8.

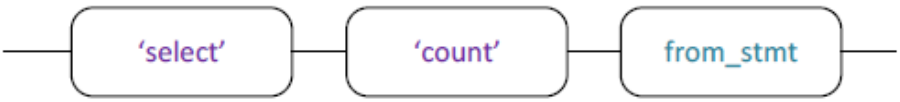


Рисунок 2.8 – Схема інструкції SELECT із оператором count

На рис. 2.9 представлено схему інструкції SELECT з оператором from_stmt вибору джерел даних.

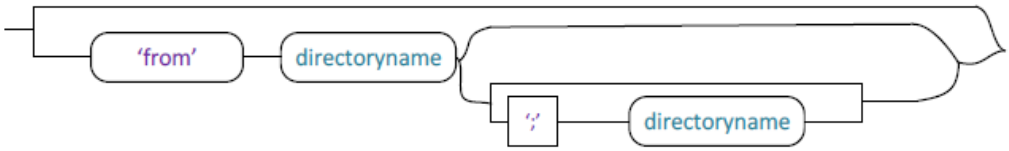


Рисунок 2.9 – Схема інструкції SELECT із оператором from_stmt

Оператор умови результату має вигляд:

WHERE_STMT

: ('where' clause ('and' clause))?*

Схема оператора WHERE_STMT представлена рис. 2.10.

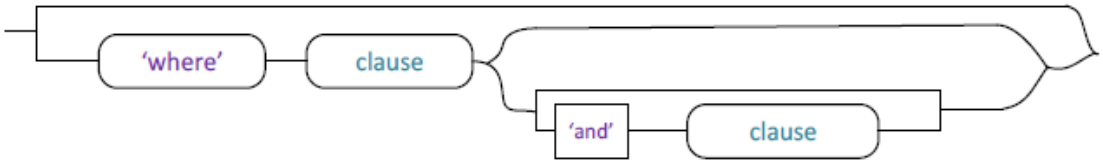


Рисунок 2.10 – Схема оператора where_stmt

Вираз CLAUSE може містити інформацію про назву файлу, тип файлу або час модифікації/додавання файлу, з яких витягнуті дані рис. 2.11.

clause: file_name | pattern | modified;

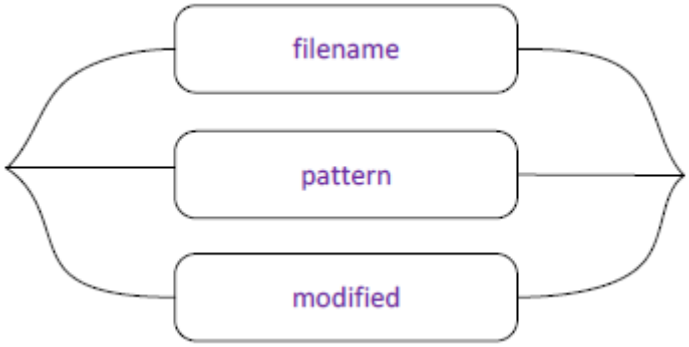


Рисунок 2.11 – Схема виразу CLAUSE

Назва файлів може бути послідовністю літер та/або цифр.

file_name : 'file' '=' IDENT ;

*IDENT: LETTER (LETTER | DIGIT) *;*

Вираз IDENT ілюструється схемою, що на рис. 2.12.

Рисунок 2.12 – Схема вираження IDENT

Вираз LETTER містить послідовність літер.

LETTER: 'a'..'z'|'A'..'Z';

Вираз DIGIT містить послідовність цифр.

DIGIT: '0'..'9';

У представленій граматиці пропускаються символи пропусків, перенесення рядка та табуляції.

WS: [\t\r]+ -> skip;

Символ «*» означає, що висловлювання умов результатів може бути послідовністю умов із логічним зв'язком AND.

Вираз DIRECTORYNAME містить найменування джерел даних, з яких вилучаються дані. Цей вираз містить послідовність лише літер та цифр. Вираз clause містить умову фільтру результатів.

Для реалізації запропонованої граматики було розроблено програму мовою Java з використанням фреймворку ANTLR 3.0. Всі змінні, що використовуються в граматиці, оголошені в тезі @members рис. 2.13.

```
@members{
   public List directories = new ArrayList();
   public List clauseses = new ArrayList();
   public HashMap<String, String> hmapclause = new HashMap<String, String>();
   public Hashtable table = new Hashtable();
}
```

Рисунок 2.13 – Тимчасові змінні, які використовують у розробці граматики

Розглянемо послідовність операції генерації механізмів аналізу запитів, складених відповідно до граматики. Усі Java-операції описані в тегах {.}. Генерація Java-коду (перетворення граматики у вигляді *.g4 на парсер) відбувається за допомогою фреймворку ANTLR 3.0. Для ручної генерації парсера можна використати таку команду:

java -jar antlr-3.2.jar -o VFSQL.g4

В результаті буде отримано програмний код (у вигляді класів) для подальшого використання в модулі аналізу запитів.

VFSQL.tokens – службовий файл із списком токенів граматики;

VFSQLBaseListener.java – абстрактний клас для кожного правила парсера;

VFSQLLexer.java – клас лексера;

VFSQLLexer.tokens – службовий файл зі списком токенів лексера;

VFSQLListener.java – абстрактний клас;

VFSQLParser.java – клас парсера.

2.4 Метод з елементами доповненої реальності

Реаліхація систем керування транспортною інфраструктурою з елементами доповненої реальності (Augmented Reality, AR) потребує аналізу обох типів даних: відеопотоків (розпізнавання або сегментації зображень) та подійних даних, отриманих з ТС. Інформація має бути узгоджена та надана для кінцевого користувача для подальшого прийняття рішень. Час обробки запитів та надання результатів користувачеві є критичним показником. У роботі пропонується підхід для об'єднання даних про ТЗ джерел аналізу та позиціонування у системі управління транспортною інфраструктурою з елементами доповненої реальністю. Відмінними рисами методу є: (1) механізм уточнення положення транспортних засобів на основі аналізу координат спостерігача, камери та об'єкта, що рухається; (2) ідентифікація об'єкта з використанням підходів розпізнавання зображень; (3) обробка подійних даних, одержаних від транспортних засобів.

Постановка задачі. Нехай є набір транспортних засобів V, зареєстрованих у базі даних SD, де SD це статична база даних, що містить загальну інформацію про ТС. До кожного транспортного засобу зберігаються такі дані: код VIN, ідентифікатор, інформація про власника транспортного засобу та додаткова дорожня статистика (порушення, аварії). Кожен транспортний засіб відправляє пакет даних DPv(t) дискретний період часу в динамічну базу даних DD. На рис. 2.14 показана структура даних, що передаються.

```
{ data
     [
     "VehicleID" : {String} ,
     "DriverID"  : {String} ,
     "eventStart": {Long int } ,
     "eventEnd"  : {Long int } ,
     "longitude" : {Double} ,
     "latitude"  : {Double} ,
     "velocity"  : {Double} ,
     "status"    : {String}
     ]
}
```

Рисунок 2.14 - Схема переданих даних на основі формату JSON

Припустимо, що існує спостерігач о, який має своє місце розташування ($long_o$,lat_o). Спостерігач має намір отримати додаткову інформацію про ТЗ, що проїжджає. Ця інформація має бути отримана із SD. Вважаємо, що відеокамера, формалізована як C=($long_c$,lat_c) встановлена в певному місці транспортної інфраструктури (ділянка дороги RS), центр якого має координати задані довготою at_c. Камера формує відеопотоки частини доріг із усіма учасниками дорожнього руху. Припустимо, що розташування камери таке ж, як і місце розташування ділянки ділянки дороги, що спостерігається. Відеопотік VS_c передається на сервер для подальшого аналізу. Припустимо, що $long_o$=$long_c$,lat^o=lat_c.

Використовуючи відеопотоки з камери VS_c, дані про місцезнаходження спостерігача, дані, зібрані з ТЗ, та дані про транспортні засоби, що зберігаються в SD, необхідно надати відповідні дані про розпізнаний транспортний засіб як з SD так і з DD до спостерігача.

На рис. 2.15 представлено екземпляр переданих даних відповідно до визначеної схеми. VehicleID - унікальний номер транспортного засобу, який зберігається як у статичній базі даних StaticDatabase, так і в динамічній базі даних DynamicDatabase. Параметр VehicleID вибирається з бази даних демонстрації користувачеві. DriverID – це тимчасові дані про водія (або власника) транспортного засобу.


```
{
"VehicleID":"0a955f61-65c3-44f9-8893-f49163225c05",
"DriverID":"123456",
"Speed":78.0,
"Timestamp":"1516661614478",
"Latitude":"45.07256",
"Longitude":"43.995274"
}
```


Рисунок 2.15 - Приклад даних, що передаються від транспортного засобу до сервера

Зберігання даних реалізовано відповідно до аналізу у попередньому підрозділі моделі, структура зберігання подійних даних представлена на рис. 3.16. Ім'я файлу – це час t генерації запису в блоці подійних даних. Час t має формат

тимчасової мітки UNIX. Кожен файл містить дані про ТЗ. Коли дані з ТС засобу відправляються на сервер, створюється файл подійових даних і індексується відповідно до стратегії індексування рис.2.16, де IDSource є унікальним ідентифікатором об'єкта (джерела), що розглядається, tstart – це час генерації файлу File_Start в JSON форматі, у якому зберігаються дані про ТЗ.

Розглянемо метод аналізу різноманітних даних як реального часу у системах управління транспортної інфраструктурою з елементами доповненої реальності. Пропонований метод містить такі кроки, що виконуються у певному дискретному часі τ (час початку події).

Припускаємо, що зображення з камери VS_c безперервно передається на сервер, сервер розбиває відеопотік на зображення через фіксовані інтервали часу Δt. Дані транспортного засобу $DSlg$ передаються на сервер в момент часу Tv.

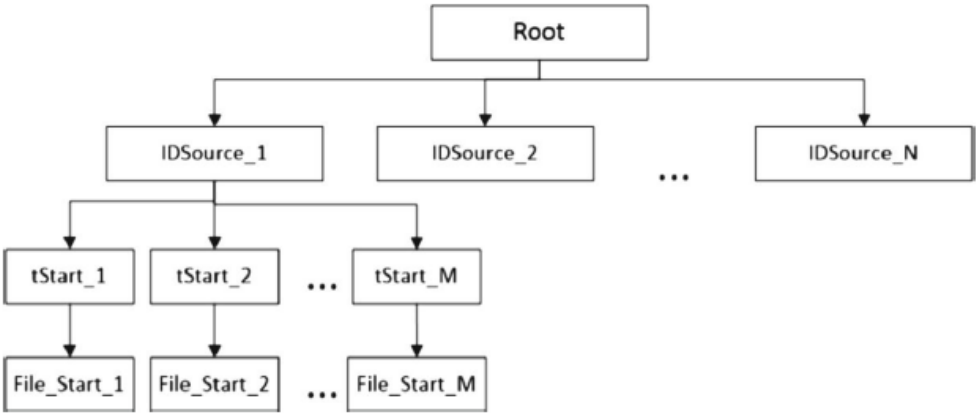


Рисунок 2.16 – Структура зберігання даних про ТЗ

Етап 1. Спостерігач, який використовує AR-систему, починає захоплювати зображення ділянки дороги у певний час τ (час початку події).

Етап 2. Пакет даних DP про запуск події надсилається на сервер. Пакет містить дані про спостерігача (його місцезнаходження) та час початку події τ.

Етап 3. У момент часу надходження пакетів даних на сервер, формується та надсилається запит відеопотоку VSc відповідно до місця розташування спостерігача та місця розташування камери.

Етап 4. Запускається процедура розпізнавання транспортного засобу за зображеннями, витягнутими з відеопотоків VS_c в інтервалі часу $[\tau-\varepsilon,\tau+\varepsilon]$.

Етап 5. Якщо транспортний засіб розпізнаний, виконуються такі дії.

1). Зберегти тимчасову мітку $\tau*$ розпізнаного МС.

2). Створити та надіслати запит $R1$ до DD, що містить час розпізнавання τ* і пару $(long_c,lat_c)$.

3). Вибрати з DD всі ТЗ, які мають однакове місце розташування в інтервалі часу $[\tau-\varepsilon,\tau+\varepsilon]$. Однакове місце розташування означає, що L < d, де d - зумовлений поріг. Значення L обчислюється таким чином:

$$L=R\cdot c,$$

де R - радіус Землі (константа)

$$\mathrm{c} = sin^2\big((lat_v - lat_c)/2\big) + cos(lat_v)\cdot sin^2\big((lat_v - lat_c)/2\big)$$

4). Якщо ТС вибрано на попередньому кроці з ідентифікатором ID*, створити та надіслати запит $R2$ на SD. Цей запит містить унікальний ідентифікатор ID* ТС.

5). Отримати та підготувати пакет даних відповіді DP_r, що містить інформацію про МС з ідентифікатором ID*.

Етап 6. Надіслати пакет даних DP_r кінцевому спостерігачеві.

Якщо бази даних DD або SD не мають даних про ТС (відповіді на запити порожні), DPr містить повідомлення типу «No data available for the vehicle».

РОЗДІЛ 3 РЕАЛІЗАЦІЯ АРХІТЕКТУРИ СИСТЕМИ ЗБОРУ ТА ОБРОБКИ ДАНИХ

3.1 Реалізація фреймворку генерації подій транспортної інфраструктури для оцінки ефективності досліджуваних методів

Концепція фреймворку генерації подій транспортної інфраструктури на основі Apache Kafka. У дипломній роботі досліджено та обрано фреймворк EVGEN (скорочено EVent GENerator) для генерації потоку даних подій транспортної інфраструктури для тестування запропонованих методів обробки даних [15]. Фреймворк дозволяє створювати потік JSON-даних відповідно до попередніх налаштувань: формат даних (структура) та інтенсивність потоку даних. Архітектура системи базується на основі моделі обміну повідомленнями публікації-підписки (Publish-Subscribe), надалі за текстом PS-модель. Фреймворк EVGEN був використаний для створення даних про транспортну інфраструктуру для оцінки проблеми транспортного трафіку.

Для проектування фреймворку генерації даних подій було проведено аналіз існуючих генераторів даних та подій, у результаті можна зробити висновок, що майже всі генератори використовуються для генерації випадкових даних/подій формат XML. Такий спосіб не підходить для створення вхідних подій для системи реального часу, наприклад, системи обробки реальних транспортних подій. Транспортні події можуть бути різними та можуть бути отримані з різних джерел (з датчиків транспортних засобів, зі смартфонів, наприклад, GPS дані тощо). Крім того, використання даних у форматі XML призводить до збільшення розміру даних при їх передачі. У цьому роботі використовується формат JSON. Дані у форматі JSON складніше обробити, ніж дані у форматі XML, але розмір даних у форматі JSON менший за розмір даних у форматі XML з однаковою інформацією [16]. Також показана висока ефективність передачі даних у форматі JSON. Таким чином, при передачі даних у мережі часто використовується формат JSON.

Для розробки розподіленої системи генерації даних подій проведено аналіз PS-моделей. Існують такі відомі PS-моделі: PS-модель Apache Kafka, Apache Flume, Google Cloud Pub/Sub. Apache Kafka та Flume забезпечують надійну, масштабовану та високопродуктивну обробку даних великих обсягів. Однак Apache Kafka - це більш універсальна система, де кілька виробників і передплатників можуть ділитися кількома темами. Навпаки, Flume – це спеціальний інструмент для подальшого збереження даних HDFS. В результаті аналізу було обрано PS-модель у системі Apache Kafka для розробки та проектування системи генерації даних подій.

Apache Kafka – це розподілена система обміну повідомленнями. Вона спочатку була розроблена в корпорації LinkedIn, а пізніше стала частиною проекту Apache. Apache Kafka відрізняється від традиційної системи обміну повідомленнями тим, що Apache Kafka розроблена як розподілена система, яку дуже легко масштабувати, вона забезпечує високу пропускну здатність як для публікації, так і для передплати; і автоматично балансує споживачів і зберігає повідомлення на диску які можуть використовуватися для пакетної обробки (ETL) на додаток до реального часу. Архітектура Apache Kafka складається з наступних компонентів: виробники (Producers) – це додатки, які створюють дані та надсилають їх брокерам Apache Kafka для поширення серед споживачів; споживачі (Consumers) – це додатки, які підписуються на теми (Topics) та зчитують дані з брокерів; брокери (Brokers) представляють сервери (обчислювальні машини) для обробки черг даних від виробників та подальшого відправлення їх споживачам з урахуванням реплікації даних; У Kafka записи (дані) зберігаються в розділах, які являють собою фізичні слоти пам'яті теми; теми (Topics) – категорії повідомлень, на які споживачі підписують та зчитують дані; Zookeeper - це централізована служба підтримки інформації про конфігурації, іменування, забезпечення розподіленої синхронізації і надання групових служб. Розглянемо приклад використання Apache Kafka у розподіленій системі збору, передачі та обробки транспортних даних. Виробники збирають дані з різних транспортних засобів через веб-сервіси, і отруюють їх на обчислювальні машини в кластері за однією або

декількома темами, де дані зберігаються в розділах. Кожен споживач підписується на певні теми і отримує дані по черзі для наступного етапу обробки. На рис.3.1 показано основну концепцію Kafka від Hortonworks.

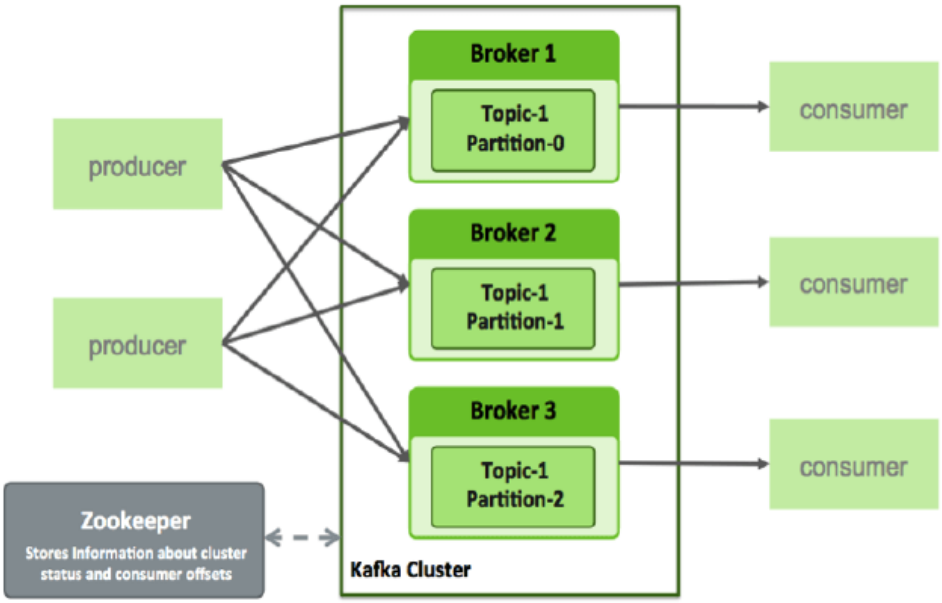


Рисунок 3.1 – Основна концепція Apache Kafka

В даний час LinkedI та інші компанії використовують Apache Kafka для читання каналів даних та організації їх у теми. Apache Kafka – популярна система, яка використовується в системах обробки даних великої кількості фінансових компаній, таких як Yahoo, Oracle, Pinterest, Shopify і т.д. Наприклад, веб-сервер Clickstreams, де Apache Kafka використовується як шина повідомлень для зберігання згенерованих подій.

Структура генерованих подій. Насправді події формуються різними елементами транспортної інфраструктури, отже дані надходять із різних джерел із різною структурою. При розробці фреймворку розглянуто два типи подій, що генеруються елементами транспортної інфраструктури: події транспортного засобу (ТЗ) у форматі подійових файлів та відеозображення перехресть у форматі відеопотоків. Події ТЗ формуються генераторами, написаними мовою Java, і мають шаблон:

```
{
data : [
"uid"          :  {Integer},      // Object ID
"eventStart"  : {Long int},      // Starting time of the event
"eventEnd"   :  {Long int},     // End time of the event
"long"         : {Double},       // Longitude of the object space
"lat"          : {Double},       // Latitude of the object space
"velocity"     : {Double},      // The current velocity of the object
"status"       : {String},       //  Status of the object
"country"      : {String}        // Country
        ]
}
```

Рисунок 3.2 – Шаблон події ТЗ які формуються генератором

Події ТЗ містять такі властивості: ObjectID - унікальний ідентифікатор об'єкта (автомобіля), eventStart - час початку події, час завершення події - eventEnd, long - довгота розташування об'єкта, lat - широта розташування об'єкта, velocity - поточна швидкість об'єкта, status - стан об'єкта та country – місто, де об'єкт знаходиться.

Події, які реєструються на транспортному перехресті, відповідають наступною схемою:

```
{
data :
[
 uid : {Integer} // ID transportation intersections
 Long: {Double} // Longitude of the object (Node)
 Lat : {Double} // Latitude of the object (Node)
 Count: {Integer} // The number of vehicles in the transportation intersections
 Status : {String} // Status of the transportation intersections
]
}
```

Рисунок 3.3 –Схема подій, які реєструються на транспортному перехресті

Опис фреймворку EVGEN. Було розроблено архітектуру фреймворку для генерації подій транспортної інфраструктури. Архітектура фреймворку реалізована з використанням мови Java для створення виробників та мови Python для створення споживачів. На рис.3.4. представлена пропонована архітектура фреймворку для створення подій.

Пропонована архітектура складається з двох частин: EventDataGenerators і EventDataGenerationConsumers.

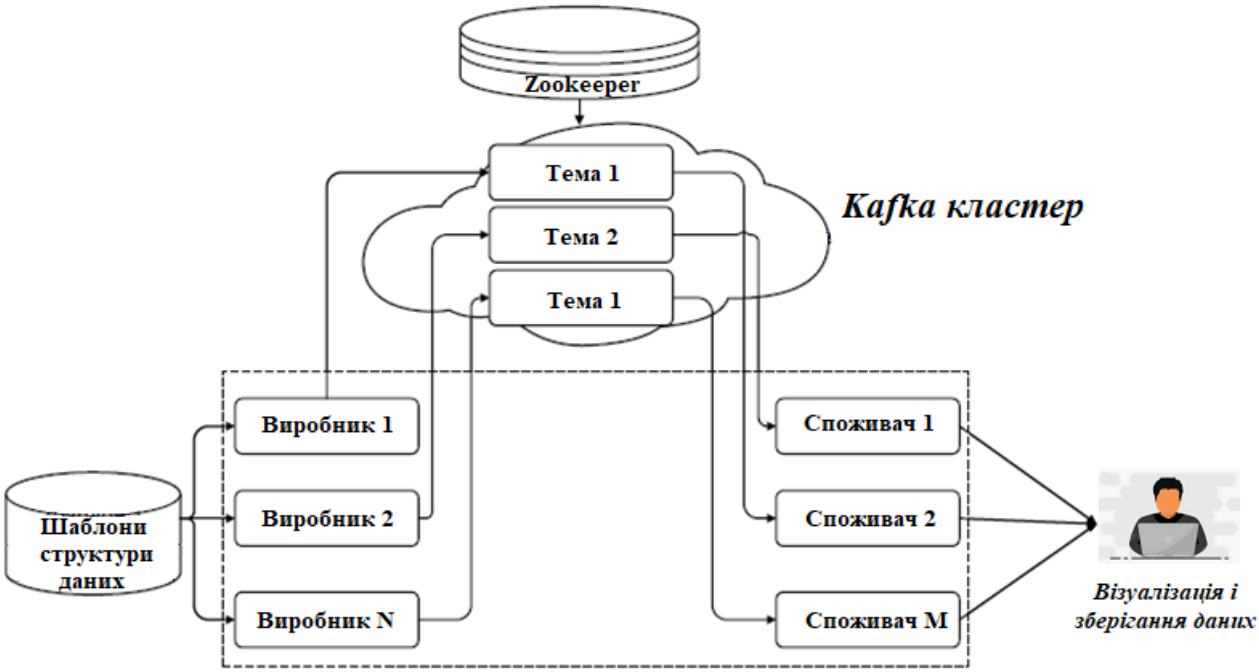


Рисунок 3.4 – Архітектура пропонованого фреймворку генерації подій у режимі реального часу

Розглянемо етапи використання фреймворку:

Етап 1. Визначення структур подій для генерації та настроювання конфігурації розподіленої системи обміну повідомленнями Apache Kafka. На цьому етапі буде визначено кількість виробників N_p для генерації даних, кількість споживачів N_c , кількість брокерів N_b та теми (топіки) для передачі кожних типів подій. Крім того, для того, щоб система обробляла дані великого обсягу, необхідно налаштувати системні параметри брокерів, такі як message.max.bytes – максимальний розмір блоку даних, який може прийняти брокер, log.segment.bytes – розмір файлу даних Kafka, replica .fetch.max.bytes – максимальний розмір блоку даних, що брокер може видати.

Етап 2. Генерація подій виробниками з розроблених структур даних та отримання споживачами генерованих подій за темами для подальших операцій обробки даних.

Модулі EventDataGenerators, написані на мові Java, включають реалізацію декількох генераторів, які формують блоки даних про різні події ТС відповідно до JSON шаблонів. Ініціалізація генератора здійснюється методом GeneratorKafka.initialize(). Генератор формує потік подій (блоки даних подій), що записуються в кластер серверів Apache Kafka. Події генеруються методом GeneratorKafka.Generator(). Публікація на теми здійснюється методом GeneratorKafka.PublishEvent(event). Події приймаються споживачами на певні теми відповідно до черги.

Метод генерації синтактичних включає кроки:

Крок 1. Ініціалізація налаштувань для сервера Kafka та Zookeeper Server.

Крок 2. Створення виробників.

Крок 3. Генерація даних журналу у форматі файлу JSON.

Крок 4. Перетворення JSON файлу на текст і надсилання блоку даних відповідної теми на сервер Kafka.

Крок 5. Зупинення роботи генератора у випадковий час (Random Time) та перехід до кроку 3.

EventDataGenerationConsumers також складається зі споживачів, які отримують події, що генеруються, відповідно до тем і обробляють їх. Кожен споживач підписується на певну тему, використовуючи метод ConsumerKafka.Subscribe (host, topic) і зчитує події в черзі за правилом FIFO (First In – First Out) з використанням методу ConsumerKafka.Message.Value. Модулі аналізу блоку даних формату JSON і рендерингу отриманих подій, що генеруються, також реалізовані в цій частині. Модулі були написані на Python, використовуючи бібліотеки PyKafka, mplleaflet та scikit-learns.

Пропонований метод у модулі EventDataGenerationConsumers включає наступні кроки:

Крок 1. Підключення до сервера та передплата теми (наприклад, topic_sensor_1).

Крок 2. Очікування даних, які у підписану тему чи читання історичних даних.

Крок 3. Отримання даних теми, на яку було здійснено підписку з черги.

Крок 4. Візуалізація потоків подій.

У EventDataGenerationConsumers PyKafka використовувалося підключення до кластера Kafka на локальному комп'ютері. Споживач створює екземпляри topic.get_simple_consumer() для отримання подій із теми, на яку підписано.

Тестування досліджуваного фрейворку генерації подій транспортної інфраструктури. Досліджуваний фреймворк протестували на локальній машині з використанням різних варіантів використання. На рис. 3.5 показаний скріншот логів згенерованих подій виробниками та логів подій, одержаних подій споживачами.

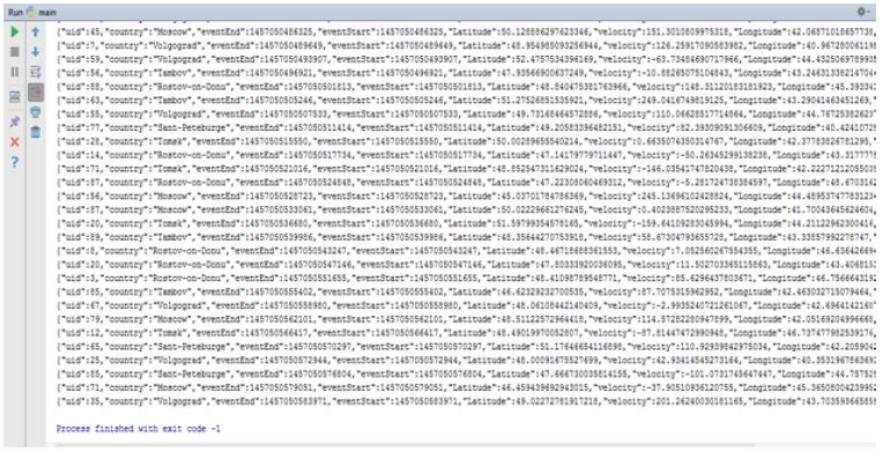


Рисунок 3.5 - Структура лога-файлу згенерованих/отриманих подій

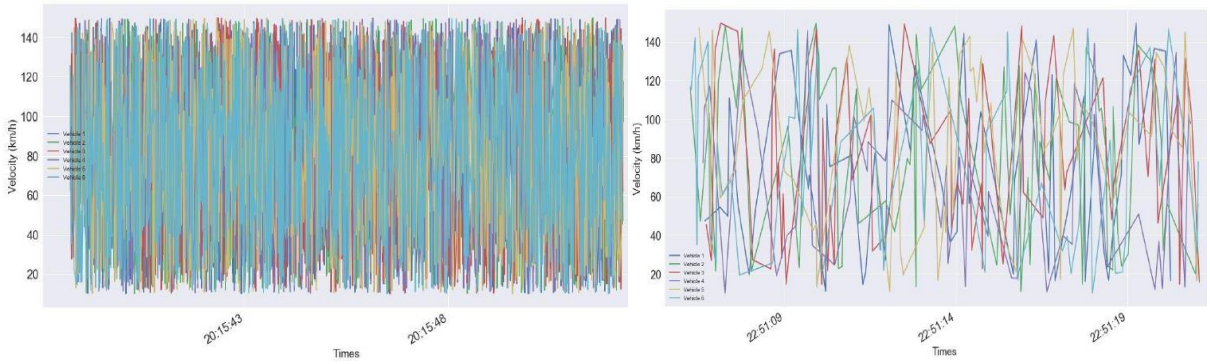
Для аналізу функціонування споживачів створено кілька модулів візуалізації даних про транспортні події, отримані з Kafka. Візуалізація дозволяє наочно уявити поточний статус ТЗ, швидкість ТЗ та інші характеристики.

На рис.3.6 представлені результати моделювання параметра швидкості транспортних об'єктів у реальному часі з різною інтенсивністю подій, що входять. На рис.3.6а наведено дані швидкості для всіх транспортних засобів, отриманих з високою інтенсивністю: ~912 подій за секунду. На рис.3.6б показані швидкість ТЗ, отриманих в режимі з низькою інтенсивністю надходження подій: ~ 47 подій в секунду. Кожна лінія графіка є зміною швидкості МС у реальному часі.

Візуалізація згенерованих геопросторових даних виконана за допомогою бібліотеки mplleaflet для мови програмування Python. Для кластеризації

геопросторових даних використано алгоритм K- середніх з розрахунком відстаней між транспортними об'єктами. У роботі використано бібліотеку scikit-learn для кластеризації геопросторових даних. На малюнку 4.5 показаний результат кластеризації геопросторових даних, зібраних із GEA. Візуалізація геопросторових даних також дозволяє бачити місця у місті, де багато автомобілів. Це допомагає зрозуміти історію руху мешканців міста для дослідження напрямків розвитку транспортної інфраструктури.

Таким чином, архітектура фреймворку дозволяє генерувати події транспортної інфраструктури з різною частотою.



а) б)

Рисунок 3.6 – Візуалізація даних про швидкість МС з інтенсивністю потоку подій а) l = 912 подій на секунду; б) l = 47 подій на секунду

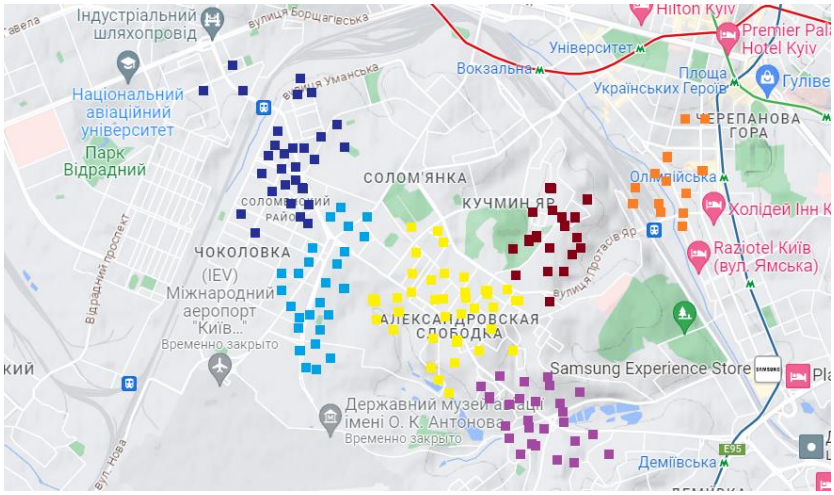


Рисунок 3.7 - Візуалізація згенерованих геопросторових даних із використанням бібліотеки Python mplleaflet

3.2 Архітектура технологічного стека для реалізації досліджуваних методів

Для апробації та тестування ефективності досліджуваної моделі та методів було використано архітектуру проактивної системи управління транспортною інфраструктурою на основі лямбда-архітектури системи розподіленої обробки даних. Архітектура рішення представлена рис.3.8.

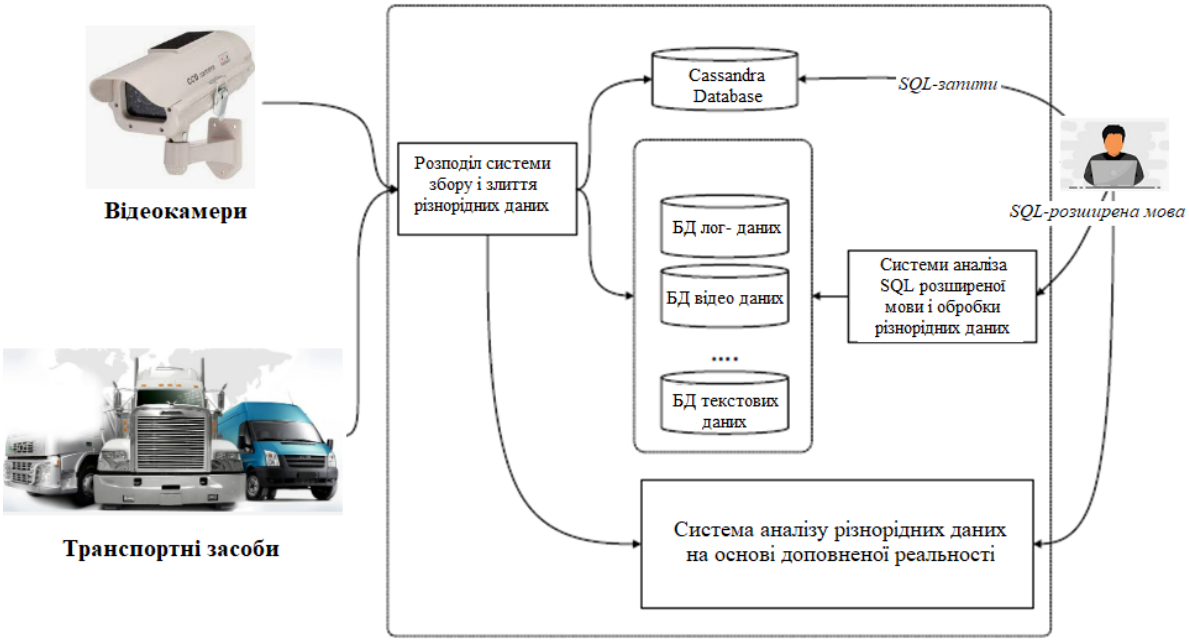


Рисунок 3.8 – Архітектура проактивної системи керування транспортною інфраструктурою на основі злиття та аналізу різноманітних даних

Припустимо, що у різних транспортних об'єктах є прилади для запису інформації про рух, але в елементах дорожньої інфраструктури -- відеокамери для відеофіксації поточної транспортної ситуації. Ці дані безперервно (через деякий інтервал часу) вирушають у систему управління транспортною інфраструктурою.

Відповідно до архітектури дані, отримані з різних джерел, будуть оброблені в модулі «Розподілена система збору та злиття різнорідних даних». Модуль «Розподілена система збору та злиття різнорідних даних» реалізована на основі розподіленого підходу з використанням технологій Apache Kafka та Spark Streaming. Технологія Apache Kafka дозволяє підвищити ефективність передачі

різноманітних даних з джерела до модулів обробки, а технологія Spark Streaming дозволяє підвищити швидкість обробки даних у пам'яті обчислювальної машини. У цьому модулі також застосовується метод збору та злиття різнорідних транспортних даних. Дані зберігаються у вихідному форматі в ХДОД з використанням досліджуваних методів. Після обробки даних зберігаються у базі даних Cassandra.

Для отримання даних з ХДОД була реалізована «Система аналізу SQL розширеної мови та обробки різнорідних даних». У цьому модулі використовується розроблений метод вибірки даних зі сховища різнорідних даних.

У «Системі аналізу різнорідних даних з урахуванням доповненої реальності» застосовується метод обробки різнорідних даних як реального часу. У цьому модулі реалізовано методи обробки відеопотоків із використанням бібліотеки OpenCV для розпізнавання транспортних об'єктів. Для покращення якості розпізнавання були використані моделі глибокого навчання, такі, як: нейронні мережі, мережна CNN, застосована готова система розпізнавання об'єкта в реальному часі YOLO (You Only Look Once), яка може обробити фрейми відео до 30 FPS (фрейм на секунду).

Програмну реалізацію системи управління транспортною інфраструктурою виконано з використанням мов програмування Java, Python в IDE Eclipse.

3.3 Обґрунтування ефективності досліджуваних методів у системі керування транспортною інфраструктурою

Обґрунтування ефективності метод збору та злиття різнорідних даних. Метод збору та злиття різнорідних даних був апробований при вирішенні завдання збору даних про транспортну ситуацію за даними з рухомих ТЗ (у вигляді логів) та відеопотоків [6]. Передбачається, що користувачеві необхідно отримувати дані про переміщення ТЗ в заданий проміжок часу $[t_0, t_k]$ в заданій географічній околиці всіх наявних джерел даних, тобто. з лог-файлів та з відеопотоків. При цьому з

відеопотоку мають бути вилучені кадри (у форматі JPG) відповідно до визначеного параметра дискретизації за часом. Час відгуку на запит має бути мінімальним.

Як базовий метод, з яким здійснювалося порівняння продуктивності пропонованих підходів, використовувалася реалізація підходу П1, що обробляє «сирі» дані в ХДОД без попередньої обробки. Час обробки даних пропонованими дисертаціями методами складалося з часу передобробки вихідних даних і часу виконання запиту. Для випробування були використані синтетичні дані, що генеруються авторською системою EVGEN з різним ступенем інтенсивності.

На рис. 3.9 показано архітектуру системи, що реалізує запропонований метод. Архітектура включає 5 шарів: рівень джерел даних (s-шар), рівень виробників (producer) даних (p-шар), рівень брокерів (brokers) (b-шар), рівень споживачів (c-шар) даних та рівень зберігання даних організації доступу до них стандартними засобами, наприклад SQL запитами (k-шар). Необхідна схема даних задається в єдиному просторі ключів (KEYSPACE) та у двох таблицях iotvideo та iotdata СУБД Cassandra у k-шарі. Опис джерел даних та налаштування параметрів збору даних здійснюється в JSON файлах налаштувань систем збору та завантажуються відповідними компонентами p-шару. Зв'язки між структурами задані у вигляді інсталяційних JSON файлів, але вже завантажуються компонентами з шару. Також компоненти з шару реалізують алгоритми перетворення відповідно до файлів налаштуваннями. Слід звернути увагу до роботи з відеопотоком. Для обробки та передачі відео використана бібліотека cvBlobs та Librdkafka. Також компоненти c шару записують отримані дані в базу даних k шару. Для виключення втрат у передачі даних їхнього p-шару в c-шар за високого ступеня інтенсивності, використовується b-шар брокерів Apache Kafka під керуванням Zookeeper.

Експерименти проводилися для різного ступеня інтенсивності даних, що надходять: для лог-файлів - 1 млн повідомлень в секунду і для одного джерела з відеопотоком. Середній час реалізації запропонованого методу склало відповідно 0,14 с та 0,75 с, тобто. у сумі 0,89 с. Також змінювалася інтенсивність запитів: 10, 30, 300 та 3000 запитів на секунду. Кожен експеримент проводився 10 разів.

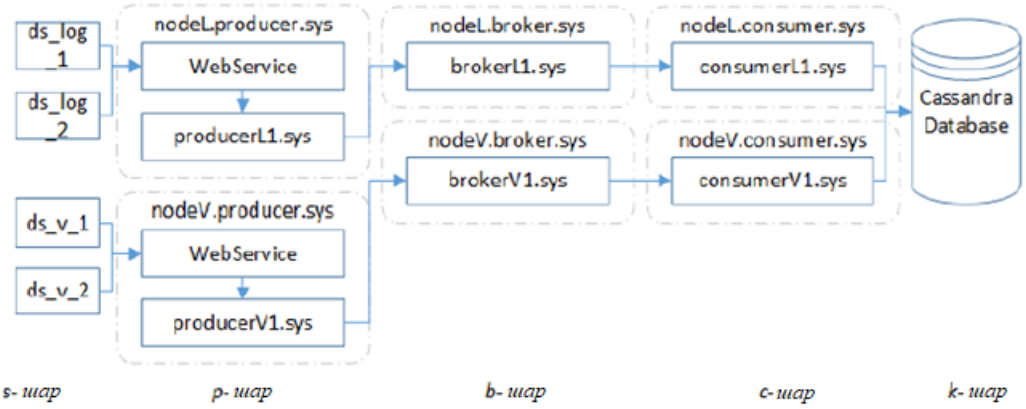


Рисунок 3.9 - Архітектура системи збору та злиття різнотипних даних із різних джерел

Так, для 10 запитів в секунду час запиту П1 склало 0,703±0,16 проти 1,087±0,32 с для запропонованого методу, що не є значною різницею. Однак вже для 30 запитів на секунду П1 показав час 1,925 ± 0,09 с, а пропонований метод - 1,368 ± 0,25 с, що в 14 рази менше П1. Для 300 запитів на секунду час виконання запропонованого методу становив 2,904±0,33 с, що у 1,7 разів менше ніж П1 (4,995±0,28 з). Для 3000 на секунду час виконання запропонованого методу становило 14,153±1,04 з, що у 3,24 разу менше, ніж П1 (45,899±2,56 з). Слід зазначити, що з підтвердження переваги запропонованого методу використовувалася непараметрична процедура тестування двох незалежних вибірок, запропонована Сигелом із рівнем значимості $\alpha = 0,01$.

Таким чином, запропоновано метод попередньої обробки даних, що одержуються з різнорідних джерел, таких як сенсорні дані та відеопотоки. Основною відмінністю методу від наявних є перетворення даних до потрібного формату у процесі передачі. Метод розрахований на операції перетворення вихідних даних або розділення відео на кадри з подальшою розміткою. Реалізація методу в програмній архітектурі «producer»-«broker»-«consumer» та експерименти вказують на скорочення часу запиту з видачею структурної інформації та кадрів відео в ~3.4 рази (для 3000 запитів на секунду) порівняно з методом, що реалізує обробку в процесі запиту.

Пропонований метод доцільно застосовувати в проактивних системах управління транспортною інфраструктурою, в яких критично час вибірки даних з консолідованого сховища даних, при цьому кількість запитів на секунду не менше десяти.

Обґрунтування ефективності методу збору та злиття різнорідних даних на основі підходу поділу даних на мікропотоки. Для апробації модифікованого методу також використані відеопотоки та зареєстровані файли про трафік із різних джерел даних. Відеопотоки, отримані з перехресть або дорожніх камер, та зареєстровані файли, зібрані з ТС. Система управління транспортною інфраструктурою повинна надавати інформацію про кількість транспортних засобів на перехресті за всіма наявними джерелами даних. Ця інформація є результатом запиту користувача. Ціль полягає в тому, щоб звести до мінімуму час виконання запитів користувача до сховища даних, що містить описані типи даних. Очікуваним результатом є скорочення часу виконання попередньої обробки даних.

У цьому дослідженні було використано розподілена архітектура, у якій компоненти Spark Streaming поєднуються із сервером Apache Kafka. Spark Streaming дозволяє виконувати попередню обробку даних у пам'яті. На рис.3.10 показано уявлення пропонованої розподіленої архітектури на вирішення, реалізує цей метод.

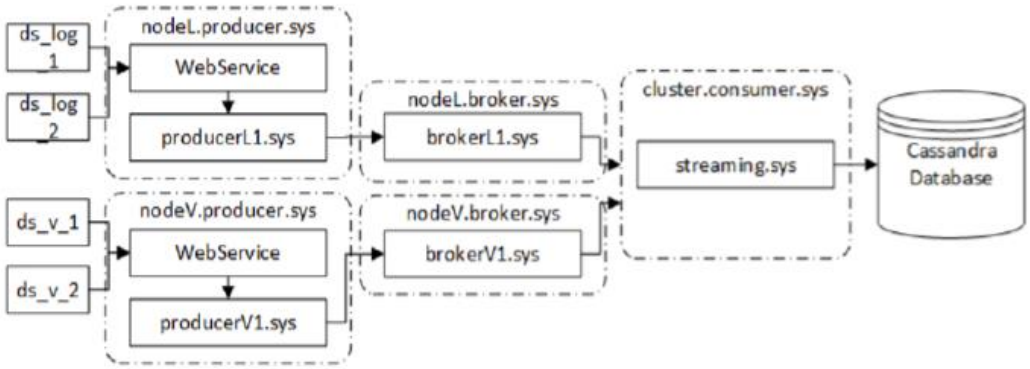


Рисунок 3.10 – Пропонована розподілена архітектура системи, що реалізує метод збору та злиття різнорідних даних

Пропоноване рішення засноване на розподіленій архітектурі, що міститься на п'яти рівнях: s-шар - шар джерел даних з різними джерелами даних, позначених як ds _*_* в архітектурі; p-шар - шар виробників з компонентами з ім'ям producer*.sys; b-шар - шар брокера (broker*.sys); c-шар - шар споживачів (cluster.consumer.sys); k-шар - шар сховищ даних з доступом до даних на базі SQL (база даних Cassandra).

Попередня певна схема визначається у загальному просторі ключів у двох таблицях MSDB Cassandra IoTCameraTraffic та IoTVehicleTraffic у k-шарі.

Сценарій створення простору ключів та таблиць у Cassandra MSDB представлено на рис.3.11.

```
// create key space
CREATE KEYSPACE Traffic_Space WITH
replication = {'class':'SimpleStrategy',
'replication_factor':3};
// create table for saving iot vehicle traffic
CREATE TABLE Traffic_Space.IoTVehicleTraffic
(IdVehicle text, TypeofVehicle text, timeStamp timestamp,
recordDate text, SpeedofVehicle float,
PRIMARY KEY (IdVehicle,recordDate,TypeofVehicle));
// create table for saving iot camera traffic
CREATE TABLE Traffic_Space.IoTCameraTraffic
(IdCamera text, timeStamp timestamp,
recordDate text, totalCount bigint,
PRIMARY KEY (IdCamera,recordDate));
```

Рисунок 3.11- Сценарій створення простору ключів та таблиць у Cassandra MSDB

Налаштування p-шару містить опис джерел даних та налаштування інструментів збору даних. Зазвичай ці параметри представлені у вигляді налаштувань, які завантажуються компонентами p-шару. Налаштування прив'язки різних схем даних знаходиться в файлах JSON для c-шару. C-шар містить алгоритми передачі даних у реальному часі відповідно до правил прив'язки та функціональності Spark Streaming. У разі високого навантаження для запобігання втраті даних при переході від p-шару до c-шару використовується b-шар брокерського рівня. В Apache Kafka b-шар керується сервером Zookeeper. У c-шарі

з використанням технології Spark Streaming поділяються потоки даних, отримані з b-шару, мікропотоки (мікро-пакети) даних. Потоки даних у форматі мікро-пакетів завантажуються в DStream для обробки пам'яті. Після цього оброблені дані зберігаються у базі даних Cassandra у k-шарі.

У цій версії реалізації аналізу відеопотоків використовується бібліотека обробки зображень OpenCV, обробка відеоданих виконується в p-шарі. Основною перевагою пропонованої архітектури є скорочення часу попередньої обробки даних. Слід звернути увагу, що кількість виробників має бути пропорційною до кількості джерел даних.

Для тестування ефективності роботи запропонованого рішення було проведено кілька експериментів тривалістю 30 хв 38 сек (365 експериментів, 1318747 записів). Кожен експеримент складався із трьох частин:

- частина A: де подієві файли та відеопотоки (тут і після відео-журналу) приходять з низькою інтенсивністю ~100 подій на секунду;

- частина B: де тільки подієві файли (тут і після «log») надходять із частотою ~1500 подій на секунду;

- частина C: де лог-файли та відеопотоки мають інтенсивність ~1000 подій за секунду.

На рис. 3.12 представлено інтенсивність даних під час експерименту. Під час експерименту використовувалось 9 типів режимів інтенсивності:

1. 69 пакетів (18,9%) з інтенсивністю в інтервалі [1252; 1408] подій/сек;
2. 46 пакетів (12,6%) з інтенсивністю в інтервалі [1095.5; 1252] подій/сек;
3. 70 пакетів (19,8%) з інтенсивністю в інтервалі [939; 1095.5] подій/сек;
4. 29 пакетів (7,95%) з інтенсивністю в інтервалі [782.5; 939] подій/сек;
5. 7 пакетів (1,92%) з інтенсивністю в інтервалі [626; 782.5] подій/сек;
6. 6 пакетів (1,64%) з інтенсивністю в інтервалі [469.5; 626] подій/сек;
7. 3 пакети (0,82%) з інтенсивністю в інтервалі [313; 469.5] подій/сек;
8. 4 пакети (1,1%) з інтенсивністю в інтервалі [156.5; 313] подій/сек;
9. 130 пакетів (35,621%) з інтенсивністю в інтервалі (0; 156.5) подій/сек.

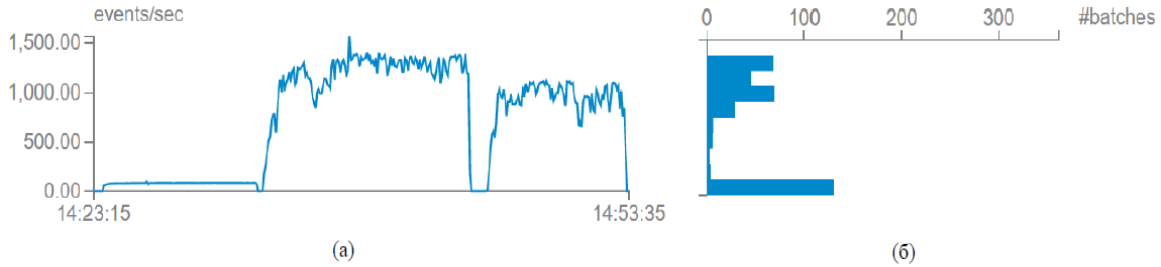


Рисунок 3.12 – (а) Візуалізація результатів під час експерименту (б) Гістограма показує розподіл пакетів залежно від інтенсивності

Результати отримані з використанням робочої станції з ОС Linux оперативною пам'яттю 8 ГБ. Застосовувалися виробники двох типів: виробники зареєстрованих джерел даних (виробник I типу) та виробники джерел даних відеопотоків (виробник II типу). Відеопотоки були дефрагментовані в p шарі, де виходять зображення та їх описи (метадані). Таблиця 3.1 містить середні результати для кількох повторень експериментів (від 18 до 27 циклів) відповідно до описаного профілю експериментів.

Кількість виробників змінювалося від 1 до 10, кількість брокерів варіювалася від 1 до 3. Виробники надсилають дані брокерам Kafka Cluster. Споживачі, засновані на Spark Streaming, одержують дані відповідно до заданих тем. На рис.3.13 показана візуалізація параметра часу затримки для одного експерименту (кількість виробників дорівнює 1, кількість брокерів дорівнює 1).

Таблиця 3.1 – Результати експериментів для певного профілю навантаження: середній час обробки та середня затримка для різної конфігурації

Частини експеремену	Число брокерів	Інтенсивність (подій в секунду)	Середній час обробки, с
Part A: video & log	1 broker	100	0,339 ± 0,151
Part B: log	1 broker	1500	3,095 ± 0,944
Part C: video & log	1 broker	1000	2,195 ± 0,905
Part A: video & log	3 brokers	100	0,191 ± 0,074
Part B: log	3 brokers	1500	1,519 ± 0,579
Part C: video & log	3 brokers	1000	1,066 ± 0,486

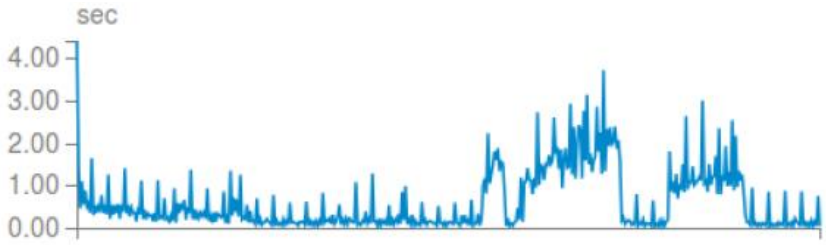


Рисунок 3.13 – Затримка під час експерименту (в секундах): вісь x – це вісь часу. Діаграма, отримана із потокової статистики Streaming Statistics

Перші групи експериментів були зроблені з кількома брокерами.

У частині A виробники I і II генерують дані з низькою інтенсивністю, тому середня затримка низька, а також 0,339 секунди. Затримка трохи відрізняється від часу обробки через специфіку обчислювального середовища. У частині B спостерігається пікова інтенсивність із середньою затримкою 3,095 секунди. Звернемо увагу, що у частині B оброблялися лише лог-файли. Середній час обробки частини C експерименту становить близько 2,195 секунд. Як і очікувалося, затримка збільшується, коли збільшується кількість підключених до джерел даних виробників. Проте піки затримки мають плавний вигляд навіть у разі максимальної інтенсивності.

Для експериментів з 3 брокерами середній час обробки зменшується: для частини A ~1,77 рази, для частини B ~2,6 рази, і для частини C зменшення склало ~2,05 рази. Основний висновок полягає в тому, що збільшення кількості брокерів для інтенсивності, що спостерігається, є розумним.

Обґрунтування ефективності методу вибірки різнорідних даних на основі досліджуваної SQL-подібної граматики. Для випробування були використані синтетичні дані, які генеруються авторською системою EVGEN. EVGEN розроблена на основі розподіленої системи обміну повідомленнями-підписками Kafka і складається з двох підсистем: підсистеми EventDataGenerators генерації подій, структура яких задається шаблонами JSON про транспортні засоби та/або транспортні перехрестя; підсистеми EventDataGenerationConsumers, що складається із споживачів (consumers), які отримують згенеровані події з топіків (topics) та обробляють їх.

На рис. 3.14 представлені шаблони, на основі яких генеруються дані в підсистемі EventDataGenerators.

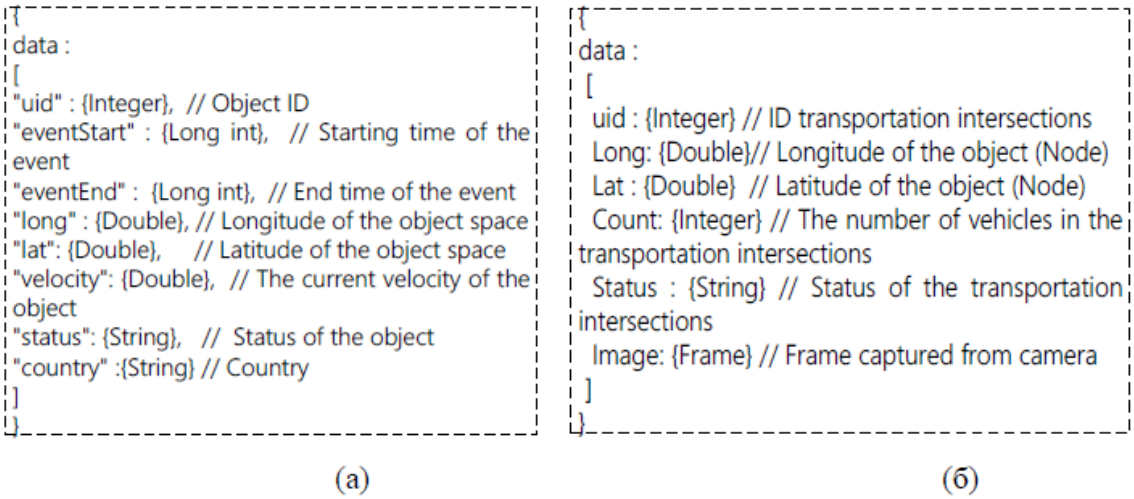


Рисунок 3.14 – Шаблони для генерації даних підсистемою EventDataGenerators: (а) JSON шаблон даних про транспортні засоби та (б) JSON шаблон даних про транспортні перехрестя

Дані, оброблені підсистемою Event Data Generation Consumers зберігаються у ХДОД для подальшої аналітики. Для тестування розробленої в SQL-подібної граматики та методу обробки різнорідних даних використано синтетичні дані з EVGEN. Як запит (запит 1) використовувався вислів «Обчислити кількість транспортних об'єктів, які надсилають дані на сервер або з'являються на ділянці дороги, камерою camera1, що охоплюється, в часовому інтервалі [t1;t2]. Запит з використанням запропонованої граматики представлений у наступному вигляді:

select count from Datalog, camera1 де змінено > 1516051435073 and modified <1516661699999;

де Datalog - шлях до JSON-файлу, що містить дані про ТС; camera1 – шлях до файлу, що містить зображення з камери №1.

Усі експерименти проводилися на ноутбуці під керуванням операційної системи Ubuntu x86_64 з процесором Intel(R) Core(TM) i5-2430M CPU з частотою 2.40GHz. Кожен експеримент повторювався 10 разів. У табл. 3.2 наведено результати експериментів для запиту 1.

Таблиця 3.2 – Результати виконання запиту 1, відповідно до розробленої граматики для різних джерел даних

№	№ Експери-мента	Об’єм лог даних (Мб)	Об’єм відео даних (Мб)	Час розбору запиту, с	Час виконання запиту, с	Середній час розбору запиту, с	Середній час виконання запиту, с
1	1	2	288,1	0,015	137,34	0,0135	148,785
	2			0,011	158,38		
	3			0,012	137,98		
	4			0,013	133,11		
	5			0,015	140,87		
	6			0,018	170,28		
	7			0,011	129,66		
	8			0,011	185,13		
	9			0,015	130,20		
	10			0,014	164,90		
2	1	2	230,4	0,012	99,00	0,0132	102,314
	2			0,015	101,22		
	3			0,014	102,56		
	4			0,008	110,62		
	5			0,014	91,6		
	6			0,016	98,44		
	7			0,017	106,56		
	8			0,012	99,92		
	9			0,011	108,77		
	10			0,013	104,45		
3	1	1	172,8	0,014	87,43	0,0139	87,836
	2			0,015	77,43		
	3			0,018	90,11		
	4			0,014	84,56		
	5			0,012	79,88		
	6			0,015	92,32		
	7			0,011	93,56		
	8			0,011	86,88		
	9			0,015	94,3		
	10			0,014	91,89		
4	1	1	115,2	0,011	58,54	0,0134	65,183
	2			0,011	67,67		
	3			0,019	68,88		
	4			0,011	72,28		
	5			0,014	58,4		
	6			0,015	66,54		
	7			0,016	66,77		
	8			0,012	59,41		
	9			0,014	65,47		
	10			0,011	67,87		

Продовження таблиці 3.2 – Результати запиту 1, відповідно до дослідженої граматики для різних джерел даних

№	№ Експери-мента	Об’єм лог даних (Мб)	Об’єм відео даних (Мб)	Час розбору запиту, с	Час виконання запиту, с	Середній час розбору запиту, с	Середній час виконання запиту, с
5	1	1		0,012	34,54	0,0133	36,772
	2			0,015	33,09		
	3			0,015	41,51		
	4			0,015	39,23		
	5			0,011	34,9		
	6			0,017	42,89		
	7			0,015	31,7		
	8			0,01	38,55		
	9			0,011	34,44		
	10			0,012	36,87		

Результат тестування запропонованого підходу до різних джерел даних подано у табл. 3.3. У табл.3.3 показано, що час парсингу даних різних експериментів відрізняється. Обробка даних значно швидше в порівнянні з обробкою відео, тому що структура зберігання даних досить проста для обробки. У той час як для обробки відео даних потрібно реалізувати алгоритми обробки зображень.

Таблиця 3.3 - Результати тестування виконання запиту 1 відповідно до розробленої граматики для різних джерел даних

	Об’єм лог-даних (Мб)	Об’єм відео даних (Мб)	Час розбору запиту, с	Час виконання запиту, с
1	2	288,1	0,0135 ± 0,0024	148,785 ± 19,442
2	2	230,4	0,0132 ± 0,0026	102,314 ± 5,588
3	1	172,8	0,0139 ± 0,0022	87,836 ± 5,770
4	1	115,2	0,0134 ± 0,0028	65,183 ± 4,777
5	1	57,6	0,0133 ± 0,0024	36,772 ± 3,686

У табл.3.4 наведено результати виконання запиту (запит 2), аналогічного попередньому за винятком того, що використовувалися джерела даних лог-файли:

select count from Datalog1, Datalog3, де modified > 1520540895935 and modified < 1546661699999;

Таблиця 3.4 - Результати виконання запиту 2 відповідно до розробленої граматики для різних джерел даних

	Об’єм лог-даних (Мб)	Час розбору запиту, с	Час виконання запиту, с
1	22,5	0,0119± 0,0039	0,9241 ± 0,303
2	45,1	0,0124± 0,009	1,2617 ± 0,639
3	67,6	0,0137± 0,0048	1,6622 ± 0,651
4	90	0,0133± 0,0046	2,3284 ± 0,440
5	108,8	0,0162± 0,0060	2,7487 ± 0,425

Третій експеримент включав запит (запит 3) тільки до відео:

select count from camera1, camera2 where modified > 1516051435073 and modified < 1546661699999;

Результати виконання запиту наведено в табл.3.5. Таким чином, експерименти проводилися для різних обсягів різноманітних даних (лог-файли та відеофайлів). Середній час аналізу SQL-запиту та реалізації одного запиту для різнорідних даних (2Мб лог-файлів та 288.1Мб відео файлів у вигляді зображень) склало відповідно 0,0135 с та 148,3 с, тобто. у сумі 148,31 с.

Таблиця 3.5 - Результати виконання запиту 3 відповідно до розробленої граматики для різних джерел даних

	Об’єм відео даних (Мб)	Час розбору запиту, с	Час виконання запиту, с
1	68,8	0,0138± 0,0055	31,529 ± 3,565
2	137,6	0,009± 0,0064	81,0256 ± 4,008
3	184,6	0,013± 0,0058	109,1793 ± 10,937
4	285,8	0,0129± 0,0067	133,0363 ± 12,255
5	442,7	0,0108± 0,0064	213,6505 ± 20,265

Також змінювалася конфігурація експериментів:

- 2Мб лог-файлів та 230,4 Мб відеофайлів у вигляді зображень;
- 1Мб лог-файлів та 172,8 Мб відеофайлів у вигляді зображень;
- 1Мб лог-файлів та 115,2 Мб відеофайлів у вигляді зображень;
- 1Мб лог-файлів та 57,6 Мб відеофайлів у вигляді зображень;

3.4 Обґрунтування ефективності методу у режимі реального часу із застосуванням елементів доповненої реальності

Для апробації методу аналізу різнорідних даних у режимі реального часу в системах управління транспортною інфраструктурою з елементами доповненої реальності було спроектовано архітектуру системи, представлену на рис.3.15.

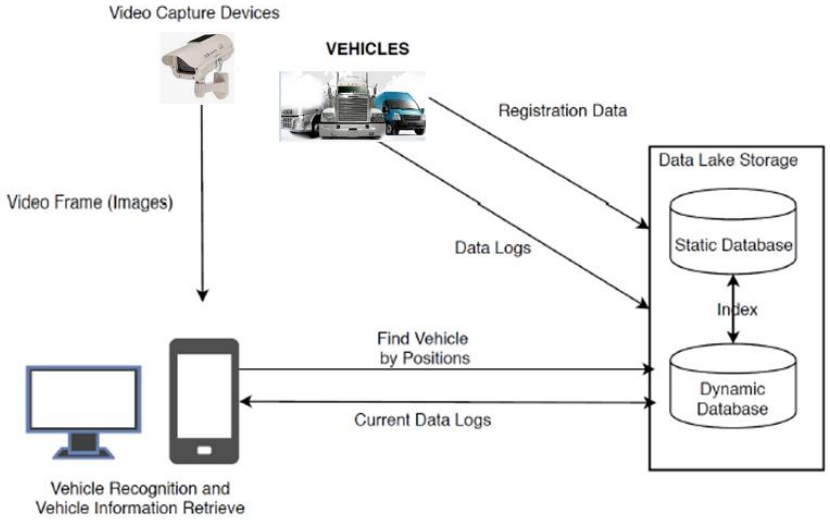


Рисунок 3.15 – Концепція системи, що реалізує запропонований метод аналізу різноманітних даних у режимі реального часу в системах керування транспортною інфраструктурою з елементами доповненої реальності

Пропонована система складається з чотирьох основних модулів:

1. Модуль збирання даних. Цей модуль фіксує зображення, які генеруються відеокамерами та відправляється в модуль синхронізації різнорідних даних.

2. Модуль синхронізації різноманітних даних. Модуль обробляє отримані зображення з модуля збору даних та виконує алгоритм локалізації об'єкта на сцені зображення. Функціонування модуля ґрунтується на використанні бібліотеки OpenCV. Коли об'єкт розпізнається в області зображення, модуль створює запит до модуля для роботи зі сховищем даних ХДОД для пошуку інформації про розпізнаний об'єкт.

3. Модуль для роботи зі сховищем даних ХДОД. Модуль реалізує методи пошуку інформації про розпізнаний транспортний засіб у сховищі даних (динамічна база даних), де зберігаються подійні дані про ТЗ. Сховище даних ХДОД включає і статичну базу даних і динамічну бази даних. Всі експерименти проводилися на ноутбуці під керуванням операційної системи Ubuntu x86_64 із процесором Intel (R) Core (TM) i5-2430M з частотою 2,40 ГГц.

Перший сценарій, у якому розпізнаються всі транспортні засоби на дорозі (зображення автомобілів відзначені зеленим прямокутником). Як тільки автомобіль перетинає червону лінію (обрану ділянку дороги), здійснюється розпізнавання. На рис. 3.16 показані результати розпізнавання ТЗ.

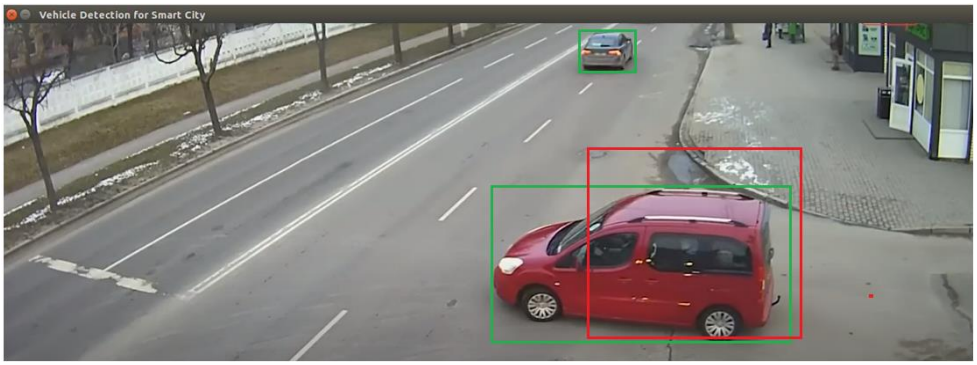


Рисунок 3.16 – Результати розпізнавання ТЗ, що перетинає задану область (червону область)

На рис. 3.17 показана інформація про розпізнаний об'єкт з ID 8389fa2f - 52a2-4d1b - 985c - 2b7de1f6fa0b, отримана з баз даних DS і DD.

У другому сценарій як ТЗ виступає ТЗ з великими габаритними розмірами (автобус), а інформації про це ТЗ немає у статичній базі даних ХДОД.

```
------------------------
Vehicle ID: 8389fa2f-52a2-4d1b-985c-2b7de1f6fa0b
Vehicle Speed: 57.91188681192046
Vehicle Location (latitude): 44.0000286909151
Vehicle Location (longitude): 46.00009722627896
------------------------
Current number of vehicle: 3
```

Рисунок 3.17 – Інформація про розпізнаний автомобіль з ID 8389fa2f-52a2-4d1b-985c-2b7de1f6fa0b, отримана з ХДОД

На рис. 3.18 показані результати розпізнавання ТС. На рис. 3.19 показано інформацію про те, що розпізнаний транспортний засіб не включено до статичної або динамічної бази даних ХДОД.

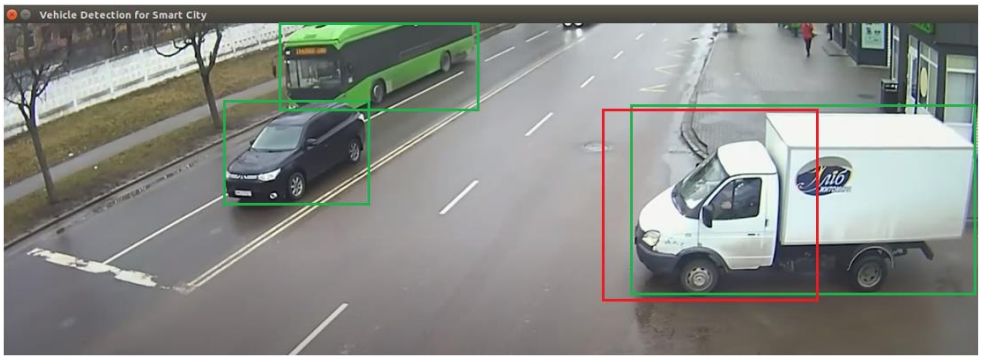


Рисунок 3.19 – Результат розпізнавання ТЗ великих розмірів

```
Current number of vehicle: 5
/home/kim/Desktop/logdata/22733645-ead0-414e-8370-e7293a19293c/1529273026332.txt
Eclapsed time:25 ms
No data available for the vehicle!
Current number of vehicle: 6
```

Рисунок 3.20 – Результат-повідомлення про те, що про МС немає інформації в ХДОД.

Зауважте, що велике ТЗ може приховувати інше ТЗ, тому це може бути ще однією проблемою, яку необхідно розглядати в майбутніх роботах.

На відміну від попереднього сценарію останній обробляє зображення невеликого ТЗ. На рис.3.21 показано результати розпізнавання невеликого ТЗ.

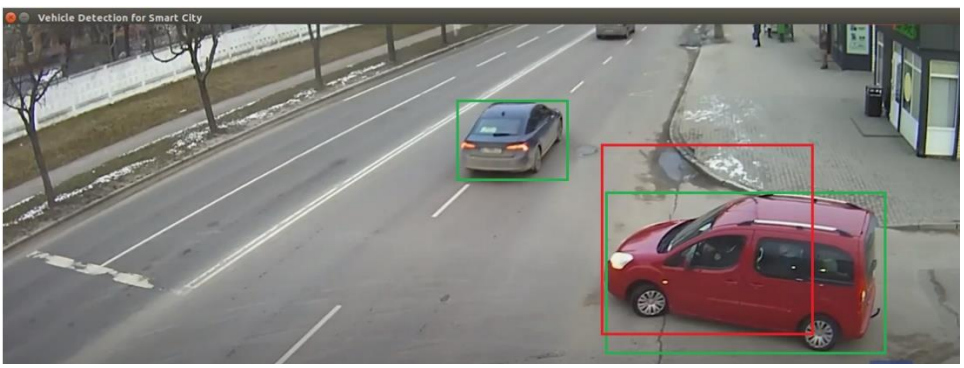


Рисунок 3.21 – Результат розпізнавання ТЗ, що перетинає задану область (червону область)

На рис.3.22 показано інформацію про розпізнане ТЗ з ідентифікатором a9ff8a33-cda4-487b-a5a5-1827115507b2, отриманим з баз даних ХДОД.

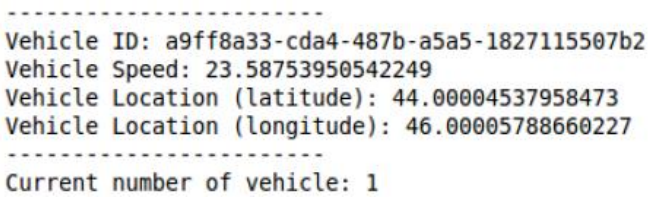


Рисунок 3.22 – Інформація про розпізнане ТЗ з ідентифікатором a9ff8a33-cda4-487b-a5a5-1827115507b2, отримана з баз даних ХДОД

Таким чином, у цьому розділі представлена апробація методу розпізнавання ТЗ, що рухаються, на основі бібліотеки OpenCV 3.0 для мови програмування Java.

Час обробки одного кадру відео (зображення) становить 10 – 35 мс, тому система підходить для роботи в реальному часі. Обсяг файлів журналів даних у динамічній базі даних становить ~2 Мб, час обробки подій даних становить 20 – 80 мс. Слід звернути увагу, що час пошуку даних транспортного засобу DD залежить від обсягу подійних даних. Технічно подієві дані можуть бути розділені на шматки даних відповідно до розташування транспортного засобу та періоду часу. В результаті тестування працездатність запропонованої системи показано можливість реалізації методу на практиці для великої кількості різних джерел. Також тестування ефективності запропонованого методу показало, що час виконання обробки різнорідних даних, таких як лог-даних та відеопотоки, досить невелике, що є основою для використання у проактивних системах управління транспортною інфраструктурою.

ВИСНОВКИ

Виконуючи поставлені задачі у магістерській роботі в першу чергу проведено аналіз систем підтримки прийняття рішень управління транспортною інфраструктурою, їх архітектури та особливості. Показано що критичним завданням у системах є завдання збору та попередньої обробки даних для подальшого прийняття рішень. Представлено класифікацію даних за різними критеріями.

В результаті системного аналізу процесу збору та злиття різнорідних даних у системах управління транспортною інфраструктурою, виділено основні операції збору та обробки різнорідних даних, для яких є проблема вдосконалення методів обробки даних. На основі аналізу сучасного стану досліджень у галузі збору та обробки різнорідних даних, досліджено концепцію сховища даних, що формується за принципом «озера даних» на основі λ-архітектури, в якій реалізуються як пакетна, так і потокова обробка даних.

Розглянуто модель та методи ефективного збору та попередньої обробки різнорідних даних відповідно до концепції «озеро даних» та наявності механізмів перетворення даних до потрібного формату в процесі їх передачі та поділу на мікропотоки даних, що дозволяє знизити час виконання запитів до різнорідних даних. Для реалізації методів використано граматику уніфікованих SQL-подібних запитів до різнорідних даних, що дозволяє формувати запити без урахування специфіки даних.

У практичній частині магістерської дипломної роботи виконано аналіз практичних випробувань та показано ефективність досліджуваних методів.

Таким чином на основі отриманих результатів можна зробити висновки про ефективність досліджуваних моделей та методів для збирання та попередньої обробки різнорідних даних у системах управління транспортною інфраструктурою.

ПЕРЕЛІК ПОСИЛАНЬ

1. Horn, P. Autonomic Computing: IBM's Perspective on the State of Information Technology/ P. Horn. – October 15, 2016.

2. Tennenhouse, D.L. "Proactive Computing"/ D.L. Tennenhouse// Communications of the ACM. – Vol. 43 . – No. 5. – May 2017. – PP. 43-50.

3. Буянов, Б.Б. Система поддержки приниятия управленческих решений с применением имитационного моделирования/ Б.Б. Буянов, Н.В. Лубков, Г.Л. Поляк // Проблемы управления. – № 6. – 2016. – С. 43-49.

4. Conner, S. Making Everyday Life Easier Using Dense Sensor Networks / S. Conner, L. Krishnamurthy, R. Want // Proceedings of Ubicomp 2001: 3rd International Conference on Ubiquitous Computing, Atlanta, GA, Springer, Lecture Notes in Computer Science 2201. – October 2017. – PP. 49-55.

5. Connecting the Physical World with Pervasive Networks / D. Estrin, D. Culler, K. Pister, G. Sukhatme // IEEE Pervasive Computing 1, No. 1, January-March 2019. – PP. 59 – 69.

6. Proactive behavior-based system for controlling safety risks in urban highway construction megaprojects / YongkuiLi et al. // Automation in Construction. – Vol. 95. – 2018. – PP. 118-128.

7. Wang, Z. Proactive traffic merging strategies for sensor-enabled cars / Z. Wang, L. Kulik, and K. Ramamohanarao.// In Proceedings of the Fourth ACM International Workshop on Vehicular Ad Hoc Networks (VANET ’07). - New York, NY, USA. - 2007.- PP. 39–48.

8. Industry Article: Proactive Event Processing in Action: A Case Study on the Proactive Management of Transport Processes / Zohar Feldman et al. // Proceedings of the Seventh ACM International Conference on Distributed Event-Based Systems, DEBS 2013, Arlington, Texas, USA . - 2018. - PP. 97 - 106.

9. Geospatial data generation and preprocessing tools for urban computing system development / Golubev et al. // Procedia Comput. Sci. 101 . – 2016. – PP. 217–226.

10. Hashem, I. A.T. The rise of ‘Big Data’ on Cloud Computing: Review and Open Research Issues / I. A.T. Hashem et al. // Information Systems. – 2015. – Vol. 47. – PP. 98–115.

11. Demchenko, Y. Defining Architecture Components of the Big Data Ecosystem / Y. Demchenko, C.D. Laat, P. Membrey. // In: Proc. of International Conference Collaboration Technologies and Systems (CTS’14). – 2014. – Vol.14. – PP. 104-112.

12. Venkatram, K. Review on Big Data Analytics – Concepts, Philosophy, Process and Applications [Electronic resource]/ K. Venkatram, M. A. Geetha // Cybernetics and Information Technologies. – 2017. – Vol. 17(2). – PP. 3–27.

13. Shafer, G. A Mathematical Theory of Evidence turns 40/ G. Shafer // International Journal of Approximate Reasoning. – Vol. 79. – December 2016. – PP. 7-25.

14. Evaluating the sustainability of Volgograd / Садовникова Н.П., Parygin D., Gidkova N., Gnedkova E., Sanzhapov B. // WIT Transactions on Ecology and the Environment. - 2014. - No. 179 (Vol. 1). - С. 279-290.

15. Want, R. Comparing autonomic and proactive computing / R. Want, T. Pering, D. Tennenhouse // IBM Systems Journal. – 2016. – Vol. 42. – Is. 1. – PP. 129 – 135.

16. Comparison of JSON and XML datainterchange formats: A case study / N. Nurseitov, M. Paulson, R. Reynolds, and C. Izurieta // In ISCA International Conference on Computer Applications in Industry and Engineering, Caine 2019, November 4-6, 2009, Hilton San Francisco Fisherman’s Wharf, San Francisco, California, Usa. – 2019. - PP. 157–162.

17. Modeling real-time human mobility based on mobile phone and transportation data fusion / Z. Huang et al. // Transportation Research Part C: Emerging Technologies.- Vol. 96.- November 2018.- PP. 251-269.

18. Proactive behavior-based system for controlling safety risks in urban highway construction megaprojects / YongkuiLi et al. // Automation in Construction. – Vol. 95. – 2018. – PP. 118-128.

19. Demchenko, Y. Defining Architecture Components of the Big Data Ecosystem/ Y. Demchenko, C.D. Laat, P. Membrey. // In: Proc. of International

Conference Collaboration Technologies and Systems (CTS'14). – 2014. – Vol.14. – PP. 104-112.

20. Acharya, T. Image Processing: Principles and Applications / T. Acharya, A. K. Ray // Hoboken, NJ: John Wiley & Sons. – 2015 . – PP. 79-104.

21. Hall, D. L. Handbook of Multisensor Data Fusion: Theory and Practice, Second Edition / D. L. Hall, J. Llinas // NY: CRC Press. Series: Electrical Engineering & Applied Signal Processing Series. – 2018. – 870p.

22. Wang, H. A Vehicle Detection Algorithm Based on Deep Belief Network/ H. Wang, Y. Cai, L. Chen // The Scientific World Journal. – 2019.

23. Benferhat, S. Reasoning with multiple-source information in a possibilistic logic framework / S. Benferhat, C. Sossai // Information Fusion .- 2016. – Vol. 7 (1). – PP. 80–96.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра Комп'ютерної інженерії

ПРЕЗЕНТАЦІЯ ДО МАГІСТЕРСЬКОЇ РОБОТИ

на тему:
«ДОСЛІДЖЕННЯ ШЛЯХІВ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ПРОЦЕСУ ЗБИРАННЯ ТА ОБРОБКИ ДАНИХ У СИСТЕМАХ УПРАВЛІННЯ ТРАНСПОРТНОЇ ІНФРАСТРУКТУРИ»

Студент: Бондаренко Є.С.
Керівник: Антоненко А.В.

Київ 2023р.

МЕТА ТА ЗАВДАННЯ

Слайд 2

Мета роботи - підвищення швидкості доступу до даних у процесі прийняття рішень управління транспортної інфраструктури.

Об'єкт дослідження – процес збору та обробки даних.

Предмет дослідження – методи збору та обробки різнорідних даних у системах керування транспортною інфраструктурою.

Для виконання поставленої мети, у магістерській роботі розроблено та виконано наступні завдання:

- аналіз процесів збору та злиття різнорідних даних у системах управління транспортною інфраструктурою.
- дослідження методів ефективного збору та обробки різнорідних даних у системах управління транспортною інфраструктурою;
- Рекомендації з практичної реалізації архітектури системи збору та обробки даних, обгрунтування ефективності досліджуваних підходів.

Системи підтримки прийняття рішень в управлінні транспортною інфраструктурою

Слайд 3

Рисунок 1 – Відношення між парадигмами обчислень

Принципи проактивних систем підтримки прийняття рішень:

1. Зв'язок із фізичним світом. *Значна частина існуючої нині обчислювальної інфраструктури пов'язує персональні комп'ютери з масивом серверів.*

2. Функціонування в масштабі реального часу та у замкнутому циклі.

3 Прогнозування. *Прогнозування – основа проактивних комп'ютерних систем. Щоб системи були справді проактивні, вони, у певному сенсі, мають передбачати майбутнє. Застосування низки перспективних методик дозволить системам швидко опрацьовувати ситуації реального світу, надаючи необхідний рівень взаємодії.*

Системи підтримки прийняття управлінських рішень, що використовуються в різних галузях дозволяють покращити якість управління процесами прийняття рішень. При реалізації управління транспортною інфраструктурою критичним завданням є завдання ефективної обробки різнорідних даних, що отримуються з різних джерел. Якість та своєчасність даних впливає на оперативність та результативність прийняття рішень. Тому для вирішення завдань управління транспортною інфраструктурою необхідно вирішити питання збору, зберігання та забезпечення ефективного доступу до різнорідних даних великого обсягу.

Аналіз класифікації даних

Слайд 4

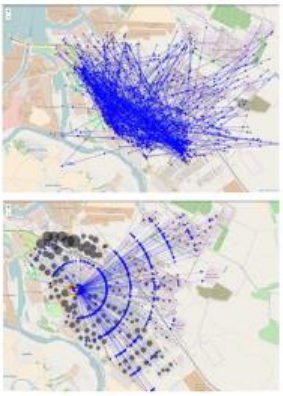


Рисунок 2 – Приклад візуалізації георозподілених даних. Вузли точки відправлення – призначення всередині міста, стрілки – напрямок необхідного руху

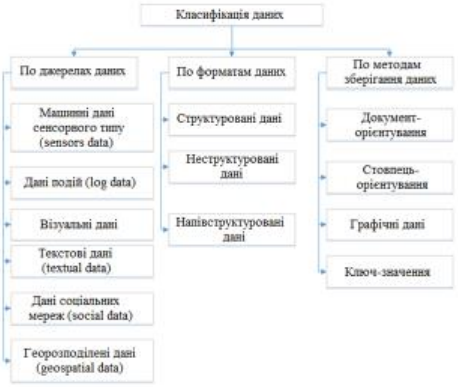


Рисунок 3 – Класифікація даних, що використовуються в обробці за проактивної підтримки прийняття рішень

Дослідження моделей зберігання різнорідних джерел даних у системах з пакетною і потоковою обробкою даних. Огляд технологічних платформ збору

Слайд 5

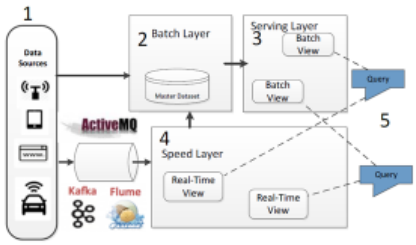


Рисунок 4 – λ-архітектура для реалізації системи обробки потоків даних у реальному часі

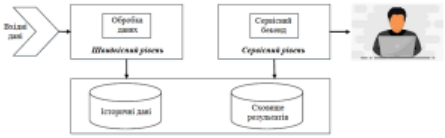


Рисунок 5 - Архітектура Карра

Таблиця 1 - Порівняння платформ обробки великих даних

Платформи	Розробник	Тип	Опис
Storm	Twitter	Потоковий	Рішення для аналізу великих даних в Twitter
S4	Yahoo	Потоковий	Платформа для обчислень з розподілених потоків даних
Hadoop	Apache	Пакетний	Перший відкритий фреймворк, який реалізує модель MapReduce
Spark	UC Berkeley AMPLab	Пакетний+ Потоковий	Аналітична платформа, яка підтримує в пам'яті різні набори даних, які володіють високою відмовостійкістю
Disco	Nokia	Пакетний	Фреймворк Nokia для розподілу моделі MapReduce
HPCC	LexisNexis	Пакетний	HPC кластер для великих даних
Apache Flink	Stratosphere	Пакетний+ Потоковий	Платформа розподілу обробки для обчислення з урахуванням стану по необмеженим та обмеженим потокам даних. Платформа розроблена для виконання розрахунків з високою швидкістю роботи і в любому маштабі.

Огляд моделі розподіленого зберігання даних, методів збирання та попередньої обробки

Слайд 6

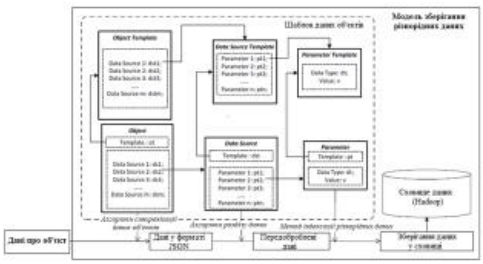


Рисунок 6 – Модель зберігання різнорідних даних

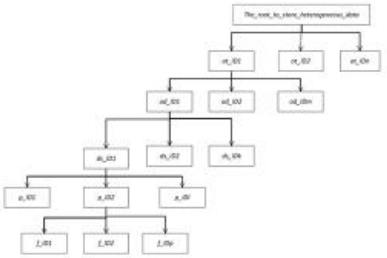


Рисунок 7 – Принцип індексування різнорідних даних у сховищі даних

Етапи досліджуваного методу:

1. Визначення необхідної схеми даних.
2. Опис джерел даних та налаштування параметрів збору даних.
3. Встановлення зв'язків між структурами.
4. Реалізація алгоритмів перетворення даних.
5. Запис до бази даних.

Рисунок 8– Кадр, отриманий із відеопотоку: стан транспортної інфраструктури

Метод реалізації запитів до різнорідних даних. Метод з елементами доповненої реальності

Слайд 7

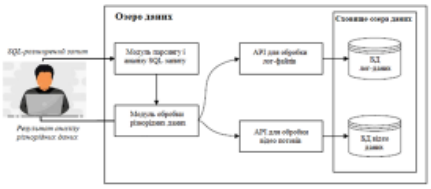


Рисунок 9- Архітектура системи обробки різнорідних даних із використанням уніфікованих запитів

Таблиця 2 – Приклад запитів до СДОД

Запит	Значення запиту
select count from DATALOG, Camera1 where MODIFIED > 1516051435073 and MODIFIED < 1516661699999;	Повернути кількість ТЗ із джерел DATALOG (лог-файл) та CAMERA1 (фрейми відеофайлів), які надіслали дані на сервер в інтервалі $t_{modified} > 1516051435073$ та $t_{modified} < 1516661699999$
select count from DATALOG, CAMERA2 where longitude > 44 and longitude < 45 and latitude > 45 and latitude < 46;	Повернути кількість ТЗ в певному розташуванні (longitude > 44 and longitude < 45 and latitude > 45 and latitude < 46) із джерел Datalog (лог-файл) та camera2 (фрейми відеофайлів).

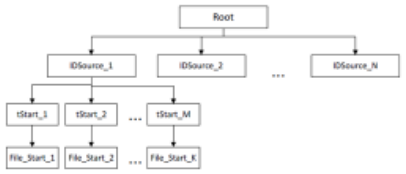


Рисунок 10 – Загальна структура зберігання даних

Кроки методу :

Етап 1. Спостерігач, який використовує AR-систему, починає захоплювати зображення дігляки дороги у певний час τ (час початку події).

Етап 2. Пакет даних DP про запуск події надсилається на сервер. Пакет містить дані про спостерігача (його місцезнаходження) та час початку події τ.

Етап 3. У момент часу надходження пакетів даних на сервер, формується та надсилається запит відеопотоку VS_C відповідно до місця розташування спостерігача та місця розташування камери.

Етап 4. Запускається процедура розпізнавання транспортного засобу за зображеннями, витягнутими з відеопотоків VS_i в інтервалі часу $[\tau-\varepsilon, \tau+\varepsilon]$.

Етап 5. Якщо транспортний засіб розпізнаний, виконуються відповідні дії.

Етап 6. Надіслати пакет даних DP_r кінцевому спостерігачеві.

Фреймворк генерації подій транспортної інфраструктури для оцінки ефективності методів

Слайд 8

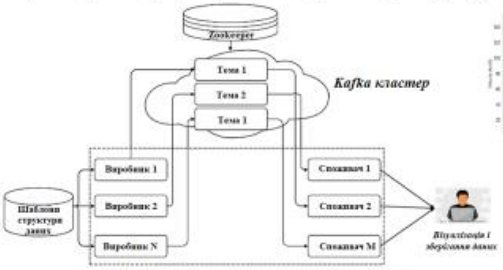


Рисунок 11 – Архітектура пропонованого фреймворку генерації подій у режимі реального часу

Рисунок 12- Структура лога-файлу згенерованих/отриманих подій

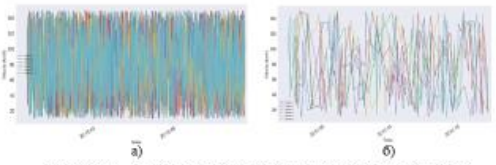


Рисунок 13 – Візуалізація даних про швидкість МС з інтенсивністю потоку подій а) l = 912 подій на секунду; б) l = 47 подій на секунду

Рисунок 14 - Візуалізація згенерованих геопросторових даних із використанням бібліотеки Python mplleaflet

Архітектура технологічного стека для реалізації досліджуваних методів

Слайд 9

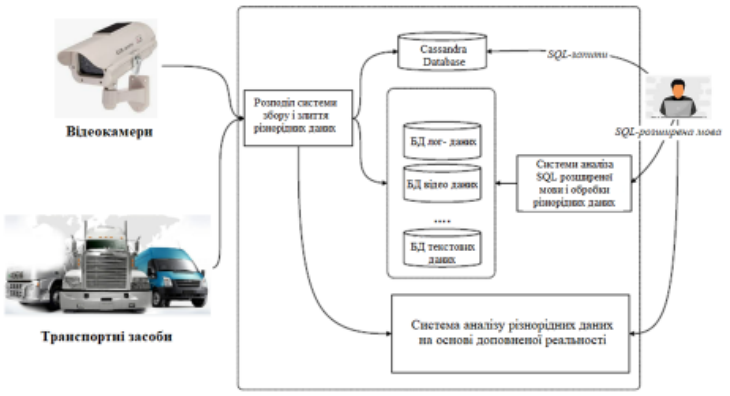


Рисунок 15 – Архітектура проактивної системи керування транспортною інфраструктурою на основі злиття та аналізу різноманітних даних

Обгрунтування ефективності досліджуваних методів у системі керування транспортною інфраструктурою

Слайд 10

Таблиця 3 – Результати експериментів для профілю навантаження: середній час обробки та середня затримка для різної конфігурації

Частини експерименту	Число брокерів	Інтенсивність (подій в секунду)	Середній час обробки, с
Part A: video & log	1 broker	100	0,339 ± 0,151
Part B: log	1 broker	1500	3,095 ± 0,944
Part C: video & log	1 broker	1000	2,195 ± 0,905
Part A: video & log	3 brokers	100	0,191 ± 0,074
Part B: log	3 brokers	1500	1,519 ± 0,579
Part C: video & log	3 brokers	1000	1,066 ± 0,486

Таблиця 4 - Результати тестування виконання запиту 1 відповідно до розробленої граматики для різних джерел даних

	Об'єм лог-даних (Мб)	Об'єм відео даних (Мб)	Час розбору запиту, с	Час виконання запиту, с
1	2	288,1	0,0135 ± 0,0024	148,785 ± 19,442
2	2	230,4	0,0132 ± 0,0026	102,314 ± 5,588
3	1	172,8	0,0139 ± 0,0022	87,836 ± 5,770
4	1	115,2	0,0134 ± 0,0028	65,183 ± 4,777
5	1	57,6	0,0133 ± 0,0024	36,772 ± 3,686

Таблиця 5 - Результати виконання запиту 2 відповідно до розробленої граматики для різних джерел даних

	Об'єм лог-даних (Мб)	Час розбору запиту, с	Час виконання запиту, с
1	22,5	0,0119± 0,0039	0,9241 ± 0,303
2	45,1	0,0124± 0,009	1,2617 ± 0,639
3	67,6	0,0137± 0,0048	1,6622 ± 0,651
4	90	0,0133± 0,0046	2,3284 ± 0,440
5	108,8	0,0162± 0,0060	2,7487 ± 0,425

Рисунок 16 – (а) Візуалізація результатів під час експерименту (б) Гістограма показує розподіл пакетів залежно від інтенсивності

Рисунок 17 – Затримка під час експерименту (в секундах): вісь x – це вісь часу. Діаграма, отримана із потокової статистики Streaming Statistics

Таблиця 6 - Результати виконання запиту 3 відповідно до розробленої граматики для різних джерел даних

	Об'єм відео даних (Мб)	Час розбору запиту, с	Час виконання запиту, с
1	68,8	0,0138± 0,0055	31,529 ± 3,565
2	137,6	0,009± 0,0064	81,0256 ± 4,008
3	184,6	0,013± 0,0058	109,1793 ± 10,937
4	285,8	0,0129± 0,0067	133,0363 ± 12,255
5	442,7	0,0108± 0,0064	213,6505 ± 20,265

Обгрунтування ефективності методу у режимі реального часу із застосуванням елементів доповненої реальності

Слайд 11

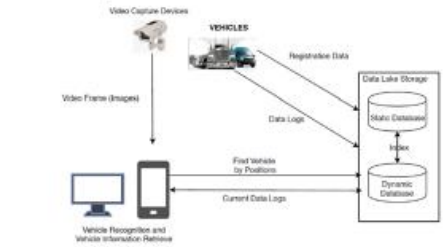


Рисунок 18– Концепція системи, що реалізує запропонований метод аналізу різноманітних даних у режимі реального часу в системах керування транспортною інфраструктурою з елементами доповненої реальності

Рисунок 19 – Результати розпізнавання ТЗ, що перетинає задану область (червону область)

Рисунок 20 – Результат розпізнавання ТЗ великих розмірів

```
Current number of vehicle: 5
/home/kjm/Desktop/logdata/22733645-ead0-414e-8370-
e7293a19293c/1529273626332.txt
Eclapsed time:25 ms
No data available for the vehicle!
Current number of vehicle: 6
```

Рисунок 21 – Результат розпізнавання ТЗ великих розмірів

```
-----------------------
Vehicle ID: a9ff8a33-cda4-487b-a5a5-18271155507b2
Vehicle Speed: 23.587539958542249
Vehicle Location (latitude): 44.00004537958473
Vehicle Location (longitude): 46.000057886660227
-----------------------
Current number of vehicle: 1
```

Рисунок 22 – Інформація про розпізнане ТЗ з ідентифікатором a9ff8a33-cda4-487b-a5a5-18271155507b2, отримана з баз даних СДОД

ВИСНОВКИ

Слайд 12

Виконуючи поставлені задачі у магістерській роботі в першу чергу проведено аналіз систем підтримки прийняття рішень управління транспортною інфраструктурою, їх архітектури та особливості. Показано що критичним завданням у системах є завдання збору та попередньої обробки даних для подальшого прийняття рішень. Представлено класифікацію даних за різними критеріями.

В результаті системного аналізу процесу збору та злиття різнорідних даних у системах управління транспортною інфраструктурою, виділено основні операції збору та обробки різнорідних даних, для яких є проблема вдосконалення методів обробки даних. На основі аналізу сучасного стану досліджень у галузі збору та обробки різнорідних даних, досліджено концепцію сховища даних, що формується за принципом «озера даних» на основі λ-архітектури, в якій реалізуються як пакетна, так і потокова обробка даних.

Розглянуто модель та методи ефективного збору та попередньої обробки різнорідних даних відповідно до концепції «озеро даних» та наявності механізмів перетворення даних до потрібного формату в процесі їх передачі та поділу на мікропотоки даних, що дозволяє знизити час виконання запитів до різнорідних даних. Для реалізації методів використано граматику уніфікованих SQL-подібних запитів до різнорідних даних, що дозволяє формувати запити без урахування специфіки даних.

У практичній частині магістерської дипломної роботи виконано аналіз практичних випробувань та показано ефективність досліджуваних методів.

Таким чином на основі отриманих результатів можна зробити висновки про ефективність досліджуваних моделей та методів для збирання та попередньої обробки різнорідних даних у системах управління транспортною інфраструктурою.