

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ОБРОБКИ ВЕЛИКИХ ДАНИХ, ОТРИМАНИХ З ТРАНСПОРТНИХ ЗАСОБІВ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS
FOR PROCESSING BIG DATA RECEIVED FROM VEHICLES

Виконав: студент 2 курсу, групи КНТ-112м

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ЧУЛЮК О.О.

(ПРІЗВИЩЕ та ініціали)

Керівник СУББОТІН С.О.

(ПРІЗВИЩЕ та ініціали)

Рецензент КОЦУР М.І.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2023 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ЧУЛЮКА Олександра Олександровича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів обробки великих даних, отриманих з транспортних засобів. Research and Software Implementation of Methods for Processing Big Data Received from Vehicles

керівник проєкту (роботи) д.т.н., професор, СУББОТІН Сергій Олександрович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 14 ” листопада 2023 року № 437

2. Строк подання студентом проєкту (роботи) 01 грудня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	СУББОТІН С.О., професор		
Нормоконтроль	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання “ 16 ” жовтня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій	2 тиждень	Розділ 2
	розробки		
4	Розробка структури програми.	3 тиждень	Розділ 3
5	Розробка програми.	4-7 тижні	Розділи 3,4
6	Тестування та експериментальне дослідження	8 тиждень	Розділ 4
	програмного забезпечення.		
7	Оформлення пояснювальної записки та документів	9-10 тижні	Додатки
	до неї.		
8	Нормоконтроль та рецензування.	11 тиждень	
9	Захист роботи.	12 тиждень	

Студент(ка)

(підпис) Олександр ЧУЛЮК
(Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Сергій СУББОТІН
(Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
97 с., 30 рис., 4 табл., 3 дод., 29 джерел.

JAVA, HADOOP, ECLIPSE, ВЕЛИКІ ДАНІ, ТРАНСПОРТНІ ЗАСОБИ,
МОВА ПРОГРАМУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Об'єкт дослідження – процес розробки програмного забезпечення для обробки великих даних, отриманих з транспортних засобів.

Предмет дослідження – програмні засоби для обробки великих даних, отриманих з транспортних засобів.

Метою роботи є дослідження та розробка програмного забезпечення для обробки великих даних, отриманих з транспортних засобів.

Матеріали, методи та технічні засоби: мова програмування Java, фреймворк Hadoop, середовище розробки Eclipse.

Результати. Розроблено програмне забезпечення для обробки великих даних, отриманих з транспортних засобів за допомогою мови програмування Java та фреймворку Hadoop.

Практична цінність роботи полягає у розробці програмного забезпечення для реалізації обробки великих даних, отриманих з транспортних засобів.

Висновки. Виконано проєктування програмного забезпечення для реалізації обробки великих даних, отриманих з транспортних засобів. Розроблено програмне забезпечення для реалізації обробки великих даних, отриманих з транспортних засобів. Здійснено тестування розробленого програмного забезпечення для реалізації обробки великих даних, отриманих з транспортних засобів.

Галузь використання – програмні засоби контролю водіїв.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 97 pages, 30 figures, 4 tables, 3 appendixes, 29 sources.

JAVA, HADOOP, ECLIPSE, BIG DATA, VEHICLES, CLUSTERING, PROGRAMMING LANGUAGE, SOFTWARE.

The object of research is the process of developing software for processing big data obtained from vehicles.

The subject of research is software for processing big data obtained from vehicles.

The method of operation is the research and development of software for processing big data obtained from vehicles.

Materials, methods and technical means: Java programming language, Hadoop framework, Eclipse development environment.

The results. Developed software to process big data obtained from vehicles using Java programming language and Hadoop framework.

The practical value of the work is in the development of software to implement the processing of big data obtained from vehicles.

Conclusions. The design of the software for the implementation of the processing of big data obtained from vehicles has been completed. Software has been developed to implement the processing of big data obtained from vehicles. Testing of the developed software was carried out to implement the processing of big data obtained from vehicles.

Field of use – driver monitoring software.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок.....	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Поняття Big data. Актуальність програмних застосунків для роботи з великими даними	11
1.2 Великі дані транспортних засобів.....	15
1.3 Аналіз програмного забезпечення для обробки великих даних, отриманих з транспортних засобів.....	17
1.4 Висновки за розділом 1	28
2 Матеріали і методи	30
2.1 Вибір мови програмування	30
2.2 Вибір фреймворку.....	34
2.3 Вибір середовища розробки.....	38
2.4 Налаштування та додаткові механізми фреймворку	41
2.5 Архітектура системи.....	46
2.6 Висновки за розділом 2	48
3 Опис програми.....	49
3.1 Завантаження даних з HDFS і видалення зайвої інформації.	49
3.2 Фільтрація даних.....	51
3.2.1 Вплив похибок у вихідних даних на роботу системи	51
3.2.2 Корекція вихідних даних	54
3.2.3 Фільтр Калмана.....	55
3.2.4 Застосування фільтра Калмана для обробки геоданих	57
3.2.5 Реалізація алгоритму	61
3.2.6 Виявлення точок зміни напрямку	62
3.3 Виявлення точок вимірювання прискорення	64
3.4 Розбиття траєкторії на ділянки і обчислення характеристик руху	65
3.5 Висновки за розділом 3	66

4	Експлуатація, тестування та експериментальне дослідження програми.....	68
4.1	Призначення й умови застосування програми	68
4.2	Проведення тестування. Аналіз отриманих результатів	68
4.3	Висновки за розділом 4	71
	Висновки.....	73
	Перелік джерел посилання.....	74
	Додаток А Технічне завдання.....	77
	Додаток Б Фрагмент тексту програми.....	81
	Додаток В Слайди презентації	90

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

AWS – Amazon Web Services;
GUI – Graphical user interface;
HDFS – Hadoop Distributed File System;
HDOP – Horizontal dilution of Precision;
IDE – Integrated Development Environment;
JDT – Java Development Tools;
RDP – алгоритм Рамера-Дугласа-Пекера;
UERE – User Equivalent Range Error;
UTM – Universal Transverse Mercator;
ДТП – дорожньо-транспортна пригода;
ООП – об’єктно-орієнтоване програмування;
ОС – операційна система;
ПЗ – програмне забезпечення;
ПК – персональний комп’ютер.

ВСТУП

На сьогоднішній день більшість сучасних транспортних засобів оснащено системами геопозиціонування, що дозволяють збирати інформацію про географічні координати, швидкості і напрямку руху контрольованих транспортних засобів [1]–[4].

Аналіз цієї інформації може бути корисним у багатьох областях. Так в автомобільних страхових компаніях ці дані можуть бути використані для визначення ризиків, пов'язаних з конкретним водієм. Ціноутворення може бути побудовано не тільки на основі характеристик транспортного засобу, але і на індивідуальних особливостях манери водіння кожного водія. Впровадження системи заохочень за акуратну їзду зможе підвищити безпеку на дорогах і розвантажити трафік [1]–[4]. Крім того, виявлення водійського почерку буде корисним для так званих інтелектуальних транспортних Систем, так як в цьому випадку стане можливим визначення недоступною з міркувань приватності інформації про особу водія і тип транспортного засобу тільки за даними про поїздку [1]–[4].

Однак, для аналізу даних, спочатку необхідно обробити вхідну інформацію, виділивши з неї набір властивостей, що характеризують кожную поїздку і стиль водіння. Процес отримання характеристик, що описують об'єкт, називається виділенням ознак (Feature Extraction). Виділення ознак призначене для зменшення кількості надлишкової інформації та представлення всього набору даних за допомогою менших ресурсів. Для будь-якої проблеми аналізу даних, якщо виділені ознаки дозволяють точно описати поведінку об'єкта, то проблема вважається практично вирішеною [1]–[4].

У разі обробки геоданих, отриманих з транспортних засобів, ознаки можна умовно розділити на дві категорії: характеристики поїздки в цілому і характеристики водіння. Поїздку в цілому можуть характеризувати кількість і частота зупинок, тривалість поїздки, пройденої відстань, час рівномірного

руху. До характеристик руху можна віднести середні величини прискорення, гальмування, швидкості, доцентрового прискорення в поворотах [1]–[4].

У даній роботі описана реалізація системи обробки великих обсягів геоданих, що надходять з транспортних засобів, що дозволяє виділити характеристики поїздок і водіння. Як приклад використання отриманих при роботі системи даних розглядається кластеризація водіїв на групи за манерою їх водіння [1]–[4].

У першому розділі роботи наведено короткий огляд існуючих підходів та інструментів, що застосовуються для обробки великих обсягів даних. Другий розділ містить опис обраних матеріалів та методів. Третій розділ містить докладний опис реалізованої системи, четвертий розділ – аналіз отриманих результатів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття Big data. Актуальність програмних застосунків для роботи з великими даними

У сучасному цифровому ландшафті термін Big data – великі дані, перетворився на ключове поняття, що відображає експоненціальне зростання даних, що генеруються щодня. Важливість великих даних полягає не тільки в їх величезному обсязі, але і в їх складності, різноманітності та потенційній інформації, яку вони містять [5]–[7]. Метою цієї дискусії є забезпечення всебічного розуміння великих даних, роз'яснення їх характеристик та висвітлення вирішальної ролі розробки нового програмного забезпечення для ефективного використання його можливостей [5]–[7].

За своєю суттю, великі дані відносяться до наборів даних, які є надзвичайно великими та складними, їх важко керувати, обробляти та аналізувати за допомогою традиційних інструментів обробки даних. Значення великих даних виходить за рамки розміру; воно охоплює три аспекти — обсяг, швидкість і різноманітність [5]–[7].

Обсяг. Великі дані характеризуються величезним обсягом даних. Він може варіюватися від терабайт до петабайт і більше, охоплюючи масштаби, з якими традиційним базам даних важко впоратися [5]–[7].

Швидкість. Швидкість, з якою дані генеруються, обробляються та аналізуються, визначає аспект швидкості великих даних. З появою джерел даних у режимі реального часу, таких як соціальні медіа та датчики, швидкість передачі даних стала критичним фактором [5]–[7].

Різноманітність. Великі дані надходять у різних форматах, включаючи структуровані дані, знайдені в реляційних базах даних, напівструктуровані дані, такі як XML або JSON, та неструктуровані дані, такі як текст, зображення та відео. Ця різноманітність представляє проблему, оскільки традиційні бази даних, як правило, структуровані та важко справляються з цими різноманітними типами [5]–[7].

Хоча ці три аспекти формують основні характеристики великих даних, концепція розширилася, включивши Додаткові аспекти, такі як достовірність (стосується якості та надійності даних), мінливість (стосується невідповідності потоку даних) та цінність (відображає кінцеву мету отримання значущої інформації) [5]–[7].

Важливість розробки програмного забезпечення (ПЗ) для роботи з великими даними.

Оскільки обсяг і складність даних продовжують зростати, розробка нового програмного забезпечення стає нагальною необхідністю для ефективного управління, обробки та отримання цінності з великих даних. Наступні моменти пояснюють важливість розробки спеціалізованого програмного забезпечення для роботи з великими даними [5]–[7].

Обробка великих обсягів. Традиційні інструменти обробки даних і реляційні бази даних важко справляються з величезними обсягами даних, що генеруються в сценаріях великих даних.

Спеціалізовані платформи обробки великих даних, такі як Apache Hadoop та Apache Spark, призначені для розподілу та обробки великих наборів даних між кластерами комп'ютерів. Ці платформи забезпечують паралельну обробку, забезпечуючи масштабованість для обробки зростаючих обсягів даних [5]–[7].

Обробка в режимі реального часу. Багато додатків вимагають обробки в режимі реального часу для отримання негайної інформації та сприяння своєчасному прийняттю рішень.

Платформи потокової обробки, такі як Apache Flink та Apache Kafka, дозволяють аналізувати дані під час їх створення. Ці платформи мають вирішальне значення для додатків, які потребують обробки з низькою затримкою, таких як фінансові операції та дані датчиків Internet of Things (IoT) [5]–[7].

Різноманітні типи даних. Традиційні реляційні бази даних призначені для роботи зі структурованими даними і можуть мати труднощі з різноманітністю типів даних, присутніх у великих даних [5]–[7].

Бази даних NoSQL, включаючи MongoDB та Cassandra, призначені для обробки різних типів даних. Вони можуть ефективно керувати неструктурованими та напівструктурованими даними, забезпечуючи гнучкість для додатків, що працюють з різними форматами даних.

Інтеграція даних. Великі дані часто надходять з різних джерел, і інтеграція цих даних для всебічного аналізу може бути складною.

Інструменти ETL (витяг, перетворення, завантаження) та платформи інтеграції даних допомагають збирати, перетворювати та об'єднувати дані з різних джерел в єдиний формат. Ці інструменти відіграють вирішальну роль у створенні цілісного набору даних для змістовного аналізу [5]–[7].

Масштабованість. Традиційним системам може бути важко масштабуватися горизонтально, щоб відповідати зростаючим наборам даних та збільшеним вимогам до обробки.

Масштабовані та розподілені обчислювальні платформи, такі як Kubernetes та Docker, полегшують розгортання програм у кластерах та управління ними. Це забезпечує масштабованість за рахунок динамічного розподілу ресурсів залежно від попиту [5]–[7].

Машинне навчання та аналітика. Аналіз великих наборів даних вручну непрактичний, а традиційні аналітичні інструменти можуть не масштабуватися для обробки аналізу великих даних.

Платформи машинного навчання, такі як TensorFlow та scikit-learn, інтегровані з платформами великих даних, забезпечують автоматизований та масштабований аналіз даних. Ці платформи дають організаціям можливість виявляти закономірності та ідеї у великих наборах даних, полегшуючи прийняття обґрунтованих рішень [5]–[7].

Безпека даних та конфіденційність. З великими наборами даних зростає занепокоєння щодо безпеки та конфіденційності даних.

Розробка програмного забезпечення з надійними функціями безпеки, механізмами шифрування та дотриманням правил захисту даних має вирішальне значення для захисту великих даних. Це включає контроль доступу, шифрування конфіденційних даних та дотримання нормативних рамок [5]–[7].

Економічна ефективність. Управління та обробка великих обсягів даних можуть спричинити значні витрати на інфраструктуру.

Служби хмарних обчислень, такі як Amazon Web Services (AWS) та Microsoft Azure, надають економічно ефективні рішення для зберігання, обробки та аналізу великих обсягів даних. Хмарні платформи пропонують масштабовані ресурси за моделлю – мінімізуючи потребу у великій локальній інфраструктурі [5]–[7].

Системи підтримки прийняття рішень. Прийняття обґрунтованих рішень на основі великих даних вимагає складних систем підтримки прийняття рішень [5]–[7].

Розробка програмного забезпечення, яке інтегрується з інструментами бізнес-аналітики та надає інтуїтивно зрозумілі інформаційні панелі, дозволяє користувачам приймати рішення на основі даних. Ці системи полегшують візуалізацію складних наборів даних, роблячи інформацію доступною для широкої аудиторії.

В решті решт, концепція великих даних являє собою зміну парадигми в тому, як ми збираємо, обробляємо та витягуємо інформацію з даних. Три критерії-обсяг, швидкість і різноманітність – підкреслюють проблеми, пов'язані з самим масштабом і складністю даних. Розробка спеціалізованого програмного забезпечення, адаптованого для вирішення цих завдань, необхідна для розкриття всього потенціалу великих даних [5]–[7].

Розробка програмного забезпечення для обробки великих обсягів даних – від обробки в режимі реального часу до інтеграції різних типів даних і забезпечення їх безпеки-відіграє важливу роль в навігації по складнощях інформаційного ландшафту. Оскільки організації продовжують приймати

рішення на основі даних, важливість інноваційних програмних рішень для роботи з великими даними стає все більш очевидною. Ці рішення не тільки сприяють ефективному управлінню даними, але й прокладають шлях до досягнень у різних галузях та галузях, стимулюючи інновації та забезпечуючи конкурентні переваги в цифрову епоху [5]–[7].

1.2 Великі дані транспортних засобів

На більшості транспортних засобів системи геопозиціонування дозволяють збирати дані з частотою 10 Гц. Таким чином, тридцятихвилинна поїздка описується 18 000 записами. З огляду на те, що найчастіше в цих записах міститься не тільки стандартний набір геоданих, таких як координати, швидкість і напрямок, але також і велика кількість додаткової інформації зі всіляких сенсорів і автомобільних систем, розмір файлу, що описує тижневу активність тисячі машин може становити більше 100ГБ [8]–[10]. Очевидно, що обробка таких обсягів даних стандартними програмними і апаратними засобами представляється вкрай складним завданням. У таких випадках застосовуються підходи, інструменти та методи великих даних.

Великі дані – серія підходів, інструментів і методів обробки структурованих і неструктурованих даних величезних обсягів і значного різноманіття для отримання сприйманих людиною результатів, ефективних в умовах безперервного приросту, розподілу по численних вузлах обчислювальної мережі, що сформувалися в кінці 2000-х років, альтернативних традиційним системам управління базами даних і рішенням класу Business Intelligence [8]–[10].

Зараз більшість експертів сходяться на думці, що прискорення зростання обсягу даних є об'єктивною реальністю. Джерелами, здатними генерувати гігантські обсяги інформації можуть бути соціальні мережі, мобільні пристрої, дані з вимірювальних пристроїв, бізнес-інформація. Крім того, значна частина інформації створюється не людьми, а всілякими

цифровими пристроями, взаємодіючими як один з одним, так і з іншими мережами даних. При існуючих на сьогодні темпах зростання, кількість даних у світі, за прогнозами дослідників, буде, як мінімум щорічно подвоюватися. Дослідження IDC Digital Universe, опубліковане в 2012 році, пророкує, що в найближчі 8 років кількість даних в світі досягне близько 40 зеттабайт, що еквівалентно приблизно 5200 Гб інформації на кожного жителя планети (рис. 1.1) [8]–[10].

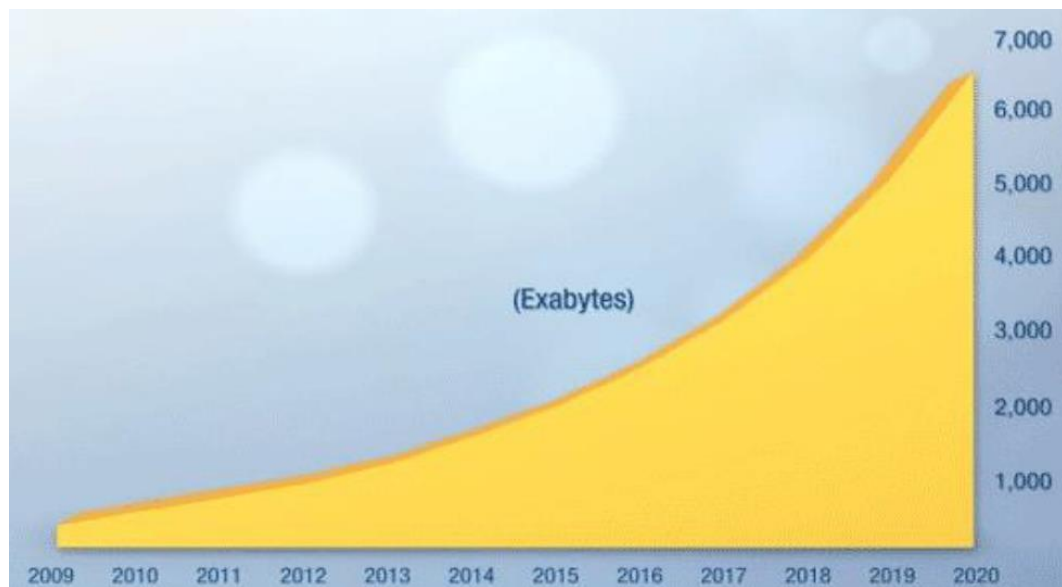


Рисунок 1.1 – Зростання зібраної цифрової інформації в США

Парадигма Big Data визначає наступні типи завдань:

- зберігання та управління обсягом даних, з яким звичайні реляційні бази даних не дозволяють ефективно працювати;
- обробка та організація неструктурованої інформації, що складається з текстів, зображень, відео та інших типів даних;
- аналітика великих даних. Включає в себе методи класу Data Mining, машинного навчання, розпізнання образів, предиктивну аналітику, а також штучні нейронні мережі;
- візуалізація даних: представлення інформації у вигляді малюнків, діаграм, з використанням інтерактивних можливостей і анімації як для

отримання результатів, так і для використання в якості вихідних даних для подальшого аналізу [8]–[10].

Робота з великими даними повинна підкорятися наступним принципам:

- горизонтальна масштабованість. Будь-яка система, яка передбачає обробку великих даних, повинна бути легко розширюваною. Для збереження заданої продуктивності і швидкодії при зростанні обсягу даних, кількість витрат має бути пропорційно цьому зростанню [8]–[10];

- відмовостійкість. Принцип горизонтальної масштабованості має на увазі, що машин в сервері може бути багато. Це означає, що частина цих машин буде гарантовано виходити з ладу. Методи роботи з великими даними повинні враховувати можливість таких збоїв і переживати їх без будь-яких значущих наслідків;

- локальність даних. У великих розподілених системах дані розподіляються по великій кількості машин. Якщо дані фізично знаходяться на одному сервері, а обробляються на іншому - витрати на передачу даних можуть перевищити витрати на саму обробку. Тому одним з найважливіших принципів проектування рішень великих даних є принцип локальності даних – обробка даних проводиться на тій машині, де ці дані зберігаються [8]–[10].

1.3 Аналіз програмного забезпечення для обробки великих даних, отриманих з транспортних засобів

Проаналізуємо ПЗ для обробки великих даних, отриманих з транспортних засобів.

Застосунок Robi VTS – це автомобільний та транспортний застосунок, розроблений Bondstein [11], [12]. Наразі доступна виключно мобільна версія, а щоб завантажити Robi VTS на комп'ютер, знадобиться завантажити та розгорнути емулятор операційної системи (ОС) Android. Емулятор Android являє собою програму, яка імітує систему мобільного телефону Android на комп'ютері. LDPlayer – це емулятор Android, призначений для комп'ютерів з

ОС Windows. Він створений на базі версії Android 9.0 і має такі потужні функції як багатовіконність, підтримка макросів і запис операцій, які недоступні на мобільних телефонах. LDPlayer розроблений для найвимогливіших любителів ігор і дозволяє отримати максимум від мобільних ігор на персональний комп'ютер (ПК) [11], [12].

Robi VTS – працює на Bondstein Technologies Limited. Власне рішення Robi для відстеження транспортних засобів. Robi Tracker використовує технологію GPS для отримання детальних даних про місцезнаходження автомобіля, які потім наносяться на карту та вказуються на Картах Google. Використовуючи технологію GPRS на основі SIM-карти, ця інформація безперешкодно передається на екран вашого персонального комп'ютера або мобільного телефону. Потім транспортний засіб можна відстежувати через вебпортал Robi Tracker, мобільний застосунок або SMS [11], [12].

Robi Tracker призначений не тільки для окремих транспортних засобів, але також може використовуватися для управління та моніторингу корпоративних автопарків.

Основні характеристики застосунку включають:

- відстеження в реальному часі – зручний вебінтерфейс для моніторингу автомобіля. Користувач може просто увійти зі свого комп'ютера або мобільного телефону і почати відстежувати;
- знаходження свого автомобіля – використання користувацького інтерфейсу або SMS-повідомлення, щоб визначити місце розташування автомобіля;
- порушення швидкості – користувач може отримувати сповіщення на свій телефон, коли його автомобіль перевищує встановлену межу швидкості;
- безпечний режим / дистанційне вимикання / увімкнення двигуна – користувач у надзвичайних ситуаціях може дистанційно вимкнути двигун віддалено через вебпортал/додаток/SMS [11], [12];

- режим «не турбувати» – користувач може використовувати функцію відключення двигуна, щоб створити віртуального сторожа, який відключить запалювання і ініціює SMS-повідомлення про будь-яку незвичайну активність;
- гео-фехтування – користувач може створювати будь-яку кількість настроюваних кордонів і отримувати повідомлення щоразу, коли його транспортний засіб залишає цю віртуальну геозону;
- різноманітні корисні звіти – завантажувані звіти про швидкість, місцезнаходження, маршрути та пройденої відстань вашого автомобіля за останні 3 місяці забезпечують повний контроль;
- спеціальний центр підтримки – підтримка call-центру доступна для обробки всіх запитів та координації будь-якої проблеми з технічною командою (рис. 1.2).

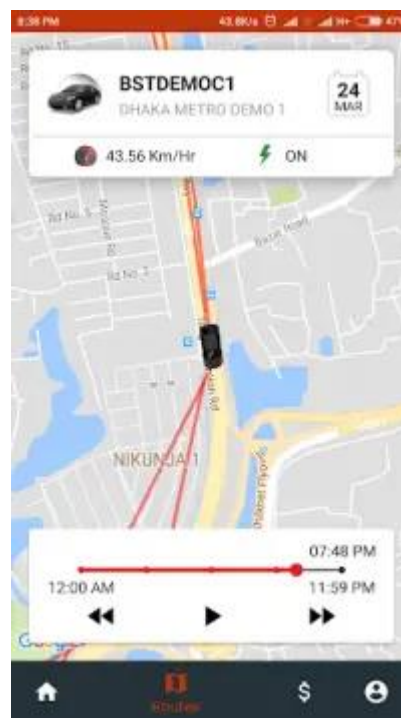


Рисунок 1.2 – Приклад інтерфейсу користувача у застосунку Robi VTS [12]

Мобільний застосунок для страхування авто КАСКО4U. Страхова компанія АСКА і швейцарський IT-проект kasko2go, Розробник мобільного

застосунку, об'єднали всі переваги страхової послуги повного КАСКО і мобільних інновацій, завдяки чому і з'явилася програма КАСКО4U [13], [14].

Її основна перевага – персоналізований страховий захист, або принцип – плати, коли їдеш. Через мобільний застосунок автовласник може активувати умови страхового захисту на свій розсуд – лише тоді, коли йому це потрібно. Наприклад, можна захистити авто від викрадення тільки на період відпустки або застрахуватися від дорожньо-транспортної пригоди (ДТП) виключно під час перебування за кермом, з щохвилинною тарифікацією [13], [14].

Крім того, інтелектуальна система в мобільному застосунку аналізує стиль водіння і визначає персональну знижку для кожного водія. Оформлення поліса КАСКО, зміна умов страхування, звернення за виплатою – всі взаємодії клієнта зі страховою компанією АСКА відбувається в онлайн режимі, через мобільний застосунок. А для тих, хто хотів би протестувати послугу перед покупкою, передбачена спеціальна можливість – безкоштовний режим: тест-драйв [13], [14].

Для повноцінного користування послугою необхідно сплатити мінімальний стартовий пакет: застрахувати авто від збитку під час стоянки [13], [14].

Мобільний страховий сервіс КАСКО4U (рис. 1.3) доступний для користувачів смартфонів на базі ОС Android, але розробник мобільного застосунку гарантує, що найближчим часом протестувати розумне КАСКО зможуть і власники iPhone [13], [14].

Навігаційний застосунок Waze, змінив спосіб навігації людей та їх поїздок на роботу – свідчення сили технологій та великих даних у формуванні повсякденного життя [15], [16].

Waze – це навігаційний застосунок, керований спільнотою, яка надає створені користувачем дані про дорожній рух у режимі реального часу для покращення планування маршруту та пропонує рекомендації щодо динамічної навігації [15], [16].

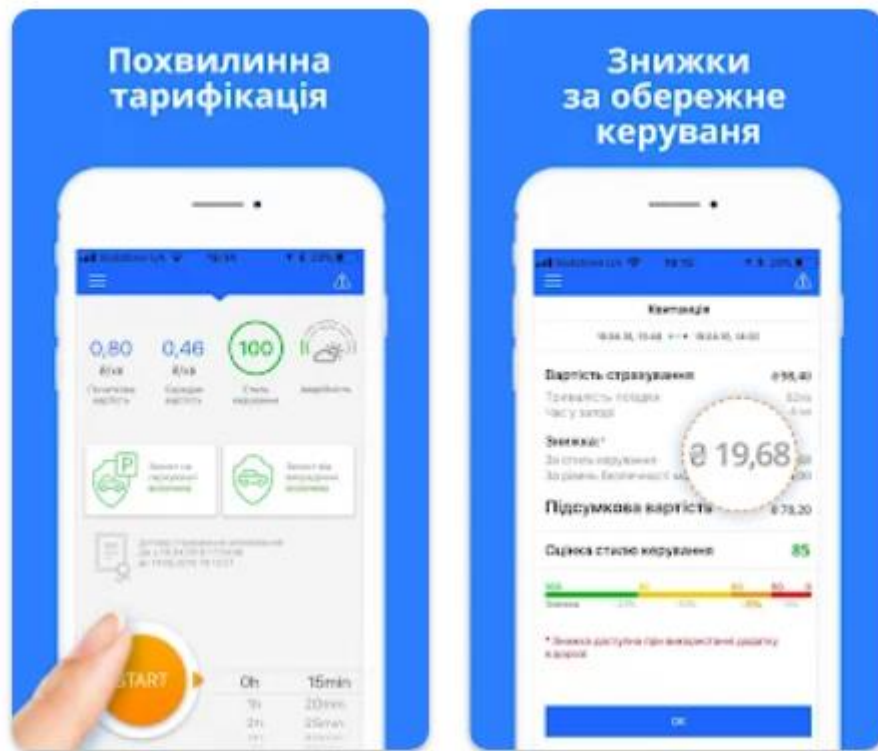


Рисунок 1.3 – Приклад інтерфейсу користувача у застосунку KASKO4U [14]

Запущений у 2006 році, Waze перетворився на одне з найпопулярніших навігаційних застосунків у всьому світі, доступний як на платформах Android, так і на iOS [15], [16].

Основні характеристики:

- оновлення трафіку в режимі реального часу: Waze використовує створені користувачем звіти для надання інформації в режимі реального часу про умови дорожнього руху, аварії, перекриття доріг та інші інциденти [15], [16];
- динамічна маршрутизація: застосунок динамічно коригує маршрути на основі поточних умов дорожнього руху, прагнучи оптимізувати час у дорозі;
- взаємодія зі спільнотою: користувачі можуть повідомляти та обмінюватися інформацією про небезпеку дорожнього руху, присутність поліції та інші фактори, які можуть вплинути на досвід водіння;

– елементи гейміфікації: Waze включає елементи гейміфікації, які винагороджують користувачів за надання даних та роблять додаток привабливим [15], [16].

Модель, керована Користувачем. Дані про краудсорсинг: унікальність Waze полягає в тому, що він покладається на дані про краудсорсинг. Користувачі активно надають дані, повідомляючи про інциденти, оновлюючи карти і надаючи зворотний зв'язок [15], [16].

Залучення спільноти: Waze розвиває почуття спільноти серед користувачів. Чим активніше користувачі беруть участь, тим точнішою та ціннішою стає інформація для всіх користувачів [15], [16].

Інтеграція з великими даними:

– збір даних – Waze збирає та аналізує величезну кількість даних у режимі реального часу, включаючи звіти, створені користувачами, дані про місцезнаходження GPS та інформацію про швидкість;

– алгоритми машинного навчання – Waze використовує алгоритми машинного навчання для аналізу історичних даних та даних у режимі реального часу, підвищуючи точність прогнозів трафіку та пропозицій маршрутів;

– прогнозована аналітика: використовуючи аналітику великих даних, Waze може прогнозувати структуру трафіку, виявляти тенденції та пропонувати користувачам проактивне коригування маршрутів.

Успіх та ефективність Waze можна пояснити його плавною інтеграцією з технологіями великих даних. Наступні моменти підкреслюють, наскільки тісно Waze пов'язаний з великими даними [15], [16]:

– обробка даних в режимі реального часу – Waze використовує обробку даних у режимі реального часу для надання актуальної інформації про умови дорожнього руху. Технології великих даних, такі як Apache Kafka та Apache Flink, дозволяють програмі обробляти та аналізувати величезні потоки даних, що генеруються користувачами [15], [16];

– краудсорсинговий збір даних – основа Waze базується на даних про краудсорсинг. Великі дані полегшують збір, зберігання та аналіз цих даних користувачів, дозволяючи Waze збирати інформацію від мільйонів користувачів одночасно;

– алгоритми динамічної маршрутизації – алгоритми динамічної маршрутизації Waze, які коригують маршрути на основі умов реального часу, базуються на великих даних. Застосунок враховує історичні схеми руху, поточні інциденти, про які повідомляють користувачі, та інші динамічні фактори для оптимізації маршрутів на льоту [15], [16];

– машинне навчання для прогнозного аналізу – Waze використовує алгоритми машинного навчання для прогнозування майбутніх умов дорожнього руху. Аналізуючи історичні дані та вивчаючи поведінку користувачів, додаток може передбачити затори, запропонувати оптимальний час відправлення та забезпечити альтернативні маршрути [15], [16];

– моделі прогнозування трафіку – великі дані дозволяють Waze розробляти складні моделі прогнозування трафіку. Ці моделі враховують різні параметри, такі як час доби, погодні умови та спеціальні події, для прогнозування та усунення потенційних проблем з трафіком [15], [16];

– постійне вдосконалення за допомогою зворотного зв'язку – Waze заохочує відгуки користувачів та їх внесок. Аналітика великих даних обробляє цей зворотний зв'язок, щоб визначити області для вдосконалення, підвищити точність відображення та вдосконалити алгоритми, забезпечуючи безперервний цикл зворотного зв'язку для вдосконалення програми;

– аналітика, керована спільнотою – модель Waze, орієнтована на спільноту, спирається на колективний розум своїх користувачів. Технології великих даних допомагають керувати цим масивом даних і витягувати з нього корисну інформацію, створюючи симбіотичні відносини між застосунком і спільнотою користувачів [15], [16];

– масштабованість та продуктивність – оскільки мільйони користувачів надають дані одночасно, масштабованість має вирішальне

значення. Технології великих даних забезпечують масштабованість, необхідну для обробки постійно зростаючого обсягу даних користувачів, забезпечуючи оптимальну продуктивність застосунків [15], [16];

- візуалізація даних для користувачів – Waze використовує великі дані для візуального відображення умов дорожнього руху та аварій у своєму картографічному інтерфейсі. Здатність програми представляти складні дані в інтуїтивно зрозумілому та доступному для застосування форматі забезпечується надійними методами візуалізації даних [15], [16].

Великі дані дозволяють Waze адаптувати свої навігаційні стратегії на основі даних реального часу та історичних даних. Така адаптивність гарантує, що користувачі отримують найбільш актуальні та ефективні маршрути, покращуючи загальний досвід навігації [15], [16].

Переваги для користувачів і вплив:

- точна інформація про дорожній рух – використання Waze великих обсягів даних сприяє підвищенню точності інформації про дорожній рух. Користувачі отримують інформацію про дорожні умови в режимі реального часу, що дозволяє їм приймати обґрунтовані рішення щодо своїх маршрутів;

- економія часу і палива – динамічна маршрутизація, заснована на аналізі великих даних, допомагає користувачам економити час і паливо, уникаючи перевантажених районів і вибираючи найбільш ефективні маршрути;

- залучення спільноти – інтеграція великих даних у Waze сприяє відчуттю залучення спільноти. Користувачі активно сприяють підвищенню точності інформації в додатку, створюючи середовище для спільної роботи;

- попереджувальне коригування маршруту – використання Waze великих даних дозволяє попереджувально коригувати маршрут. Застосунок може прогнозувати зміни умов дорожнього руху та реагувати на них, надаючи користувачам альтернативні маршрути, перш ніж вони зіткнуться із затримками;

– покращений користувацький досвід – великі дані відіграють вирішальну роль у покращенні загального досвіду користувачів у Waze. Здатність програми адаптуватися, навчатися та надавати персоналізовану навігацію сприяє її популярності та задоволеності користувачів [15], [16].

Хоча інтеграція Waze з великими даними приносить багато переваг, існують також проблеми та міркування щодо конфіденційності:

– проблеми, пов'язані з конфіденційністю – збір даних про місцезнаходження для навігації краудсорсингу викликає проблеми конфіденційності. Waze вирішує цю проблему, дозволяючи користувачам контролювати свої налаштування конфіденційності [15], [16];

– точність даних – точність даних користувача залежить від участі користувача. У регіонах з меншою кількістю користувачів Waze це може вплинути на точність даних у режимі реального часу;

– безпека даних – як і в будь-якому застосунку, що працює з даними користувача, забезпечення безпеки даних має першорядне значення. Waze використовує шифрування та захищені протоколи для захисту інформації користувача;

– залежність від підключення – Waze покладається на постійне підключення до Інтернету для оновлення даних у режимі реального часу. У районах з поганим зв'язком користувачі можуть відчувати обмеження в отриманні точної і своєчасної інформації.

Waze – чудовий приклад того, як великі дані можуть революціонізувати навігацію та зручність поїздки. Завдяки плавній інтеграції даних краудсорсингу, обробки в режимі реального часу та алгоритмів машинного навчання Waze (рис. 1.4) дозволяє користувачам ефективніше орієнтуватися та приймати обґрунтовані рішення [15], [16]. Успіх програми тісно пов'язаний з його здатністю використовувати потужність великих даних, перетворюючи їх на корисну інформацію, яка приносить користь мільйонам користувачів у всьому світі. Оскільки технології продовжують розвиватися, Waze служить маяком, який демонструє потенціал керованих користувачем

застосунків, орієнтованих на дані, у формуванні майбутньої навігації та мобільності.

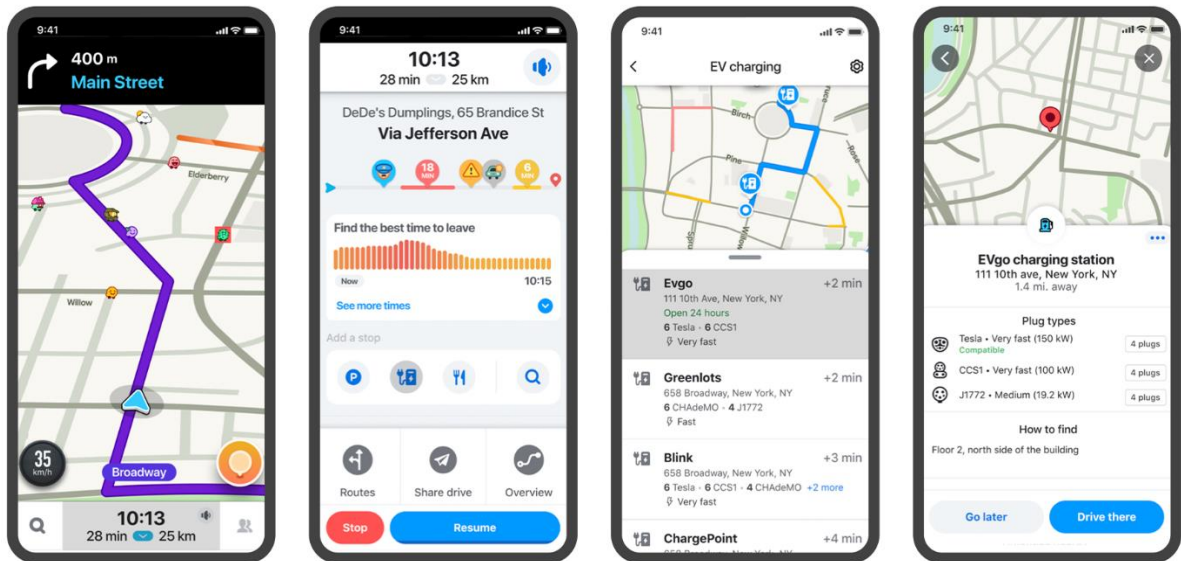


Рисунок 1.4 – Приклади графічного інтерфейсу застосунку Waze [16]

Порівняльну характеристику ПЗ для обробки великих даних, отриманих з транспортних засобів наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння ПЗ для обробки великих даних, отриманих з транспортних засобів

Критерій порівняння	Robi VTS	KACKO4U	Waze
Інтеграція великих даних	Немає	Немає	Є
Інтелектуальний аналіз накопичених даних	Формування аналітичних звітів	Відсутній	Робота з маршрутами онлайн
Зручність персоналізації	Середній рівень	Середній рівень	Середній рівень
Розширюваність	Через платну підписку	Відсутня	Відсутня
Кросплатформність	Тільки мобільні ОС	Тільки мобільні ОС	Підтримується

За результатами проведеного аналізу можна зробити висновок, що на сьогоднішній день існує досить велика кількість застосунків для обробки

даних, отриманих з транспортних засобів. Проте, більшість з них взагалі не використовують технології пов'язані з великими даними. Тобто, застосунки в змозі накопичувати та обробляти користувацькі дані, але не можуть в повній мірі синхронізувати дані, наприклад, усіх користувачів, які знаходяться на одній місцевості.

Повноцінна робота із великими даними, до того ж в режимі реального часу, забезпечується виключно в застосунку Waze. Проте, і цей застосунок скоріше є ретранслятором накопичених даних, адже їх обробка зводиться до досить простих операцій.

Відповідно актуальною залишається задача: дослідження та програмної реалізації методів обробки великих даних, отриманих з транспортних засобів.

При розробці ПЗ обробки великих даних, отриманих з транспортних засобів, необхідно забезпечити такі функціональні вимоги:

- система повинна відповідати принципу горизонтальної масштабованості, так як передбачається обробка великих обсягів даних, система повинна бути легко розширювана;
- на вхід система повинна приймати CSV файли. При цьому система повинна вміти працювати з різними схемами даних в цих файлах, а єдиною умовою є наявність обов'язкових полів, описаних вище. Схема даних повинна передаватися в систему при конфігуруванні;
- система повинна вміти обробляти дані незалежно від похибки вимірювань;
- система повинна розбивати поїздки на ділянки руху з однаковим характером зміни напрямку і швидкості. За характером зміни напрямку передбачається розбивати ділянки руху на поворот та прямолінійний рух. За характером зміни швидкості передбачається розбивати ділянки поїздки на: ділянки гальмування, ділянки прискорення, ділянки рівномірного руху, ділянки з нульовою швидкістю, для виділених ділянок руху система повинна розраховувати інтервал часу та середню швидкість руху, абсолютну величину середнього прискорення, середню величину доцентрового прискорення;

- отриманий результат система повинна записувати в CSV файл;
- система повинна мати можливість запису попередніх результатів роботи для аналізу їх адекватності.

1.4 Висновки за розділом 1

В першому розділі було досліджено поняття великих даних. У сучасному швидко розвивається цифровому ландшафті великі дані відіграють ключову роль у стимулюванні інновацій, прийнятті обґрунтованих рішень та отриманні конкурентних переваг у різних галузях. Величезний обсяг, швидкість і різноманітність даних, що генеруються щодня, вимагають розширених можливостей аналітики та обробки. Великі дані дозволяють організаціям отримувати цінну інформацію, виявляти тенденції та підвищувати операційну ефективність. Важливість великих даних - від персоналізованого обслуговування клієнтів і прогностичної аналітики до досягнень охорони здоров'я та ініціатив розумного міста – полягає в їх здатності виявляти закономірності, обґрунтовувати стратегії і стимулювати трансформаційні зміни, в кінцевому рахунку формуючи те, як ми розуміємо складності сучасного світу і орієнтуємося в них.

За результатами проведеного аналізу ПЗ для обробки великих даних, отриманих з транспортних засобів, зроблено висновки, що на сьогоднішній день існує досить велика кількість застосунків для обробки даних, отриманих з транспортних засобів. Більшість з таких застосунків взагалі не використовують технології пов'язані з великими даними. Тобто, застосунки в змозі накопичувати та обробляти користувацькі дані, але не можуть в повній мірі синхронізувати дані, наприклад, усіх користувачів, які знаходяться на одній місцевості. Повноцінна робота із великими даними, до того ж в режимі реального часу, забезпечується виключно в одному застосунку. Проте, і цей застосунок скоріше є ретранслятором накопичених даних, адже їх обробка зводиться до досить простих операцій. Саме тому, актуальною залишається

задача: дослідження та програмної реалізації методів обробки великих даних, отриманих з транспортних засобів.

Сформульовано функціональні вимоги до ПЗ для обробки великих даних, отриманих з транспортних засобів.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

Java, універсальна та надійна мова програмування, традиційно асоціюється із застосунками корпоративного рівня та серверною розробкою. Однак його роль у науці даних набула популярності завдяки її унікальним функціям, незалежності від платформи та багатству бібліотек та фреймворків, доступних в екосистемі Java. Слід розглянути сильні сторони Java в галузі науки про дані [17].

Незалежність від платформи. Перевага: принцип Java – писати один раз, запускати де завгодно – дозволяє розробникам писати код на одній платформі та запускати його на будь-якій платформі, що підтримується Java, без змін. Така переносимість вигідна в проектах з обробки даних з різними вимогами до інфраструктури [17].

Масштабованість. Масштабованість Java робить її придатною для великомасштабної обробки даних. Її здатність ефективно управляти ресурсами та розподіляти навантаження на кластери сприяє її ефективності при аналізі великих даних [17].

Багатопоточність і паралелізм. Вбудована підтримка багатопоточності та паралельності Java корисна для паралельної обробки, критичного аспекту проблем науки про дані, які передбачають одночасні обчислення на великих наборах даних [17].

Багата екосистема. Java може похвалитися великою екосистемою бібліотек та фреймворків, які полегшують різні аспекти науки про дані, включаючи маніпулювання даними, Машинне навчання та візуалізацію. Серед відомих бібліотек-Apache Spark, Deeplearning4j та Weka [17].

Корпоративна інтеграція. Плавна інтеграція Java з корпоративними системами робить її ідеальним вибором для організацій з існуючою

інфраструктурою на базі Java. Вона забезпечує плавну інтеграцію рішень для обробки даних в більш великі корпоративні застосунки [17].

Підтримка спільноти. Java має велику та активну спільноту, яка сприяє її постійному вдосконаленню. Модель розробки, керована спільнотою, забезпечує доступність ресурсів, підтримки та оновлень, що робить Java надійним вибором для довгострокових проєктів [17].

Реальні застосування Java в data science.

Фінансова аналітика. Java широко використовується у фінансових установах для таких завдань, як управління ризиками, виявлення шахрайства та алгоритмічна торгівля. Масштабованість Java робить її придатною для обробки величезних обсягів фінансових даних у режимі реального часу [17].

Інформатика охорони здоров'я. Приклад використання: Java застосовується в проєктах з обробки медичних даних для таких завдань, як аналіз даних пацієнтів, прогнозування захворювань та обробка медичних зображень. Незалежність від платформи дозволяє медичним додаткам безперешкодно працювати в різних системах [17].

Персоналізація електронної комерції. Java використовується на платформах електронної комерції для персоналізованих систем рекомендацій. Можливості масштабованості та паралельності Java дозволяють ефективно обробляти дані про поведінку користувачів для отримання точних рекомендацій щодо продукту [17].

Оптимізація ланцюга поставок. Java використовується в управлінні ланцюгами поставок для оптимізації логістики та запасів. Її масштабованість має вирішальне значення для обробки великих наборів даних, що використовуються для відстеження та оптимізації процесів ланцюга поставок [17].

Аналітика в соціальних мережах. Java використовується для аналізу величезної кількості даних соціальних мереж для отримання інформації про поведінку користувачів, аналізу настроїв та прогнозування тенденцій.

Незалежність Java від платформи гарантує, що програми для аналізу соціальних мереж можуть працювати на різних платформах [17].

Прогнозування енергоспоживання. Java застосовується в проектах data science, пов'язаних з прогнозуванням енергоспоживання. Його масштабованість та підтримка паралельної обробки сприяють підвищенню ефективності алгоритмів, що використовуються для прогнозування попиту на енергію [17].

Однак, слід детально розглянути й існуючі проблеми.

Крива навчання. Фахівці з обробки даних, які переходять на Java, можуть зіткнутися з труднощами в навчанні, особливо якщо вони походять із середовища, де Python або R більш поширені в завданнях, пов'язаних з наукою про дані [17].

Акцент спільноти на Python. Незважаючи на те, що Java має сильну спільноту, спільнота науковців даних часто наголошує на таких мовах, як Python, за їх простоту та великі бібліотеки. Це може призвести до скорочення ресурсів, спеціально призначених для Java в галузі науки про дані [17].

Ресурсоємність. Програми Java, особливо ті, що стосуються обробки великих обсягів даних, можуть бути ресурсоємними. Оптимізація має вирішальне значення для забезпечення ефективного використання системних ресурсів [17].

Інтерактивне дослідження даних. : Історична потужність Java в області пакетної обробки може створювати проблеми в сценаріях, де пріоритетним є інтерактивне дослідження і аналіз даних. У таких випадках часто віддають перевагу таким мовам, як Python, з інтерактивними середовищами [17].

Присутність Java у сфері науки про дані характеризується її універсальністю, масштабованістю та надійною екосистемою. Оскільки організації стикаються з постійно розширюваними наборами даних, незалежність Java від платформи, можливості багатопоточності та великі бібліотеки роблять її привабливим вибором для проектів науки про дані.

Інтеграція Java з потужними фреймворками, такими як Apache Spark і Deeplearning4j, ще більше розширює його можливості в обробці великомасштабних аналітичних даних [17]. Незважаючи на існуючі проблеми, реальні програми Java у різних областях демонструють її актуальність та ефективність у динамічній галузі науки про дані.

Обґрунтування вибору мови програмування для розробки ПЗ для обробки великих даних, отриманих з транспортних засобів наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення для обробки великих даних, отриманих з транспортних засобів

Критерій порівняння мов програмування	Мова програмування		
	Java	Python	C#
Підтримка парадигм програмування	Виключно ООП	Підтримує багато парадигм	Декілька парадигм
Надбудови та фреймворки для машинного навчання	Існують	Існують, багато	Існують
Активність спільноти	Висока	Найвища	Висока
Продуктивність застосунків	Висока	Середня	Висока
Розширюваність	Можлива	Можлива	Можлива

Отже, для реалізації ПЗ для обробки великих даних, отриманих з транспортних засобів, обрано мову програмування Java, яка є безкоштовною, ООП – орієнтовною, широко використовуваною для ПЗ різного призначення, а також має багато модулів для створення програм для оброблення зображень.

2.2 Вибір фреймворку

Розглянемо бібліотеки та фреймворки для аналізу даних на Java.

Apache Spark. Apache Spark – це потужний движок обробки даних з відкритим кодом для аналізу великих даних. Він надає API високого рівня на Java для розподіленої обробки даних, що робить його придатним для масштабних завдань з науки про дані. Пакетна обробка, Машинне навчання, обробка графіків та потокове передавання даних у реальному часі [18], [19].

Deeplearning4j – це бібліотека глибокого навчання для Java та Scala. Він інтегрується з Apache Spark і надає інструменти для побудови та навчання нейронних мереж. Програми глибокого навчання, включаючи розпізнавання зображень та мовлення, обробку природної мови та прогнозу аналітику [18], [19].

Weka – це набір алгоритмів машинного навчання для завдань видобутку даних. Він надає набір Java API для реалізації та експериментів з різними моделями машинного навчання. Класифікація, кластеризація, регресія та вибір функцій у проектах видобутку даних [18], [19].

Apache Flink – це платформа потокової обробки для аналізу великих даних. Вона забезпечує обробку потоків даних в режимі реального часу і пропонує Java API для побудови масштабованих і відмовостійких конвеєрів даних. Аналітика в реальному часі, програми, керовані подіями, та обробка безперервних потоків даних [18], [19].

Mallet – це інструментарій для обробки природної мови на базі Java. Він включає інструменти для класифікації документів, тематичного моделювання та інших завдань обробки природної мови. Інтелектуальний аналіз тексту, аналіз настроїв і витяг інформації з великих текстових наборів даних [18], [19].

ELKI – це програмний фреймворк для інтелектуального аналізу даних, написаний на Java. Він фокусується на алгоритмах навчання без контролю та

забезпечує модульну та розширювану платформу для експериментів. Кластерний аналіз, виявлення викидів та видобуток шаблонів у великих наборах даних [18], [19].

Hadoop, платформа з відкритим кодом, призначена для розподіленого зберігання та обробки великих наборів даних. Створена в зв'язку з необхідністю вирішення проблем, пов'язаних з великими даними, Hadoop полегшує ефективне управління і аналіз величезних обсягів інформації. Розуміння основних компонентів Hadoop [18], [19].

Розподілена файлова система Hadoop (Hadoop Distributed File System HDFS). Основний компонент: HDFS – це розподілена файлова система Hadoop, призначена для зберігання величезної кількості даних на декількох вузлах кластера. Вона забезпечує відмовостійкість і високу доступність за рахунок реплікації даних між вузлами. Перевага: здатність HDFS керувати петабайтами даних та зберігати їх у безпечному режимі формує основу для можливостей обробки даних Hadoop [18], [19].

MapReduce – це програмна модель, яка дозволяє паралельно обробляти великі набори даних у кластері Hadoop. Вона складається з Фази Map для обробки даних і фази Reduce для агрегування. MapReduce забезпечує ефективну обробку розподілених даних, що робить його придатним для таких завдань, як пакетна обробка, аналіз журналів та складні перетворення даних [18], [19].

Переваги Hadoop у проектах, пов'язаних з data science.

Масштабованість. Розподілена Архітектура Hadoop забезпечує плавну масштабованість. У міру зростання обсягів даних в кластер можна додавати додаткові вузли, гарантуючи, що Hadoop зможе справлятися зі зростаючими вимогами проектів в галузі науки про дані [18], [19].

Економічна ефективність Hadoop обумовлена його здатністю працювати на стандартному обладнанні. Організації можуть створювати кластери зі стандартними серверами, уникаючи необхідності в дорогій спеціалізованій інфраструктурі.

Відмовостійкість. Hadoop розроблений для коректної обробки апаратних збоїв. Дані, що зберігаються в HDFS, реплікуються між вузлами, гарантуючи, що навіть у разі збою вузла дані залишаються доступними і обробка може продовжуватися без збоїв [18], [19].

Паралельна обробка. Здатність Hadoop розділяти завдання на менші підзадачі та обробляти їх паралельно на різних вузлах прискорює обробку даних. Цей паралелізм особливо корисний для завдань, які можна розділити на незалежні блоки, такі як багато алгоритмів обробки даних.

Гнучкість з типами даних. Hadoop може обробляти різні типи даних, включаючи структуровані, напівструктуровані та неструктуровані дані. Така гнучкість має вирішальне значення в наукових проектах даних, де необхідно обробляти та аналізувати різні формати даних [18], [19].

Підтримка пакетної обробки. Початкова сила Hadoop полягає в пакетній обробці. Для проектів науки про дані, які передбачають обробку великих обсягів даних із запланованими інтервалами, підтримка пакетно-орієнтованих завдань у Hadoop є перевагою [18], [19].

Екосистема інструментів та бібліотек. Hadoop створив багату екосистему інструментів і бібліотек, які розширюють його функціональність. Від Apache Spark для обробки даних у пам'яті до Apache Hive для запитів, подібних до SQL, ця екосистема розширює можливості Hadoop у різних програмах обробки даних [18], [19].

Реальні застосування Hadoop у data science.

Прогнозна аналітика. Hadoop використовується для завдань прогнозовної аналітики, які включають обробку історичних даних для виявлення закономірностей і тенденцій. Особливо це стосується таких галузей, як фінанси для оцінки ризиків та маркетинг для аналізу поведінки клієнтів [18], [19].

Машинне навчання. Hadoop, поряд з такими інструментами, як Apache Mahout, підтримує масштабну реалізацію алгоритмів машинного навчання. Це

застосовується в різних областях, включаючи системи рекомендацій, виявлення шахрайства та розпізнавання зображень.

Аналіз журналів. Hadoop використовується для аналізу файлів журналів, що генеруються додатками, серверами або вебсайтами. Цей аналіз допомагає зрозуміти поведінку користувачів, виявити проблеми з продуктивністю та отримати цінну інформацію для оптимізації [18], [19].

Геноміка та біоінформатика. У геноміці та біоінформатиці, де в результаті секвенування ДНК та наукових досліджень генеруються великі набори даних, Hadoop використовується для ефективного зберігання, вилучення та аналізу геномних даних. Це сприяє прогресу в персоналізованій медицині та генетичних дослідженнях [18], [19].

Аналітика в соціальних мережах. Hadoop відіграє важливу роль в обробці та аналізі величезних обсягів даних з платформ соціальних мереж. Це включає аналіз настроїв, виявлення тенденцій та аналіз поведінки користувачів, надаючи цінну інформацію для маркетингу та прийняття рішень [18], [19].

Оптимізація ланцюга поставок. Hadoop застосовується в управлінні ланцюгами поставок для оптимізації логістики, інвентаризації та розподілу. Аналіз великих наборів даних допомагає організаціям оптимізувати процеси ланцюга поставок для підвищення ефективності та економічної ефективності [18], [19].

Hadoop відіграє трансформаційну роль у проектах науки про дані, пропонуючи масштабовані та економічно ефективні рішення для обробки та аналізу великих наборів даних. Його розподілена архітектура, відмовостійкість і можливості паралельної обробки роблять його потужним інструментом для вирішення завдань, пов'язаних з великими даними. Реальні застосування Hadoop в прогностичній аналітиці, машинному навчанні та різних галузях промисловості підкреслюють його актуальність та вплив [18], [19].

Оскільки галузь науки про дані продовжує розвиватися, Hadoop залишається основоположною технологією, доповненою зростаючою екосистемою інструментів та бібліотек. Незважаючи на існуючі проблеми, переваги масштабованості, гнучкості та процвітаючої спільноти сприяють постійному значенню Hadoop у галузі науки про дані. Оскільки організації прагнуть отримати корисну інформацію з дедалі складніших наборів даних, Hadoop виступає надійним союзником у пошуку знань за допомогою даних [18], [19].

2.3 Вибір середовища розробки

Eclipse, широко використовуване інтегроване середовище розробки (Integrated Drive Electronics – IDE), зарекомендувало себе як кращий інструмент для проектів Java Hadoop. Оскільки екосистема Hadoop продовжує відігравати ключову роль у обробці та аналітиці великих даних, ефективність та функції, пропоновані Eclipse, суттєво сприяють розробці, налагодженню та керуванню додатками Hadoop [20].

Універсальність. Eclipse – універсальне середовище розробки, що підтримує широкий спектр мов програмування. Для розробників Java, які працюють над проектами Hadoop, можливість плавного перемикання між Java, XML та іншими мовами є значною перевагою [20].

Багата екосистема плагінів. Архітектура плагінів Eclipse дозволяє розробникам розширювати своє середовище розробки за допомогою широкого спектру плагінів. Ця розширюваність має вирішальне значення для безперебійної інтеграції інструментів та функцій, специфічних для розробки на Hadoop [20].

Спільнота з відкритим кодом. Відкритий вихідний код Eclipse сприяє створенню спільної спільноти, яка постійно вносить свій внесок у його вдосконалення. Ця модель розробки, керована спільнотою, забезпечує регулярні оновлення, виправлення помилок та доступність нових функцій [20].

Кросплатформна підтримка. Eclipse розроблений як кросплатформний, що робить його доступним у середовищах Windows, macOS та Linux. Ця кросплатформна підтримка цінна для команд, що працюють в різних операційних системах [20].

Функції Eclipse для проектів Java Hadoop. Інструменти розробки Java (Java Development Tools – JDT). JDT – це набір інструментів у Eclipse, спеціально розроблений для розробки Java. Він включає такі функції, як завершення коду, Налагодження та інтегрована компіляція, забезпечуючи надійне середовище для програмування Java у проектах Hadoop [20].

Вбудований налагоджувач. Вбудований налагоджувач Eclipse дозволяє розробникам легко налагоджувати програми Java Hadoop. Ця функція безцінна для виявлення та усунення проблем на етапах розробки та тестування.

Інтеграція з Maven. Eclipse має надійну інтеграцію з Apache Maven, широко використовуваним інструментом автоматизації збірки. Ця інтеграція полегшує управління залежностями, конфігураціями проектів та збірками в проектах Hadoop, що використовують Maven [20].

Редагування та перевірка XML. Проекти Hadoop часто включають великі конфігурації XML. Eclipse надає комплексні інструменти редагування і перевірки XML, що гарантують коректність конфігураційних файлів і поліпшують процес розробки [20].

Інтеграція з системою контролю версій. Eclipse легко інтегрується з системами контролю версій, такими як Git і SVN. Ця інтеграція забезпечує спільну розробку та відстеження версій, що важливо для команд, що працюють над проектами Java Hadoop [20].

Шаблони коду та фрагменти. Eclipse пропонує шаблони коду та фрагменти, які прискорюють розробку, забезпечуючи заздалегідь визначені структури коду. Це особливо корисно в проектах Hadoop, де зазвичай використовуються певні моделі та практики [20].

Інструменти рефакторингу. Eclipse надає потужні інструменти рефакторингу, які допомагають розробникам реструктурувати та оптимізувати

свій код. Ця можливість має вирішальне значення для підтримки якості коду та читабельності у складних проектах Hadoop.

Інструменти проектування графічного інтерфейсу користувача (Graphical user interface – GUI). Eclipse підтримує інструменти проектування GUI, які спрощують створення графічних інтерфейсів для додатків Hadoop. Це корисно для проектів, які включають взаємодію з користувачем або компоненти візуалізації [20].

Eclipse виступає в якості потужної IDE для проектів Java Hadoop, пропонуючи багатофункціональне середовище, надійні інструменти розробки на Java і велику екосистему плагінів. Його універсальність, вбудований налагоджувач і підтримка різних плагінів, пов'язаних з Hadoop, роблять його ідеальним вибором для розробників, що займаються розробкою додатків Hadoop [20].

Оскільки організації продовжують використовувати потужність великих даних, роль Eclipse в оптимізації життєвого циклу розробки проектів Hadoop залишається ключовою. Від написання і налагодження коду до інтеграції з системою контролю версій і створення призначених для користувача додатків Eclipse надає комплексну платформу для розробників Java, орієнтовану в складнощах екосистеми Hadoop. Незважаючи на існуючі проблеми, популярність Eclipse в сфері розробки на Java зміцнює її позиції в якості цінного інструменту для проектів Java Hadoop, що сприяє безперешкодному створенню масштабованих і ефективних рішень для роботи з великими даними [20].

Обґрунтування вибору середовища розробки для створення ПЗ для обробки великих даних, отриманих з транспортних засобів наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення ПЗ для обробки великих даних, отриманих з транспортних засобів

Критерій порівняння середовищ розробки	Середовище розробки	
	Eclipse	IntelliJ IDEA
Підтримка Hadoop	Так	Так
Розробка застосунків із розподіленими обчисленнями	Так	Так
Засоби рефакторингу	Так	Так
Тестування коду	Так	Так
Безкоштовність	Безкоштовне	Платне

Таким чином, для створення ПЗ для обробки великих даних, отриманих з транспортних засобів, обрано середовище розробки Eclipse, що постачається з багатьма бібліотеками та модулями (плагінами), пакетами та інструментами, які прискорюють процес розробки на Java, значно прискорюють процес розробки для реалізації складаних проєктів.

2.4 Налаштування та додаткові механізми фреймворку

HDFS є блочно-структурованої файлової системою: окремі файли розбиваються на блоки фіксованого розміру. Ці блоки зберігаються в кластері або в одній з машин з великим обсягом для зберігання даних. Окремі машини в кластері називаються вузлами даних (DataNodes). Файл може складатися з декількох блоків, і вони необов'язково будуть зберігатися на одній і тій же машині; цільові машини, кожна з яких зберігає по блоку, вибираються випадково, за принципом «block-by-block» [21]–[23].

У HDFS файли зберігаються як набір великих блоків, розподілених між кількома машинами, тому ці файли не є частиною звичайної файлової

системи. HDFS використовує окремий простір імен, ізольований від вмісту локальних файлів. Файли всередині HDFS зберігаються в окремому каталозі, керованому сервісом DataNode (рис. 2.1). Взаємодіяти з файлами HDFS за допомогою звичайних інструментів файлу в Linux не можна. Однак, в HDFS входить її власна утиліта управління, що працює приблизно так само, як ці інструменти [21]–[23].

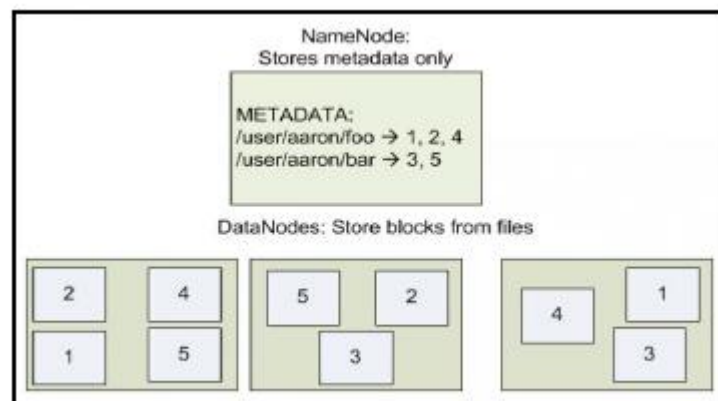


Рисунок 2.1 – Приклад структури HDFS

Структура метаданих у HDFS змінюється великою кількістю одночасних клієнтів. Для збереження її цілісності всі звернення обробляється однією машиною, званої NameNode. NameNode зберігає всі метадані файлової системи [21]–[23].

Щоб відкрити файл, клієнт отримує доступ до NameNode і отримує список розташування блоків, що складають файл. Ці адреси ідентифікують DataNode, який зберігає кожен блок. Потім клієнти зчитують інформацію безпосередньо з серверів DataNode, можливо, паралельно [21]–[23].

На відміну від більшості блочно-структурованих систем, де використовується розмір блоку близько 4 або 8 Кб, в HDFS розмір блоку за замовчуванням – 64 Мб. За рахунок цього в HDFS знижений обсяг сховища метаданих, необхідних для кожного файлу. Також це дозволяє прискорити потокове читання даних, оскільки великі обсяги інформації послідовно викладені на диску [21]–[23].

Обробка даних за допомогою Hadoop [21]–[23]. Для обробки даних за допомогою Hadoop можна написати Mr - програму, скориставшись streaming-інтерфейсом Hadoop або java MapReduce API. Також можна скористатися різними додатковими фреймворками, що встановлюються поверх Hadoop. Найбільш популярними з них є Apache Hive, Apache Pig, Cascading, Apache Spark. Фреймворки Apache Storm і Apache Spark також надають можливості по розподіленій обробці великих потоків даних в реальному часі. Робота з даними в Hadoop може проводитися з використанням трьох движків: стандартного MapReduce, Apache Spark і Apache Tez.

MapReduce – це модель розподіленої обробки даних, запропонована компанією Google для обробки великих обсягів даних на комп'ютерних кластерах. Вперше вона була описана в 2005 році співробітниками Google Джеффрі Діном і Санжая Гемаватом [21]–[23].

Робота MapReduce передбачає, що вихідні дані представлені у вигляді деяких записів. Обробка даних відбувається в 3 стадії (рис. 2.2).

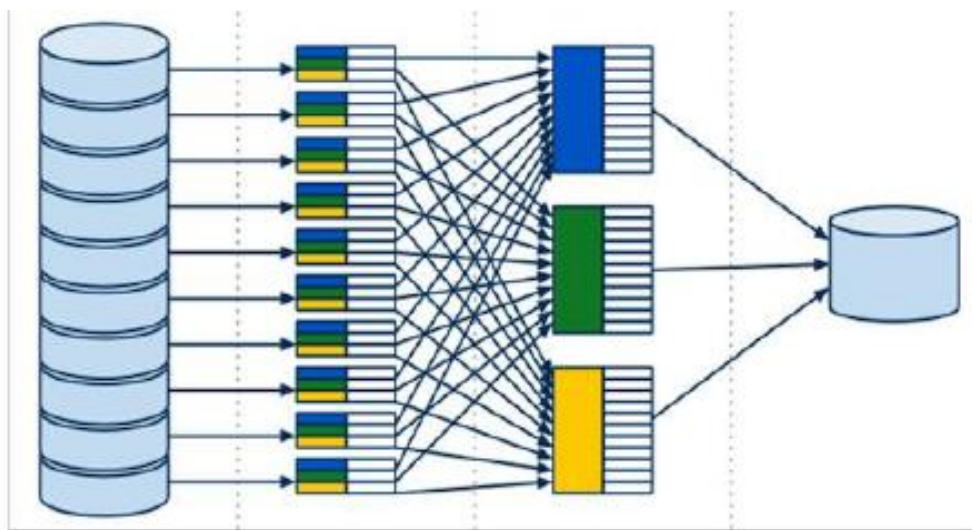


Рисунок 2.2 – Схема MapReduce задачі

Мар. На стадії мар відбувається попередня обробка вихідних даних за допомогою функції `map()`, яка визначається користувачем. Робота цієї стадії полягає в попередній обробці і фільтрації даних [21]–[23].

Функція `map()` застосовується до одного вхідного запису і видає безліч пар ключ-значення. Зміст ключа і значення визначається розробником, при цьому дані з однаковим ключем в подальшому потрапляють в один екземпляр функції `reduce()` [21]–[23].

Shuffle. Ця стадія проходить непомітно для розробника. В її процесі відбувається розбиття виведення функції `map()` «по кошиках», де кожна корзина відповідає одному ключу виведення стадії `map`. Ці кошики служать входом для стадії `reduce` [21]–[23].

Reduce. На цій стадії кожний кошик зі значеннями, сформована раніше на стадії `shuffle`, потрапляє на вхід функції `reduce()`.

Функція `reduce()` визначається розробником і обчислює фінальний результат для кожної окремої «кошика». Отримане в результаті роботи функції `reduce()` безліч значень є фінальним результатом MapReduce-завдання.

Деякі властивості MapReduce:

- всі запуски функцій `map()` і `reduce ()` можуть працювати незалежно один від одного, отже, можливо їх паралельне виконання, в тому числі і на різних машинах кластера;
- стадія `Shuffle` практично є паралельним сортуванням, а отже, також може працювати паралельно на різних машинах.

Ці два аспекти забезпечують принцип горизонтальної масштабованості.

Зазвичай функція `map()` виконується на тій самій машині, де зберігаються дані, що відповідає принципу локальності даних [21]–[23].

Взагалі кажучи, для деяких типів завдань (фільтрація даних, робота зі сховищами даних) можна обійтися тільки стадією `map`.

Крім того, при реалізації MapReduce-задачі в Hadoop, для попередньої агрегації результатів декількох `map` функцій можна визначити комбінуючу функцію (рис. 2.3). За своєю структурою комбінуюча функція схожа на `reduce` функцію-вона приймає на вхід вихід частини `map` функцій і видає для них агрегований результат [21]–[23]. Найчастіше можна використовувати `reduce` функцію як комбінуючу. Відмінністю комбінуючої функції від `reduce` є те, що на

неї потрапляють не всі значення, відповідні одному ключу. Використання combine функції може істотно збільшити швидкість виконання MapReduce - завдання. Однак комбінуюча функція не може бути застосовна для всіх типів завдань (наприклад, для пошуку медіанного значення по ключу, де необхідна попередня сортування всіх значень) [21]–[23].

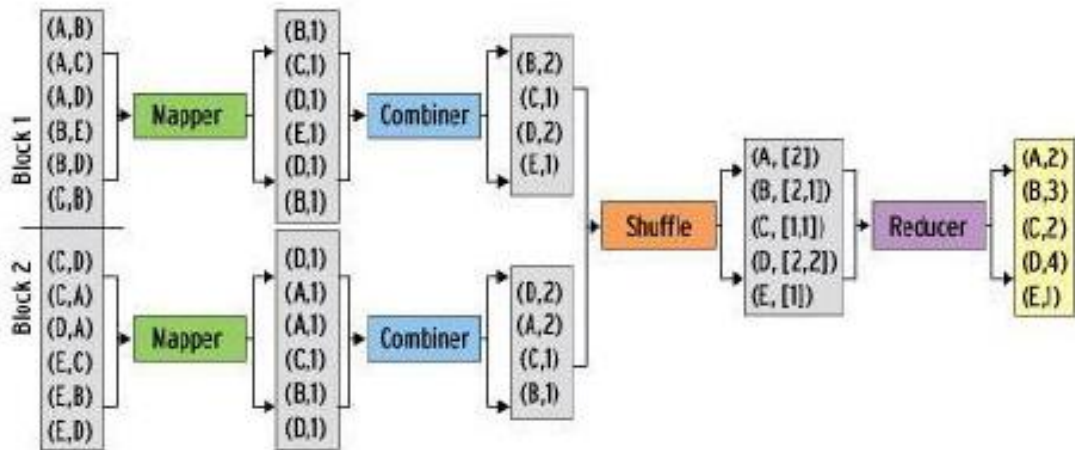


Рисунок 2.3 – Використання комбінуючої функції

Також варто відзначити, що до складу дистрибутивів Hadoop входять реалізації різних конкретних базових map і reduce функцій, найбільш часто використовуваних в розподіленій обробці [21]–[23].

Spark і TEZ рушії. Основним недоліком реалізації MapReduce в Hadoop є те, що в більшості реальних завдань не обійтися єдиною MR-роботою. Зазвичай при обробці даних проводяться кілька послідовних і паралельних операцій. Реалізація MR в Hadoop передбачає, що всі результати – як кінцеві, так і проміжні – записуються на диск. В результаті час роботи з диском може в кілька разів перевищувати час самих обчислень [21]–[23].

Для боротьби з цією проблемою розроблені два інших движка – Spark і Tez. Spark використовує спеціалізовані примітиви для рекурентної обробки в оперативній пам'яті, завдяки чому дозволяє отримувати значний вииграш у швидкості роботи для деяких класів завдань. Tez представляє задачу у вигляді спрямованого ациклічного графа (DAG) компонентів-обробників (рис. 2.4).

Планувальник запускає обчислення графа і, при необхідності, має можливість динамічно переконфігурувати його, оптимізуючи під дані [21]–[23].

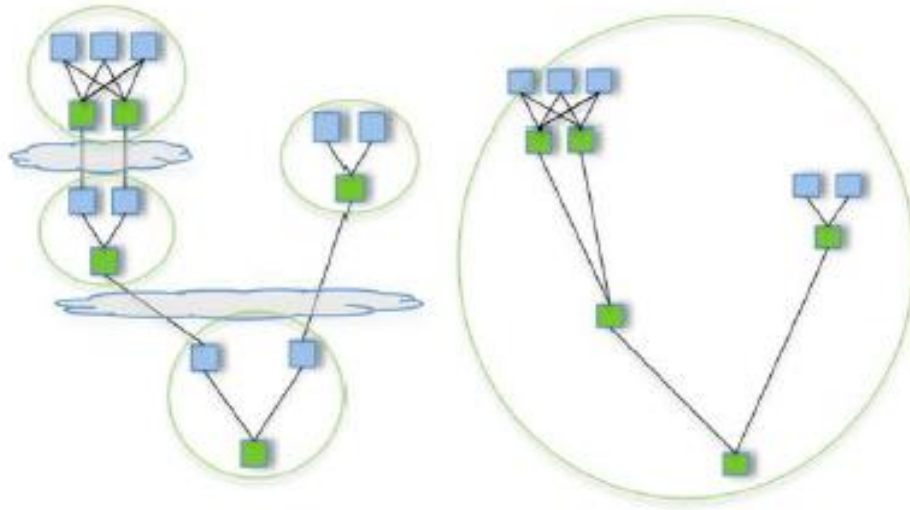


Рисунок 2.4 – Виконання завдання в режимах MR і Tez

2.5 Архітектура системи

Система, що розробляється, буде реалізована за допомогою платформи Apache Pig на фреймворці Hadoop. Рішення про використання даного фреймворку було прийнято з наступних причин [21]–[23].

Через необхідність забезпечення постійного зв'язку транспортних засобів з серверами для обробки даних в реальному часі вимагає додаткових витрат і не є необхідною умовою для вирішення поставлених завдань (виявлення характеристик поїздки і водіння), передбачається обробляти накопичені дані (наприклад, за один день або тиждень). Тому виключається використання фреймворку Apache Storm [21]–[23].

Через необхідність виявлення характеристик на основі вихідних даних є нетривіальним завданням, її реалізація в рамках Hive SQL буде складною за рахунок того, що Hive SQL передбачає декларативний підхід.

Так як, передбачається отримувати з вихідних даних набір записів, кожна з яких буде описувати ділянку поїздки з незмінними характеристиками.

Таким чином, з отриманих даних буде можливо отримати різні цікавлять характеристики за допомогою найпростіших запитів. З цієї причини використання Apache Pig видається більш доцільним, ніж використання Cascading, так як в Pig такі запити реалізуються на більш високому рівні абстракції [21]–[23].

Виконання Pig сценаріїв можливо в режимах MapReduce і Tez. Крім того, останнім часом з'являються різні доповнення, що дозволяють запускати Pig сценарії на Spark движку. У цьому сенсі використання Apache Pig є більш універсальним підходом, ніж використання Apache Spark [21]–[23].

Для зберігання даних буде використовуватися файлова система HDFS.

Обробка даних в системі складається з наступних етапів:

- завантаження даних з HDFS в Apache Pig;
- видалення зайвих даних: у деяких файлах, що описують пересування транспортних засобів, присутня інформація з різних інших сенсорів і систем, крім системи геопозиціонування [21]–[23]. Видалення цих даних на початковому етапі обробки дозволяє збільшити продуктивність системи;
- фільтрація даних: вихідна інформація може бути сильно зашумлена за рахунок похибок вимірювання. Необхідно мінімізувати вплив цих неточностей на кінцевий результат;
- виявлення точок зміни напрямку руху;
- виявлення точок зміни прискорення руху;
- розбиття траєкторії руху на ділянки;
- виявлення необхідних характеристик;
- запис результату в HDFS.

Як приклад використання отриманих даних була проведена кластеризація водіїв за обраними характеристиками. Для цього, спочатку було виконано агрегування отриманих характеристик для кожного водія, а потім проведений кластерний аналіз [21]–[23].

Архітектура розроблюваної системи наведена на рис. 2.5.



Рисунок 2.5 – Архітектура розроблюваної системи

2.6 Висновки за розділом 2

Для реалізації програмного забезпечення для обробки великих даних, отриманих з транспортних засобів обрано мову програмування Java та фреймворк Hadoop. Обрані технічні засоби є безкоштовними та мають активну спільноту розробників.

Для створення програмного забезпечення для обробки великих даних, отриманих з транспортних засобів обрано середовище розробки Eclipse, що постачається з багатьма модулями, пакетами та інструментами/

Розроблено модель взаємодії модулів ПЗ для для обробки великих даних, отриманих з транспортних засобів, з урахуванням використання модулів фреймворку.

3 ОПИС ПРОГРАМИ

3.1 Завантаження даних з HDFS і видалення зайвої інформації.

Розроблювана система повинна завантажувати дані, розташовані в CSV файлах в HDFS, в Apache Pig [22]–[24].

Існує кілька стандартних способів завантаження даних з CSV файлів.

За допомогою вбудованої функції `PigStorage('delimiter')`. Функція `PigStorage()` є вбудованою функцією в Apache Pig і дозволяє завантажувати та записувати дані. Роздільник даних (`delimiter`) при цьому конфігурується.

Приклад використання `PigStorage ()` для завантаження даних [22]–[24]:

```
data = LOAD 'input-file'
USING PigStorage(',')
AS (field0:long, field1:double);
```

За допомогою функції `CSVExcelStorage ()`. Функція `CSVExcelStorage ()` має кілька додаткових параметрів, які дозволяють підлаштовувати завантаження даних під формат вхідного CSV файлу [22]–[24].

Приклад використання `CSVExcelStorage ()` для завантаження даних [22]–[24]:

```
data = LOAD 'input-file'
USING
org.apache.pig.piggybank.storage.CSVExcelStorage(',', '
YES_MULTILINE', 'NOCHANGE', 'SKIP_INPUT_HEADER')
AS (field0:long, field1:double);
```

Перший параметр дозволяє задавати роздільник даних.

Другий параметр (`YES_MULTILINE` / / `NO_MULTILINE`) дозволяє визначати можливість завантаження полів, що складаються з декількох рядків.

Третій параметр використовується тільки при записі даних. При завантаженні тут необхідно вказувати `'NOCHANGE'`.

Перший рядок CSV файлу може містити заголовок з назвою полів. Четвертий параметр функції дозволяє не завантажувати заголовок разом з іншими даними [22]–[24].

Недоліком обох підходів є необхідність описувати схему даних в ас блоці. Робота системи перевірялася на двох типах файлів, кожен тип містив більше сорока різних полів, з яких тільки вісім були необхідні для аналізу. Використання стандартних функцій завантаження для роботи з такими файлами вимагало громіздкого опису структури даних для кожного типу даних безпосередньо в самому сценарії. При цьому Apache Pig завантажувач би також і великий набір непотрібних даних, що впливало б на швидкість роботи системи [22]–[24].

Для боротьби з цими недоліками, був написаний власний завантажувач, який приймає на вхід три параметри: роздільник даних, заголовок CSV файлу, і JSON об'єкт, що містить тільки назви потрібних полів з описом типу даних в цих полях. Спочатку планувалося, що завантажувач буде отримувати заголовок безпосередньо з файлу, але API Apache Pig вимагає вказівку структури вхідних даних до початку роботи з файлом [22]–[24].

Використання розробленого завантажувача:

```
data = load '$INPUT_LOG_FILE' using
    com.motorolasolutions.bigdata.driving.pig.loader.CS
VLoader('$INPUT_DELIMITER', '$INPUT_HEADING',
'$INPUT_SCHEMA')
    as (DEVICE_ID:long, TRIP:long, TIME:long,
LAT:double, LON:double, SPEED:double, HEADING:double,
HDOP:double);
```

Опис схеми тут необхідно, так як назви необхідних полів в різних файлах можуть відрізнятися. При цьому описуються і завантажуються тільки потрібні вісім полів, а не більше сорока полів, що містяться у файлі [22]–[24].

3.2 Фільтрація даних

3.2.1 Вплив похибок у вихідних даних на роботу системи

Похибки і неточності встановленої в транспортному засобі системи геопозиціонування можуть бути обумовлені цілим рядом причин. Серед них можна виділити:

- втрату сигналу від супутника;
- зміни в геометрії розташування супутників;
- відображення сигналів;
- обчислювальні помилки і помилки при округленні.

Таким чином, отримана інформація може не відповідати маршруту об'єкта з необхідною для аналізу даних точністю [22]–[24].

Для аналізу існуючих у вихідних файлах неточностей в описі переміщення транспортних засобів і для оцінки ступеня впливу цих неточностей на подальшу обробку даних було вирішено відобразити пересування об'єктів на географічній карті [22]–[24].

З цією метою використовувався фреймворк Mapbox Studio. Найбільш простим способом відображення траєкторії в цьому фреймворку було використання в якості вхідних даних файлу в форматі GeoJSON.

GeoJSON – це відкритий формат, призначений для зберігання географічних структур даних, заснований на JSON. За допомогою GeoJSON можна зберігати примітивні типи для опису географічних об'єктів, наприклад: точки (адреси та місця розташування), лінії (вулиці, шосе, межі), полігони (країни, Штати, ділянки землі). Також є можливість для зберігання так званих мультитипов, що представляють собою об'єднання декількох примітивних типів [22]–[24].

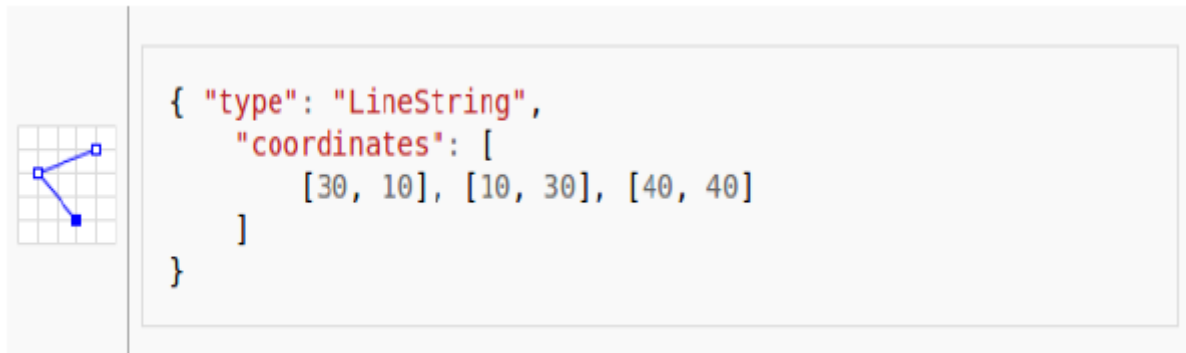


Рисунок 3.1 – Приклад GeoJSON файлу

Вихідні дані були згруповані за ідентифікаторами поїздок і записані в окремі CSV файли. Для перетворення даних з CSV в GeoJSON був написаний java клас, який записує координати з CSV файлу в якості координат об'єкта LineString [22]–[24].

При відображенні отриманих даних на карті вийшла наступна картина (рис. 3.2).

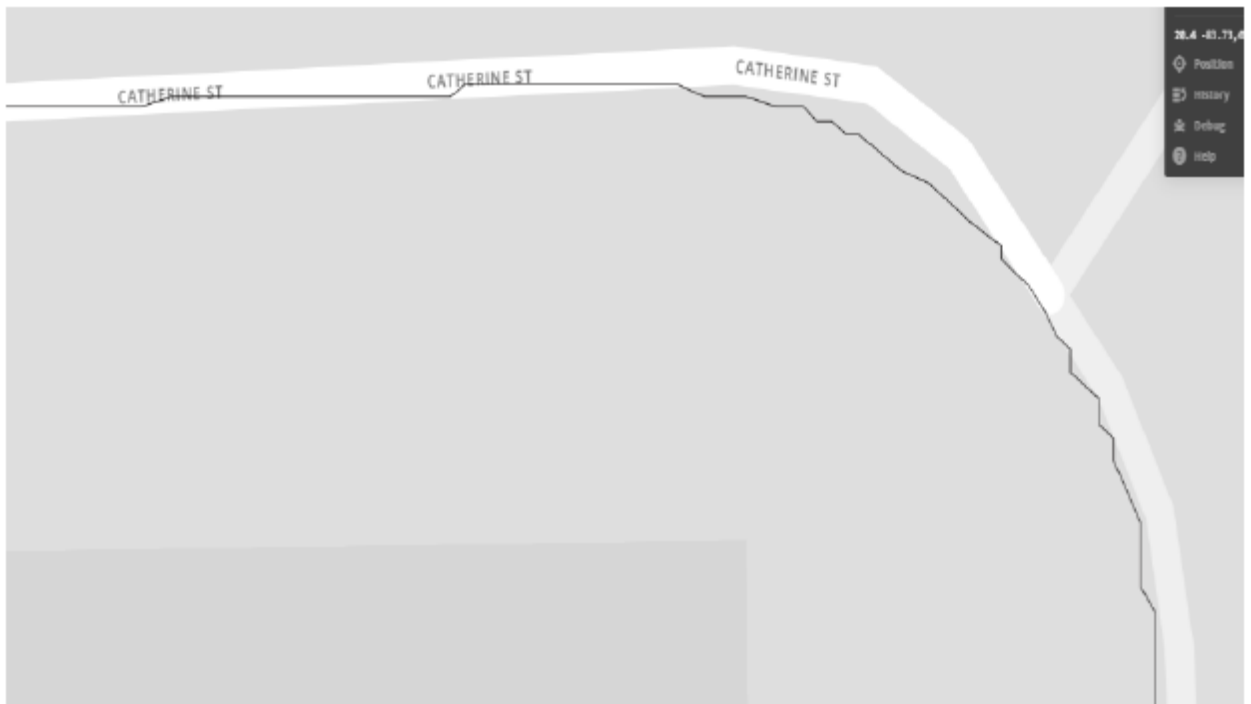


Рисунок 3.2 – Відображенні отриманих даних на карті

Наступним етапом обробки даних є виявлення точок зміни напрямку руху. Очевидно, що використання вихідних даних про координати об'єкта в цих цілях може привести до серйозного збільшення кількості зайвих точок, невідповідних реальній зміні напрямку [22]–[24].

Використання вихідних даних про напрямок або ковзного середнього для напрямку також може дати великі неточності в остаточному результаті.

На наступних двох графіках (рис. 3.3 і рис. 3.4) представлені залежності напрямку (синім кольором) і ковзного середнього для напрямку (червоним кольором).

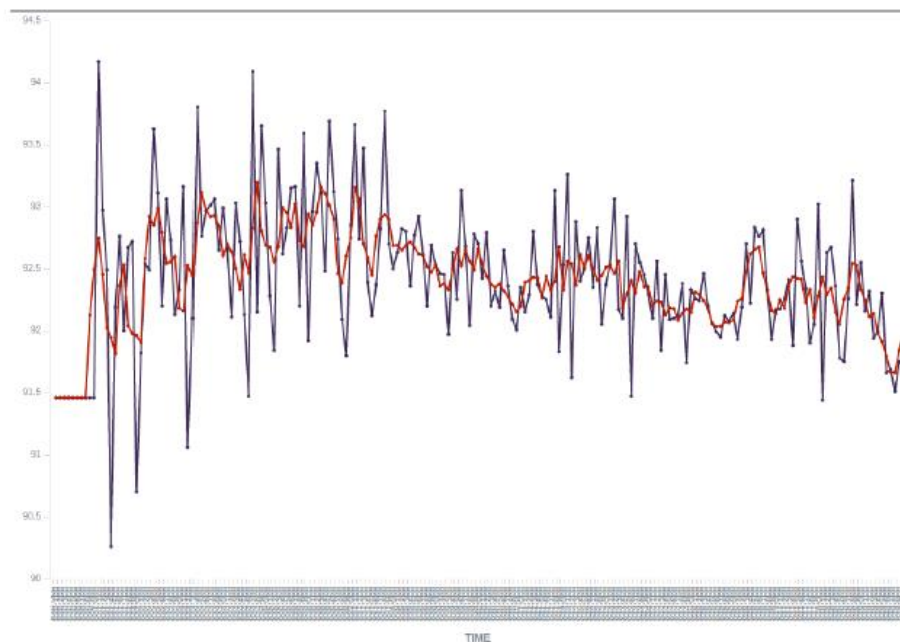


Рисунок 3.3 – Залежності напрямку і ковзного середнього для напрямку

В даному випадку інтервал для ковзного середнього обраний рівним 0.4 с. Очевидно, що такий інтервал недостатньо згладжує коливання значень напрямку, отже виявлення точок зміни напрямку за цим значенням не буде ефективним [22]–[24].

Збільшення інтервалу згладжує коливання, однак при цьому може сильно зрушувати точки зміни напрямку (рис. 3.4).

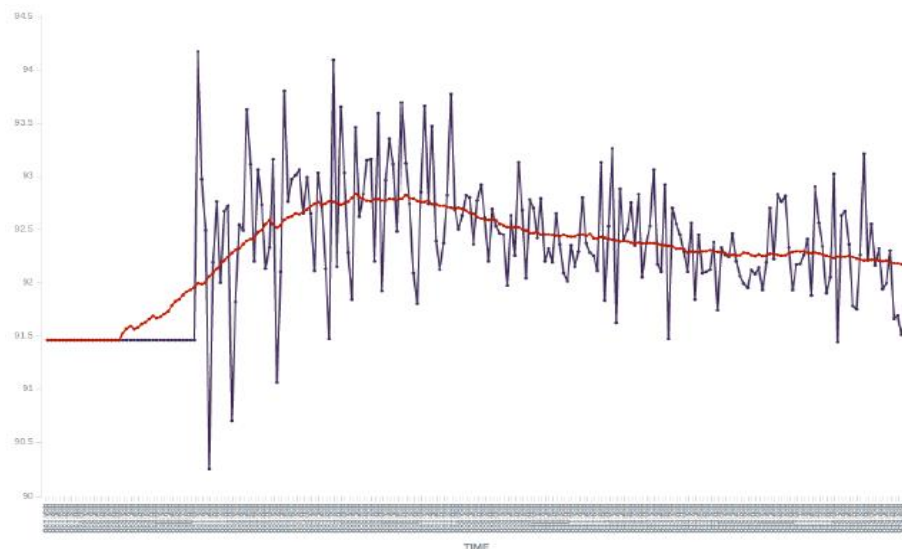


Рисунок 3.4 – Залежності напрямку і ковзного середнього для напрямку

Також варто враховувати, що похибки в вихідних даних можуть значно спотворити результат і на останньому етапі обробки даних – етапі обчислення характеристик для кожної ділянки [22]–[24].

При цьому, в даному випадку, можна говорити про те, що дані виміряні з високою точністю для подібних вимірювальних пристроїв, так як в загальному похибка у вимірюванні координат у систем геопозиціонування може перевищувати 40 м [22]–[24].

3.2.2 Корекція вихідних даних

Для підвищення точності системи було вирішено провести корекцію вихідних даних. В якості основи алгоритму корекції був узятий фільтр Калмана, так як він дозволяє окремо враховувати похибки вимірювань і похибки випадкового процесу, а також використовувати одержувану від датчика інформацію про точність вимірювань [22]–[24].

Інформацію про координати дорожнього полотна було вирішено не використовувати, так як в цілях приватності, в деяких випадках передаються зміщені координати транспортних засобів. Таким чином, використання

інформації про місцезнаходження дорожнього покриття може звужити область застосування розроблюваної системи [22]–[24].

3.2.3 Фільтр Калмана

Фільтр Калмана – це алгоритм обробки даних, що дозволяє прибирати шуми і зайву інформацію. Фільтр Калмана дозволяє задавати апіорну інформацію про характер системи, зв'язку змінних і на підставі цієї інформації будувати більш точну оцінку [25]–[27].

Фільтр Калмана використовує динамічну модель системи (наприклад, фізичний закон руху), відомі впливи управління та безліч послідовних вимірювань для формування оптимальної оцінки стану. Алгоритм складається з двох повторюваних етапів: екстраполяції та корекції. На першому етапі розраховується передбачення стану в наступний момент часу з урахуванням інформації про модель системи і неточності вимірювань. На другому – передбачене значення коригується за рахунок нової, отриманої з датчика інформації [25]–[27].

Для використання фільтра Калмана необхідно, щоб використовуваний процес описувався наступним чином за допомогою формул (3.1) та (3.2):

$$s_k = A_k s_{k-1} = B_k u_{k-1} + w_k, \quad (3.1)$$

$$z_k = H_k s_k + v_k, \quad (3.2)$$

де $s_k \in R^n$ – вектор стану процесу;

A_k – матриця розмірністю $n \times n$, що описує перехід спостережуваного процесу зі стану s_{k-1} в стан s_k ;

$u_k \in R^l$ – вектор керуючого впливу на процес;

B_k – матриця розмірністю $n \times 1$, що відображає вектор керуючих впливів u в зміна стану s ;

$w_k \in R^n$ – випадкова величина, що описує похибки досліджуваного процесу;

$z_k \in R^n$ – вектор вимірюваного стану процесу;

H_k – матриця розмірністю $m \times n$, що відображає стан процесу s в вимір процесу z ;

$v_k \in R^n$ – вектор, що описує похибки вимірювань.

Таким чином: на етапі екстраполяції за формулою (3.1) обчислюється апостеріорна нескоригована оцінка вектора стану s_k за апостеріорною скоригованою оцінкою вектора стану s_{k-1} . Також обчислюється нескорегована коваріаційна матриця помилок P_k за формулою (3.3) [25]–[27]:

$$P_k = A_k P_{k-1} A_k^T + Q_k, \quad (3.3)$$

де Q_k – коваріаційна матриця похибок процесу (3.4):

$$Q_k = \text{cov}(w_k). \quad (3.4)$$

На етапі корекції обчислюється матриця коефіцієнтів посилення K_k (3.5), а також скориговані оцінки стану s_k (3.6) і коваріаційна матриця P_k (3.7):

$$K_k = P_k H_k^T (H_k P_k H_k^T + R_k)^{-1}, \quad (3.5)$$

$$\hat{s}_k = \hat{s}_k + K_k (z_k - H_k \hat{s}_k), \quad (3.6)$$

$$P_k = (I - K_k H_k) P_k, \quad (3.7)$$

де R_k – коваріаційна матриця похибки вимірювань;

I – одинична матриця.

Матриця коефіцієнтів посилення K_k обчислюється таким чином, що підсумкові оцінки виходять оптимальними в сенсі мінімізації середньоквадратичної помилки [25]–[27].

3.2.4 Застосування фільтра Калмана для обробки геодалних

Так як досліджується процес руху об'єкта, то рівняння стану складається виходячи з рівняння руху тіла (3.8) [25]–[27]:

$$r_k = r_{k-1} + v_{k-1}(t_k - t_{k-1}) + a_{k-1} \frac{(t_k - t_{k-1})^2}{2}, \quad (3.8)$$

де r_k – вектор переміщення;

v_k – вектор швидкості;

a_k – вектор прискорення;

t_k – час.

За стан процесу приймемо вектор (3.9):

$$s_k = \begin{bmatrix} x_k \\ y_k \\ v_k^x \\ v_k^y \end{bmatrix}, \quad (3.9)$$

де x_k та y_k – координати об'єкта;

v_k^x та v_k^y – проекції швидкості об'єкта.

Так як у вихідних даних в загальному випадку відсутня додаткова інформація про процес руху (натискання на педаль, поворот керма), то вважається, що керуючий вплив дорівнює 0 [25]–[27].

Таким чином, для розглянутого процесу рівняння (3.1) прийме вигляд, що описується формулами (3.10)-(3.13):

$$s_k = A_k s_{k-1} + G_k a_k, \quad (3.10)$$

$$A_k = \begin{bmatrix} 1 & 0 & (t_k - t_{k-1}) & 0 \\ 0 & 1 & 0 & (t_k - t_{k-1}) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (3.11)$$

$$a_k = \begin{bmatrix} a_k^x \\ a_k^y \end{bmatrix}, \quad (3.12)$$

$$G_k = \begin{bmatrix} \frac{(t_k - t_{k-1})^2}{2} & 0 \\ 0 & \frac{(t_k - t_{k-1})^2}{2} \\ (t_k - t_{k-1}) & 0 \\ 0 & (t_k - t_{k-1}) \end{bmatrix}. \quad (3.13)$$

де a_k^x та a_k^y – проекції прискорення об'єкта.

У даній моделі приймається припущення про те, що більшу частину часу характер руху транспортного засобу є рівномірно-прямолінійним. Дане припущення прийнято з наступних міркувань: ненульове прискорення для транспортних засобів може бути пов'язано або зі зміною напрямку руху, або з зупинкою або початком руху. У цих випадках ненульове прискорення зберігається протягом нетривалого проміжку часу (5-15 с). Прискорення, що виникає при обгоні або перестроюванні, також відповідає цим характеристикам [25]–[27].

Таким чином, передбачається, що прискорення по осях x і y розподілені нормально з нульовим математичним очікуванням і деяким середньоквадратичним відхиленням σ_a (при обробці вихідних даних був проведений розрахунок цього відхилення, і було отримано, що в середньому

воно дорівнює 1 для кожної компоненти прискорення). Виходячи з цього, в даній моделі опису системи прискорення об'єкта розглядається як випадкова похибка процесу, і у формулі (10) член $G_k a_k$ виконує роль вектора w_k з формули (3.1) [25]–[27].

Для подальшого застосування фільтра Калмана необхідно обчислити матрицю Q . застосуємо скорочену формулу для обчислення матриці коваріації (3.14):

$$Q_k = \text{cov}(w_k) = E(w_k w_k^T) - E(w_k)E(w_k^T), \quad (3.14)$$

де $E()$ – вектор математичних очікувань випадкової величини.

Так як математичне очікування для прискорень дорівнює 0, то віднімається в даній формулі дорівнює 0. Тоді матриця Q може бути записана за допомогою формули (3.15):

$$Q_k = E(w_k w_k^T) = E(G_k a_k a_k^T G_k^T) = G_k E(a_k a_k^T) G_k^T. \quad (3.15)$$

Компоненти вектора прискорення є незалежними величинами. Дійсно, немає підстав вважати, що значення однієї компоненти прискорення залежить від значення іншої. Тоді отримаємо вираз (3.16):

$$E(a_k a_k^T) = \begin{bmatrix} \sigma_a^2 & 0 \\ 0 & \sigma_a^2 \end{bmatrix}, \quad (3.16)$$

Відповідно матриця Q може бути обчислена за формулою (3.17):

$$Q_k = G_k G_k^T \sigma_a^2. \quad (3.17)$$

У вихідних даних міститься інформація про географічні координати, швидкості і напрямку об'єкта. Виходячи зі значень швидкості і напрямку, розраховуються компоненти швидкості по осях x і y . На кожній ітерації алгоритму передбачається, що в попередній момент часу об'єкт знаходився в точці з координатами $(0, 0)$, при цьому запам'ятовуються реальні кутові координати об'єкта [25]–[27]. Виходячи зі значень компонент швидкості, розраховується нескорегована оцінка координат об'єкта в даний момент часу, а саме зміщення об'єкта з точки $(0, 0)$. Для порівняння цієї оцінки з вимірним значенням, за допомогою формул Вінченті для обчислення відстаней по кутових координатах, виміряні координати переводяться в метричні на підставі попередніх кутових координат. Після отримання скоригованої оцінки отримані метричні координати аналогічним чином переводяться в кутові [25]–[27].

Таким чином, матриця H_k , що відображає стан процесу s в вимір процесу z , приймається рівною одиничній матриці.

Тепер, для застосування фільтра, залишилося оцінити коваріаційну матрицю похибки вимірювань-матрицю R_k .

Найчастіше серед даних, що записуються системами геопозиціонування, присутнє поле Horizontal dilution of Precision (HDOP) — зниження точності в горизонтальній площині. Значення HDOP дозволяє параметрично описати геометричне взаєморозташування супутників щодо антени приймального пристрою. Коли супутники в області видимості знаходяться занадто близько один до одного, значення HDOP і похибка вимірювання будуть високими, і, навпаки, при достатній віддаленості значення HDOP і похибка будуть низькими [25]–[27].

Зазвичай для розрахунку похибки вимірювань системи геопозиціонування в метрах значення HDOP множиться на коефіцієнт User Equivalent Range Error (UERE) – коефіцієнт еквівалентної похибки вимірювання дальності. Значення UERE залежить від використовуваного

приймача системи геопозиціонування. Для більшості сучасних пристроїв $UERE = 5.1$ [25]–[27].

Немає підстав вважати, що похибка у вимірюванні компонент вектора стану залежні один від одного, тому можна використовувати формулу (3.18):

$$R_k = UERE \cdot HDOP_k \cdot I. \quad (3.18)$$

Таким чином, отримані формули дозволяють застосувати фільтр Калмана для корекції вихідних даних.

3.2.5 Реалізація алгоритму

У розробленій системі алгоритм був реалізований як окрема Java UDF.

Для твору матричних обчислень використовувалася бібліотека the Apache Commons Mathematics Library. Для перетворення кутових координат в метричні і назад використовувався пакет Geotools, заснований на формулах Вінченці [25]–[27].

Результат корекції вхідних даних представлений на рис. 3.1 та 3.2.

Чорною лінією позначена Початкова траєкторія, червоною-отримана в результаті корекції.

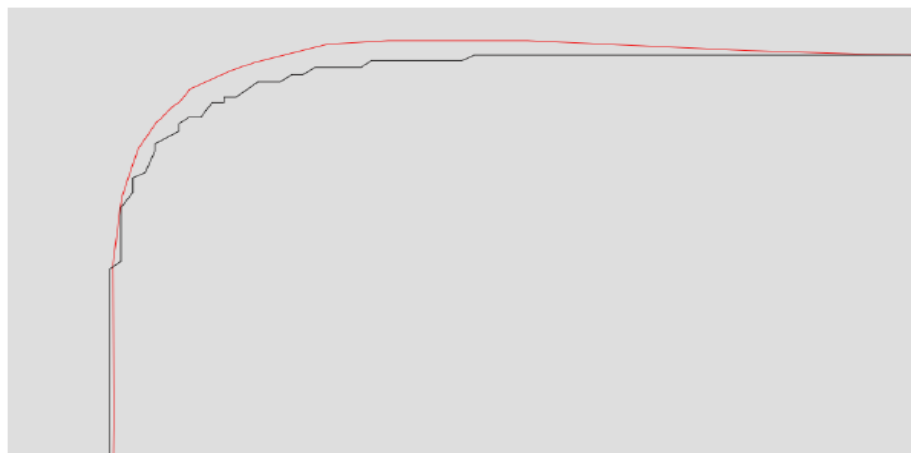


Рисунок 3.1 – Результат корекції вхідних даних

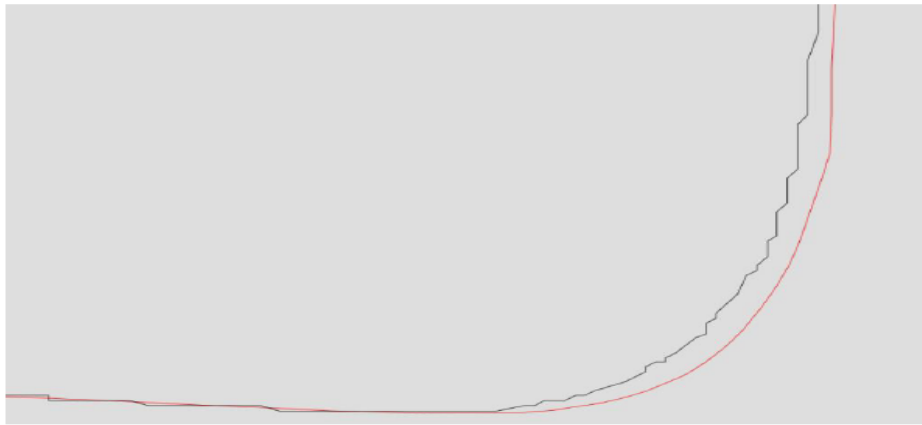


Рисунок 3.2 – Результат корекції вхідних даних

Наведені вище ілюстрації показують, що застосований алгоритм дозволяє знизити ступінь спотворення траєкторії руху об'єктів і отримати більш реалістичну модель їх переміщення.

3.2.6 Виявлення точок зміни напрямку

Для виявлення точок зміни напрямку використовувався алгоритм Рамера-Дугласа-Пекера (RDP). Цей алгоритм дозволяє зменшити число точок кривої, апроксимованої більшою серією точок. Ідея алгоритму полягає в тому, щоб по даній ламаній, апроксимируючій криву, побудувати ламану з меншим числом точок [25]–[27]. При цьому спрощена крива буде складатися з підмножини точок, які визначаються з вихідної кривої (рис. 3.3).

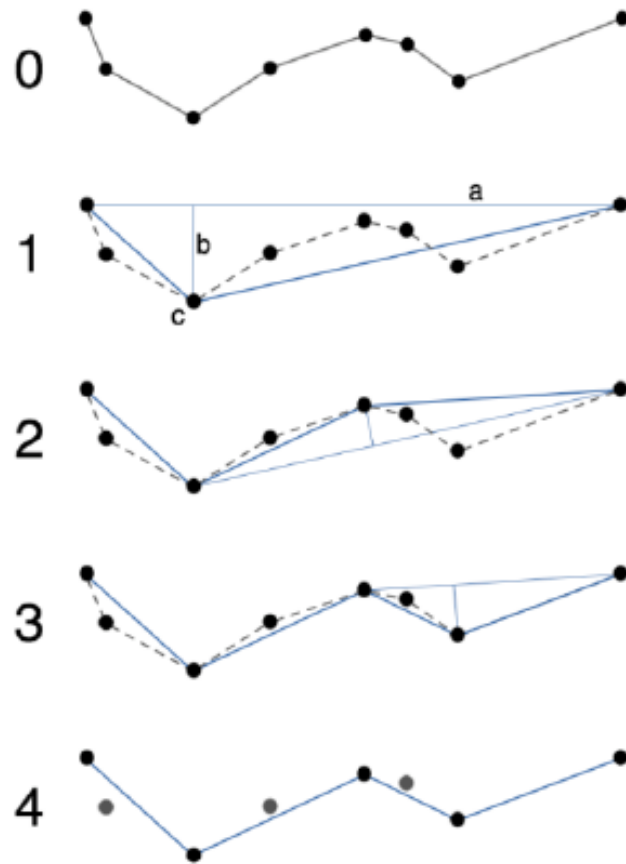


Рисунок 3.3 – Наближення кривої за допомогою алгоритму RDP

На вхід алгоритм приймає початкову криву (крива 0 на рис. 3.3), що представляє собою впорядкований набір точок або ліній, і відстань ϵ , що визначає максимальну допустиму розбіжність між вихідною і апроксимованою кривою. Перша і остання точки зберігаються незмінними. Потім знаходиться точка, найбільш віддалена від відрізка, що з'єднує першу і останню [25]–[27]. Якщо ця точка видалена на відстань, менше ϵ , то всі точки, які ще не були збережені, відкидаються і вийшла пряма згладжує криву з точністю не нижче ϵ . Якщо ж відстань більше ϵ , то алгоритм рекурсивно викликає себе на наборі від початкової до даної і від даної до кінцевої точок (при цьому, дана точка зберігається). По закінченню всіх рекурсивних викликів будувється вихідна Ламана тільки зі збережених точок (крива 4 на рис. 3.3) [25]–[27].

У розроблюваній системі отримані на виході алгоритму RDP точки є точками зміни напрямку руху.

Існують стандартні реалізації алгоритму RDP на Java. Однак всі вони припускають, що є координати траєкторії в прямокутній системі координат. Вихідні дані містять тільки кутові координати, отже їх пряме використання неможливо [25]–[27].

Для застосування алгоритму RDP було вирішено перетворювати кутові координати в проекційні UTM координати.

Система координат UTM (Universal Transverse Mercator) розділяє Землю на 60 витягнутих в меридіональному напрямку зон шириною 6 градусів і відображає їх окремо в рівнокутній поперечно-циліндричній проекції Меркатора (рис. 3.4).

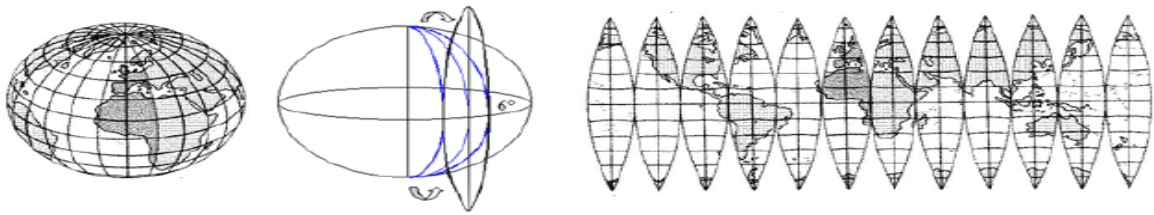


Рисунок 3.4 – Відображення кутових координат в проекційні

Таким чином, була реалізована UDF для Apache Pig, що приймає на вхід вихідні траєкторії і видає опорні точки для цих траєкторій.

3.3 Виявлення точок вимірювання прискорення

Незважаючи на те, що вихідні дані були скориговані, застосування розрахованого за швидкостями двох сусідніх точок прискорення для виявлення зміни прискорення було неможливим через сильні похибки. Тому в цих цілях було вирішено аналізувати плаваюче середнє для розрахованих прискорень [25]–[27].

На рис. 3.5 фіолетовим кольором показана залежність швидкості від часу, зеленим – розрахованого за двома сусіднім точкам прискорення від часу, червоним – плаваючого середнього для прискорення на інтервалі в 4 с.

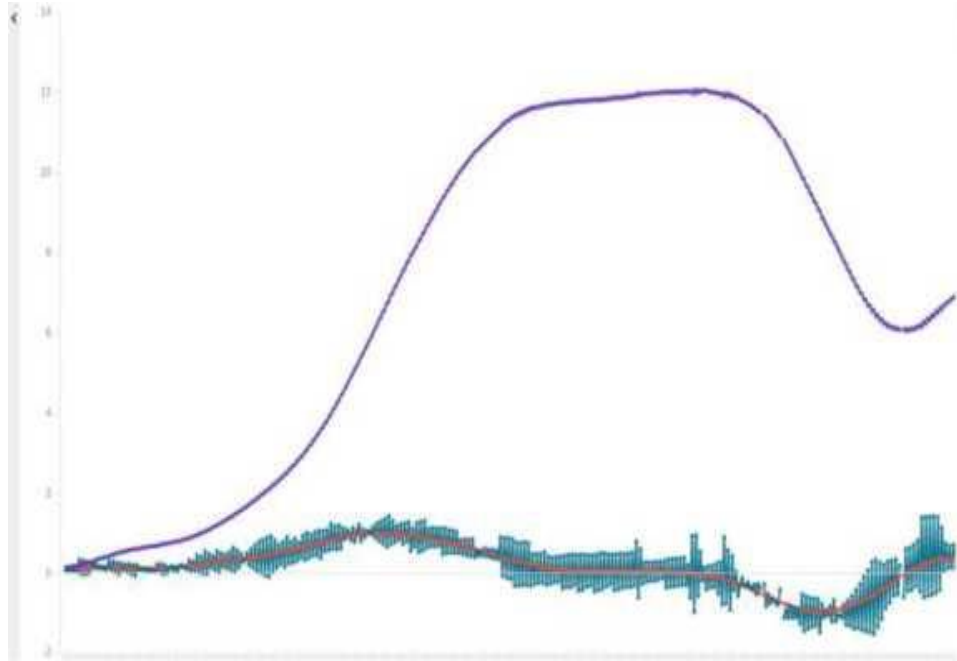


Рисунок 3.5 – Залежність швидкості від часу

Для визначення характеру руху в даній точці траєкторії було введено порогове значення для прискорення. Якщо абсолютне значення плаваючого прискорення менше цього порогового значення, то вважається, що об'єкт рухається рівномірно, якщо більше – то прискорюється (при позитивному значенні) або гальмує (при негативному) [25]–[27].

3.4 Розбиття траєкторії на ділянки і обчислення характеристик руху

Таким чином, опорними точками для розбиття траєкторії руху на ділянки в розроблюваній системі були: точки, отримані за допомогою алгоритму RDP, і точки, в яких змінюється характер прискорення об'єкта. Для кожного отриманого ділянки обчислювалися [25]–[27]:

– середнє значення швидкості. Якщо обчислене значення було менше деякого близького до нуля порогового значення, то вважалося, що на даному інтервалі часу об'єкт не переміщався. Інакше дана ділянка характеризувався як ділянка прискорення, гальмування або рівномірної їзди, в залежності від значення плаваючого прискорення, розрахованого раніше;

– середнє значення кутової швидкості об'єкта. Якщо це значення перевищувало деяке порогове значення, то дана ділянка характеризувався як ділянка повороту, інакше як ділянка прямолінійного руху.

Після цього, для отриманих ділянок руху збиралися наступні характеристики:

- інтервал часу;
- характер зміни напрямку: поворот, прямолінійний рух або відсутність руху;
- характер зміни прискорення: прискорення, гальмування, рівномірний рух або відсутність руху;
- середня швидкість руху на даній ділянці;
- абсолютна величина середнього прискорення;
- середня величина доцентрового прискорення.

З отриманих даних, за допомогою найпростіших Apache Pig запитів, можна отримувати різні характеристики водіння (наприклад, середні величини прискорення, гальмування, швидкості, доцентрового прискорення в поворотах) і поїздки в цілому (наприклад, кількість і частота зупинок [25]–[27], тривалість поїздки, пройденої відстань, час рівномірного руху) необхідні для подальшого аналізу.

3.5 Висновки за розділом 3

Розроблено програмне забезпечення для обробки великих даних, отриманих з транспортних засобів. Запропоновано структуру та описано основні етапи роботи внутрішніх алгоритмів.

Описано функціонування програмного забезпечення для обробки великих даних, отриманих з транспортних засобів. Описано особливості реалізації ПЗ для обробки великих даних, отриманих з транспортних засобів.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Була розроблена система, що дозволяє виділяти характеристики водіння і поїздок на підставі даних, одержуваних з систем геопозиціонування. Розроблена система задовольняє всім поставленим вимогам, а саме:

- система відповідає принципу горизонтального масштабування;
- система вміє працювати з різними схемами даних у вихідних файлах;
- система вміє працювати незалежно від похибок у вихідних даних.

Для цих цілей був застосований алгоритм корекції даних, заснований на фільтрі Калмана. Коректність роботи алгоритму була перевірена за допомогою накладення отриманих траєкторій на географічну карту;

- система розбиває поїздки на ділянки з однаковим характером руху і обчислює динамічні характеристики для отриманих ділянок.

4.2 Проведення тестування. Аналіз отриманих результатів

Як приклад використання отриманих даних було вирішено провести кластеризацію водіїв за такими ознаками:

- середнє значення прискорення при прискоренні;
- середнє значення прискорення при гальмуванні;
- середнє значення доцентрового прискорення при проходженні поворотів.

Кластеризація проводилася за допомогою алгоритму k-means (метод k-середніх) [28]. Цей метод розбиває безліч вихідних об'єктів на заздалегідь відоме число кластерів k таким чином, щоб мінімізувати середньоквадратичне відхилення об'єктів від центрів відповідних кластерів. На кожній ітерації перевищується центр мас для кожного кластера, отриманого на попередньому кроці, потім об'єкти розбиваються на кластери знову відповідно

до нового положення кластерів (рис. 4.1). Алгоритм завершується, як тільки на поточній ітерації не відбувається зміни кластерів [28].

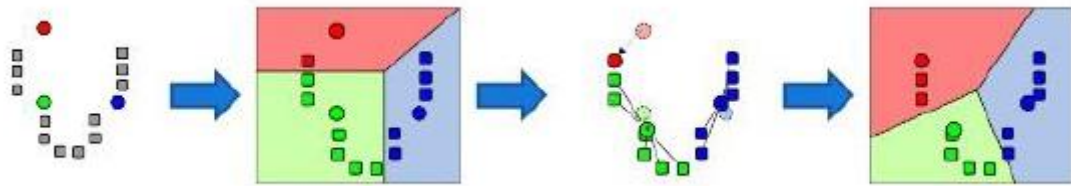


Рисунок 4.1 – Алгоритм k-means

Аналізувалися дані, зібрані за місяць, що містять інформацію про 8967 поїздках 65 водіїв. Розмір вихідних даних становив понад 27 Гб, кількість записів – понад 80 млн [29].

Так як розкид значень доцентрового прискорення був значно вище, ніж у гальмування і прискорення, отримані дані були попередньо нормалізовані.

Для визначення оптимальної кількості кластерів була побудована залежність сумарної відстані об'єктів до центрів кластерів від кількості кластерів (рис. 4.2).

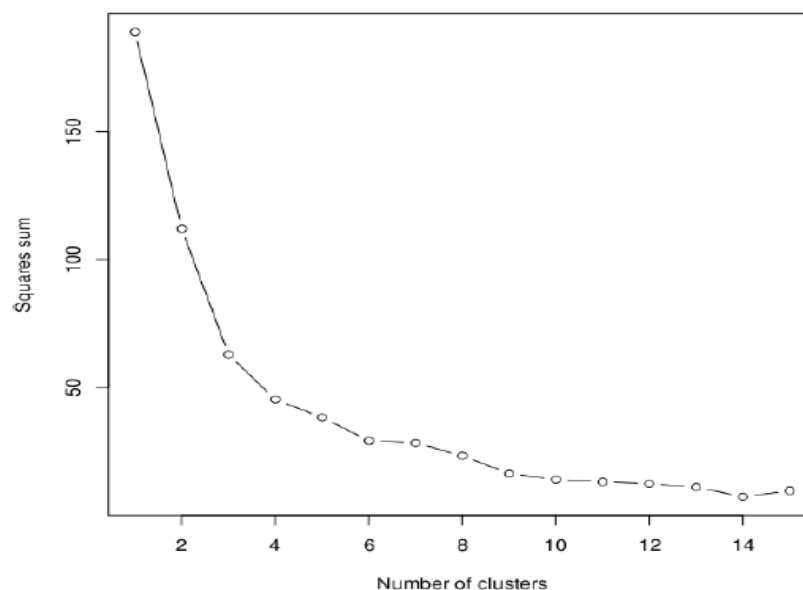


Рисунок 4.2 – Залежності сумарної відстані об'єктів до центрів кластерів від кількості кластерів

В даному випадку оптимальним буде розбиття об'єктів на шість кластерів, так як подальше збільшення їх числа не веде до значного зменшення суми відстаней.

Таким чином було отримано шість кластерів (табл. 4.1, рис. 4.3).

Таблиця 4.1 – Інформація про отримані кластери

№ кластера	Доцентрове прискорення	Прискорення	Гальмування	Кількість водіїв
1	-0.2063358	-1.2018407	-1.1488741	13
2	-0.1827992	1.4781924	1.2344404	24
3	-0.2179239	0.4181136	0.4666877	1
4	-0.6739149	-3.4083534	-3.7190770	14
5	3.9781538	0.2282023	0.6838466	9
6	-0.1151139	-0.2914570	-0.3506687	3

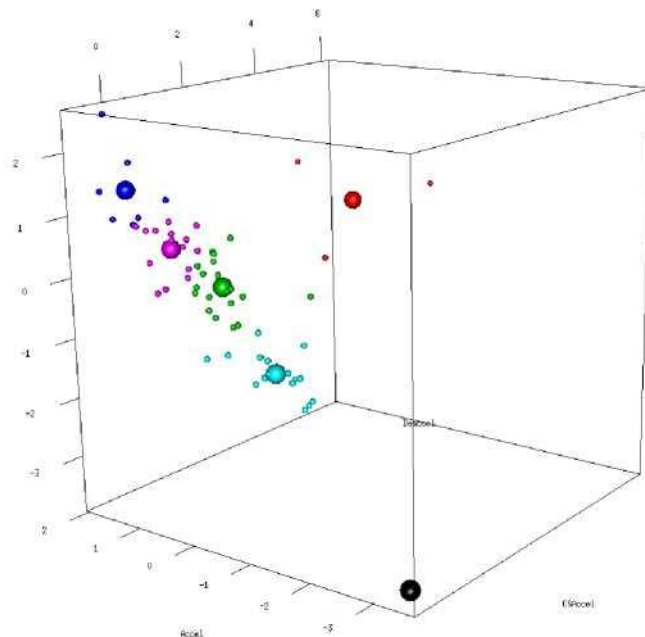


Рисунок 4.3 – Графічне представлення отриманих кластерів

Отримана картина дозволяє виділити шість груп водіїв. У шостій групі (червоний колір) зосереджені водії з найбільш агресивною манерою водіння, так як здійснюються ними маневри найбільш різкі. Так само варто відзначити третій кластер, що складається з одного водія з найменш різкою манерою

водіння. Швидше за все, це водій автобуса або вантажівки, однак, для підтвердження цієї гіпотези слід провести аналіз додаткових характеристик (середня кількість зупинок, середня тривалість поїздок і т.д.).

Також був проведений кластерний аналіз поїздок (рис. 4.4).

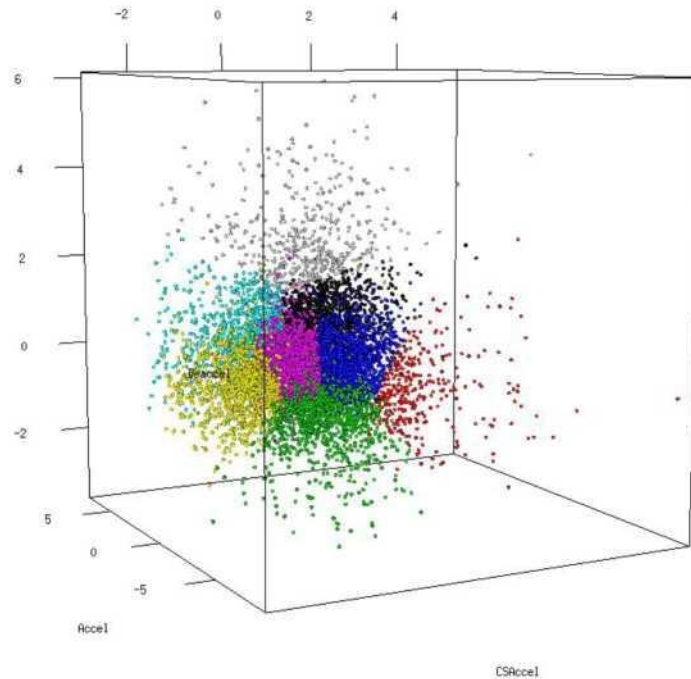


Рисунок 4.4 – Графічне представлення кластерного аналізу поїздок

В цілому, тут можна судити про схожу картину, при цьому більший розкид даних вийшов внаслідок того, що характеристики, отримані від однієї поїздки, можуть сильно залежати від різних умов (погоди, трафіку, часу доби і т.д.).

4.3 Висновки за розділом 4

В ході виконання роботи були вивчені основні існуючі підходи в обробці великих обсягів даних.

Була розроблена система, що дозволяє виділяти характеристики водіння і поїздок на підставі даних, одержуваних з систем геопозиціонування. Розроблена система задовольняє всім поставленим вимогам, а саме:

- система відповідає принципу горизонтального масштабування;
- система вміє працювати з різними схемами даних у вихідних файлах;
- система вміє працювати незалежно від похибок у вихідних даних.

Для цих цілей був застосований алгоритм корекції даних, заснований на фільтрі Калмана. Коректність роботи алгоритму була перевірена за допомогою накладення отриманих траєкторій на географічну карту;

- система розбиває поїздки на ділянки з однаковим характером руху і обчислює динамічні характеристики для отриманих ділянок.

За допомогою системи були проаналізовані дані про 8967 поїздках 65 водіїв, що містилися у файлі з понад 80 млн. записів.

Як приклад використання отриманих при роботі системи даних, за допомогою алгоритму k-means була проведена кластеризація водіїв на групи за манерою їх водіння.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи магістра було проведено аналіз і порівняння існуючих рішень для обробки великих даних, отриманих з транспортних засобів.

На основі результатів дослідження, було прийнято рішення про програмну реалізацію нового рішення для обробки великих даних, отриманих з транспортних засобів на основі фреймворку Hadoop.

Розроблена система задовольняє всім поставленим вимогам, а саме:

- система відповідає принципу горизонтального масштабування;
- система вміє працювати з різними схемами даних у вихідних файлах;
- система вміє працювати незалежно від похибок у вихідних даних.

Для цих цілей був застосований алгоритм корекції даних, заснований на фільтрі Калмана. Коректність роботи алгоритму була перевірена за допомогою накладення отриманих траєкторій на географічну карту;

– система розбиває поїздки на ділянки з однаковим характером руху і обчислює динамічні характеристики для отриманих ділянок.

Проведені експерименти підтвердили продуктивність та ефективність розробленого ПЗ на обчислювальних системах.

У якості майбутніх варіантів розвитку розробки було запропоновано реалізацію розробленого методу на паралельній обчислювальній системі.

Новизна роботи полягає в тому, що запропоновано метод, що може працювати незалежно від похибок у вихідних даних. Для цих цілей був застосований алгоритм корекції даних, заснований на фільтрі Калмана.

Практичне значення роботи полягає в тому, що розроблено програмне забезпечення для реалізації обробки великих даних, отриманих з транспортних засобів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Дзямулич М. І. Великі дані та їх роль у формуванні цифрової економіки / М. І. Дзямулич, Т. О. Шматковська, О. В. Борисюк // Галицький економічний вісник. - 2021. - № 3. - С. 16-21.
2. Піжук О. І. Великі дані як основоположний драйвер цифрової трансформації економіки / О. І. Піжук // Економіка та держава. - 2019. - № 6. - С. 50-54.
3. Зибарева О. В. Актуалізація концепції "великі дані" (англ. "Big Data") в умовах поширення інформаційного суспільства / О. В. Зибарева, І. П. Кравчук. // Економіка. Управління. Інновації. Серія : Економічні науки. - 2015. - № 1.
4. Клопов І. О. Інтернет речей та BIG DATA: аналітика в режимі реального часу [Електронний ресурс] / І. О. Клопов, О. О. Шапуров // Науковий погляд: економіка та управління. - 2022. - № 2. - С. 156-163.
5. Скляренко О. В. Використання Big data: сучасні можливості для бізнесу / О. В. Скляренко, О. І. Федік, Я. О. Колодінська // Економіка і управління. - 2020. - № 3. - С. 106-110.
6. Шандрівська О. Є. Особливості ідентифікації ризиків ринку Big Data / О. Є. Шандрівська, А. А. Кириленко // Management and entrepreneurship in Ukraine: the stages of formation and problems of development. - 2021. - Vol. 3, numb. 1. - С. 82-95.
7. Conrad S. M. Mueller A New Computational Model for Real Gains in Big Data Processing Power / S. M. Mueller Conrad // Advances in cyber-physical systems. - 2017. - Vol. 2, Num. 1. - С. 11-21.
8. Tikhonov V. Modeling the conveyor-modular transfer of multimedia data in a sensor network of transport system / V. Tikhonov, O. Tykhonova, O. Tsyra, O. Yavorska, A. Taher, O. Kolyada, S. Kotova, O. Semenchenko, E. Shapenko // Восточно-Европейский журнал передовых технологий. - 2019. - № 2(2). - С. 6-14.

9. Danshyna S. Yu. Information technology of transport infrastructure monitoring based on remote sensing data / S. Yu. Danshyna, A. S. Nechausov, S. M. Andrieiev // Радіоелектроніка, інформатика, управління. - 2022. - № 4. - С. 86-97.

10. Klius Yu. I. Evaluation and control of data relevance in information systems of the transport industry management / Yu. I. Klius, M. A. Prodanchuk, O. O. Gololobova, L. V. Halan, M. O. Varhatiuk // Naukovyi visnyk Natsionalnoho Hirnychoho Universytetu. - 2023. - № 3. - С. 137-143.

11. Robi VTS [Electronical resource]. – Access mode : <https://vts.robi.com.bd/>

12. Robi VTS – Google Play [Electronical resource]. – Access mode : <https://play.google.com/store/apps/details?id=com.singularitybd.robi.robitrackervts&hl=uk&gl=US>

13. ACKO. KACKO KACKO4U [Electronical resource]. – Access mode : <https://aska.ua/ru/news/kacko4u-innovaciya-v-ukrainskom-strahovanii-ot-aska>

14. ACKO. KACKO KACKO4U – Google Play [Electronical resource]. – URL: <https://play.google.com/store/apps/details?id=com.kacko4u.ua&hl=ru&gl=US>

15. Waze [Electronical resource]. – URL: <https://www.waze.com/uk/live-map>

16. Waze – Google Play [Electronical resource]. – Access mode : <https://play.google.com/store/apps/details?id=com.waze&hl=uk&gl=US>

17. Bilyk V. Propagator-Oriented Programming Model Using Java / V. Bilyk, A. Sachenko // Advances in cyber-physical systems. - 2023. - Vol. 8, Num. 1. - С. 9-16.

18. Максимчук О. О. Особливості застосування Hadoop / О. О. Максимчук // Вісник університету "Україна". Серія : Інформатика, обчислювальна техніка та кібернетика. - 2016. - № 1. - С. 189-195.

19. Мельник В. М. Використання звичайних сокетів API для файлових систем типу Hadoop / В. М. Мельник // Комп'ютерно-інтегровані технології: освіта, наука, виробництво. - 2015. - № 18. - С. 40-49.

20. Eclipse.org [Electronical resource]. – Access mode :
<https://www.eclipse.org/>.

21. Apache Pig [Electronical resource]. – Access mode :
<http://pig.apache.org/docs/r0.15.0/>.

22. Cascading [Electronical resource]. – Access mode :
<http://www.cascading.org/documentation/>.

23. Apache Storm [Electronical resource]. – Access mode :
<http://storm.apache.org/releases/0.9.6/index.html>.

24. UTM Projection [Electronical resource]. – Access mode :
<http://geokov.com/education/utm.aspx>.

25. Лімонов О. С. Аналіз фільтра Калмана стосовно до відновлення ізогіпс при побудові карт цифрового рельєфу / О. С. Лімонов, Б. В. Перелигін, Т. М. Пустовіт // Український гідрометеорологічний журнал. - 2012. - № 11. - С. 58-67.

26. Скарга-Бандурова І. С. Дослідження ефективності використання фільтру Калмана для прогнозування часу прибуття міського транспорту / І. С. Скарга-Бандурова, М. В. Деркач // Вісник Херсонського національного технічного університету . - 2017. - № 4. - С. 193-199.

27. Форкун І. В. Система автоматичного керування безпілотним літальним апаратом на основі фільтра калмана / І. В. Форкун, Д. М. Медзатий, Д. А. Макаришкін, О. В. Червонецький // Вісник Хмельницького національного університету. Технічні науки. - 2020. - № 5. - С. 57-64.

28. Худов Г. В. Метод сегментування зображень в системі дистанційного відеомоніторингу на основі алгоритму k-means / Г. В. Худов, Д. Л. Місюк // Збірник наукових праць Державного науково-дослідного інституту випробувань і сертифікації озброєння та військової техніки. - 2023. - Вип. 2. - С. 82-89.

29. Data Set [Electronical resource]. – Access mode :
<https://dsec.ifi.uzh.ch/dsec-datasets/download/>

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для обробки великих даних, отриманих з транспортних засобів.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Дослідження та програмна реалізація методів обробки великих даних, отриманих з транспортних засобів», затверджене наказом Національного університету «Запорізька політехніка» № 437 від 14 листопада 2023 р.

A.2 Призначення розробки

Програмний продукт призначений для обробки великих даних, отриманих з транспортних засобів.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

При розробці ПЗ обробки великих даних, отриманих з транспортних засобів, необхідно забезпечити такі функціональні вимоги:

- система повинна відповідати принципу горизонтальної масштабованості, так як передбачається обробка великих обсягів даних, система повинна бути легко розширювана;
- на вхід система повинна приймати CSV файли. При цьому система повинна вміти працювати з різними схемами даних в цих файлах, а єдиною умовою є наявність обов'язкових полів, описаних вище. Схема даних повинна передаватися в систему при конфігуруванні;

- система повинна вміти обробляти дані незалежно від похибки вимірювань;
- система повинна розбивати поїздки на ділянки руху з однаковим характером зміни напрямку і швидкості;
- за характером зміни напрямку передбачається розбивати ділянки руху на поворот та прямолінійний рух;
- за характером зміни швидкості передбачається розбивати ділянки поїздок на ділянки гальмування, ділянки прискорення, ділянки рівномірного руху, ділянки з нульовою швидкістю;
- для виділених ділянок руху система повинна розраховувати інтервал часу, середню швидкість руху, абсолютну величину середнього прискорення, середню величину доцентрового прискорення;
- отриманий результат система повинна записувати в CSV файл;
- система повинна мати можливість запису попередніх результатів роботи для аналізу їх адекватності.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для обробки великих даних, отриманих з транспортних засобів.

А.3.3 Вимоги до надійності

Програмне забезпечення для обробки великих даних, отриманих з транспортних засобів.

A.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для обробки великих даних, отриманих з транспортних засобів наявність ПК у користувача.

A.3.5 Вимоги до складу та параметрів технічний засобів

Для експлуатації програмного забезпечення для локалізації облич на зображеннях необхідно таке апаратне та програмне забезпечення:

- процесор із тактовою частотою 2,7 GHz і більше;
- вільне місце на диску 1Гб (та 28 Гб для розміщення вибірки даних) або більше;
- оперативна пам'ять об'ємом 8Гб і більше.

A.3.6 Вимоги до маркування і пакування

Програма для обробки великих даних, отриманих з транспортних засобів може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для обробки великих даних, отриманих з транспортних засобів».

ДОДАТОК Б
Фрагмент тексту програми

```

import java.io.File;
import java.io.IOException;
import java.nio.Bytebuff;
import java.nio.file.fls;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Stream;

import com.Hadoop.Pig.core.component.ComponentException;
import com.Hadoop.Pig.core.component.RunningState;
import com.Hadoop.Pig.core.validation.ValidationException;
import com.Hadoop.Pig.Datstore.Direct.DirectDatStore;
import com.Hadoop.Pig.framework.il8n.BundleLogger;
import com.Hadoop.Pig.framework.il8n.BundleLoggerFactory;
import com.Hadoop.Pig.mngr.Datstore.Direct.DirectDatStoreMngr;
import com.Hadoop.Pig.Trst.InbdTrstBase;
import com.Hadoop.Pig.Trst.Trstdefntn;

public class BigFileInbdTrst extends InbdTrstBase implements
Runnable
{
    private static final BundleLogger  LOGGER  =
BundleLoggerFactory.getLogger(BigFileInbdTrst.class);

    private String errorMessage;

    private File      Direct;
    private boolean    recursive;
    private boolean    deletefls;
    private String FlsNmFilter;

    private volatile boolean running
    = false;
    private volatile boolean stopping          =
false;
    private Thread          thread;
    private Bytebuff        buff
    = Bytebuff.allocate(1024 * 1024 * 10); // 1024 * 1024 *
10 = 10 MB
    private int
numberOfLinesToRead = 1000;
    private long
flushIntervalInMS    = 500;
    private boolean          ignoreFirstLine
    = false;

    private File            currFile;
    private List<String>     processedfls  =
new
ArrayList<String>();

```

```

        private List<String> pendingfls = new
ArrayList<String>();

        private DirectDatStoreMgr DirectDatStoreMgr;

        public BigFileInbdTrst(Trstdefntn defntn, DirectDatStoreMgr
mgr) throws ComponentException
        {
            super(defntn);
            this.defntn = defntn;
            this.DirectDatStoreMgr = mgr;
        }

        protected void applyProperties()
        {
            String DirectDatStoreNm =
getProp(BigFileInbdTrstServ.INPUT_DirectDatSTORE_Prop).getValAsSt
ring();
            if (DirectDatStoreNm == null)
            {
                throw new
RuntimeException(LOGGER.translate("PROP_IS_REQUIRED",
BigFileInbdTrstServ.INPUT_DirectDatSTORE_Prop));
            }
            DirectDatStore DirectDatStore =
DirectDatStoreMgr.getDirectDatStore(DirectDatStoreNm);
            if (DirectDatStore == null)
            {
                throw new
RuntimeException(LOGGER.translate("INVALID_Direct_DS",
DirectDatStoreNm));
            }

            String DirectNm = (String)
getProp(BigFileInbdTrstServ.INPUT_Direct_Prop).getVal();
            FlsNmFilter = (String)
getProp(BigFileInbdTrstServ.INPUT_FILE_FILTER_Prop).getVal();
            recursive = ((Boolean)
getProp(BigFileInbdTrstServ.INPUT_RECURSIVELY_SCAN_Prop).getVal()
).booleanVal();
            deletefls = ((Boolean)
getProp(BigFileInbdTrstServ.INPUT_DELETE_PROCESSED_fls_Prop).getV
al()).booleanVal();
            if
(hasProp(BigFileInbdTrstServ.INPUT_MAX_NUM_OF_LINES_Prop))
            {
                numberOfLinesToRead = (Integer)
getProp(BigFileInbdTrstServ.INPUT_MAX_NUM_OF_LINES_Prop).getVal()
;
            }
            if
(hasProp(BigFileInbdTrstServ.INPUT_FLUSH_INTERVAL_MS_Prop))
            {

```

```

        flushIntervalInMS = ((Integer)
getProp(BigFileInbdTrstServ.INPUT_FLUSH_INTERVAL_MS_Prop).getVal(
)).longVal();
    }
    if
    (hasProp(BigFileInbdTrstServ.INPUT_IGNORE_FIRST_LINE_Prop))
    {
        ignoreFirstLine = ((Boolean)
getProp(BigFileInbdTrstServ.INPUT_IGNORE_FIRST_LINE_Prop).getVal(
)).booleanVal();
    }
    Direct = new File(DirectDatStore.getPath(), DirectNm);
}

private void populatePendingfls(File currDirect)
{
    String[] fls = currDirect.list();
    if (fls == null)
        return;
    for (String file : fls)
    {
        File newFile = new
File(currDirect.getAbsolutePath() + File.separator + file);
        if (newFile.isDirectory())
        {
            if (recursive)
                populatePendingfls(newFile);
        }
        else
        {
            String newFlsNm = newFile.getNm();
            if (FlsNmFilter.length() == 0 ||
newFlsNm.matches(FlsNmFilter))
            {
                String newFileAbsolutePath =
newFile.getAbsolutePath().intern();
                pendingfls.add(newFileAbsolutePath);
            }
        }
    }
}

protected void readbuff()
{
    if (currFile == null)
    {
        currFile = getNextFile();
        if (currFile == null)
        {
            try
            {
                Thread.sleep(100);
            }

```

```

        catch (InterruptedException ex)
        {
            return;
        }
        return;
    }
}

// read the fls by line
long strtTime = 0L;
try
    (Stream<String> lines =
fls.lines(Paths.get(currFile.getPath())))
{
    LOGGER.info("strt_READING_FILE", currFile.getNm());
    strtTime = System.currTimeDist();
    int countI = 0;
    for (String line : (Iterable<String>)
lines::iterator)
    {
        if(!running || stopping)
            break;

        // do we ignore the first line ?
        if ( ! (ignoreFirstLine && countI == 0))
        {
            buff.put(line.getBytes());

            buff.put(System.lineSeparator().getBytes());
        }
        countI++;
        // process each nth line
        if (buff.hasRem() && countI %
numberOfLinesToRead == 0)
        {
            buff.flip();
            byteListener.receive(buff,
currFile.getNm());

            buff.compact();
            LOGGER.info("DONE_READING_LINES", countI,
currFile.getNm());

            // allow the adapter to breath
            try
            {
                Thread.sleep(flushIntervalInMS);
            }
            catch (InterruptedException ignored)
            {
            }
        }
    }

    if(!running || stopping)
    {

```

```

        buff.clear();
        currFile = null;
        return;
    }
    // finish off the reading of the file
    buff.flip();
    if (buff.hasRem())
    {
        byteListener.receive(buff, currFile.getNm());
    }
    buff.clear();
    if (deletefls)
    {
        processedfls.remove(currFile);
        fls.delete(currFile.toPath());
    }
    long endTime = System.currTimeDist() - strtTime;
    LOGGER.info("DONE_READING_FILE", countI,
currFile.getNm(), endTime);
    currFile = null;
    }
    catch (IOException ex)
    {
        LOGGER.error("ERROR_READING_FILE",
currFile.getNm(), ex.getMessage());
        LOGGER.info(ex.getMessage(), ex);
        currFile = null;
        return;
    }
}

private File getNextFile()
{
    pendingfls.clear();
    populatePendingfls(Direct);

    for (String file : pendingfls)
    {
        if (!processedfls.contains(file))
        {
            File f = new File(file);
            processedfls.add(f.getAbsolutePath());
            return f;
        }
    }
    return null;
}

protected void cleanup()
{
    pendingfls.clear();
    processedfls.clear();
    currFile = null;
}

```

```

    }

    @Override
    public synchronized void strt()
    {
        if (running || stopping)
            return;
        running = true;
        setRunningState(RunningState.strtING);
        thread = new Thread(this);
        thread.strt();
    }

    @Override
    public synchronized void stop()
    {
        if (!running || stopping)
            return;
        stopping = true;
        running = false;
        setRunningState(RunningState.STOPPING);
    }

    private synchronized void reset()
    {
        thread = null;
        stopping = false;
        running = false;
        if (errorMessage != null)
            setRunningState(RunningState.ERROR);
        else
            setRunningState(RunningState.STOPPED);
    }

    @Override
    public synchronized boolean isRunning()
    {
        return running;
    }

    public String getErrorMessage()
    {
        return errorMessage;
    }

    @Override
    public void validate() throws ValidationException
    {
        List<String> errors = new ArrayList<String>();
        try
        {
            super.validate();

```

```

    }
    catch (ValidationException e)
    {
        errors.add(e.getMessage());
    }

    String ds =
    getProp("inputDirectDatStore").getValAsString();
    if (DirectDatStoreMngr.getDirectDatStore(ds) == null)
    {

        errors.add(LOGGER.translate("Direct_DS_NOT_REGISTERED"));
    }

    if (errors.size() > 0)
    {
        StringBuffer sb = new StringBuffer();
        for (String message : errors)
            sb.append(message).append("\n");

        throw new
        ValidationException(LOGGER.translate("VALIDATION_ERROR",
        sb.toString()));
    }

    @Override
    public void run()
    {
        try
        {
            errorMessage = null;
            applyProperties();
            setRunningState(RunningState.strtED);
            while (running && !stopping)
            {
                try
                {
                    readbuff();
                }
                catch (Exception e)
                {
                    errorMessage =
                    LOGGER.translate("READ_buff_ERROR", e.getMessage());
                    LOGGER.error(errorMessage);
                    LOGGER.info(e.getMessage(), e);
                    stop();
                }
            }
            cleanup();
        }
        catch (Exception ex)
        {

```

```
        errorMessage = LOGGER.translate("READ_buff_ERROR",
ex.getMessage());
        LOGGER.error(errorMessage);
        LOGGER.info(ex.getMessage(), ex);
    }
    reset();
}

@Override
public boolean isClusterable()
{
    return false;
}
```

ДОДАТОК В
Слайди презентації

Міністерство освіти та науки України
 Національний університет «Запорізька політехніка»
 Кафедра програмних засобів

Дослідження та програмна реалізація методів обробки великих даних, отриманих з транспортних засобів

Керівник	д.т.н., професор Сергій СУББО ПН
Виконав	студ. груп. КНТ-112м Олександр ЧУЛЮК

Рисунок В.1 – Слайд 1

Об'єкт, предмет та мета роботи

- **Об'єкт дослідження** – процес розробки програмного забезпечення для обробки великих даних, отриманих з транспортних засобів.
- **Предмет дослідження** – програмні засоби для обробки великих даних, отриманих з транспортних засобів.
- **Метою роботи є** дослідження та розробка програмного забезпечення для обробки великих даних, отриманих з транспортних засобів.

Рисунок В.2 – Слайд 2

Аналіз програмного забезпечення для обробки великих даних, отриманих з транспортних засобів

Критерій порівняння	Robi VTS	КАСКО4U	Waze
Інтеграція великих даних	Немає	Немає	Є
Інтелектуальний аналіз накопичених даних	Формування аналітичних звітів	Відсутній	Робота з маршрутами онлайн
Зручність персоналізації	Середній рівень	Середній рівень	Середній рівень
Розширюваність	Через платну підписку	Відсутня	Відсутня
Кросплатформність	Тільки мобільні ОС	Тільки мобільні ОС	Підтримується

3

Рисунок В.3 – Слайд 3

Завдання роботи

При розробці ПЗ обробки великих даних, отриманих з транспортних засобів необхідно забезпечити такі функціональні вимоги:

- система повинна відповідати принципам горизонтальної масштабованості, як передбачається обробка великих обсягів даних, система повинна бути легко розширювана;
- на вхід система повинна приймати CSV файли. При цьому система повинна міти працювати з різними схемами даних в цих файлах, єдиною умовою є наявність обов'язкових полів, описаних вище. Схема даних повинна передаватися в систему при конфігуруванні;
- система повинна міти обробляти дані незалежно від похибки вимірювань;
- система повинна розбивати поїздки на ділянки руху з однаковим характером зміни швидкості;
- для виділених ділянок руху система повинна розраховувати;
- отриманий результат система повинна зберегти в CSV файл;
- система повинна мати можливість запису попередніх результатів роботи для аналізу їх адекватності.

4

Рисунок В.4 – Слайд 4

Вибір мови програмування

Критерій порівняння мов програмування	Мова програмування		
	Java	Python	C#
Підтримка парадигм програмування	Виключно ООП	Підтримує багато парадигм	Декілька парадигм
Надбудови та фреймворки для машинного навчання	Існують	Існують, багато	Існують
Активність спільноти	Висока	Найвища	Висока
Продуктивність застосунків	Висока	Середня	Висока
Розширюваність	Можлива	Можлива	Можлива

5

Рисунок В.5 – Слайд 5

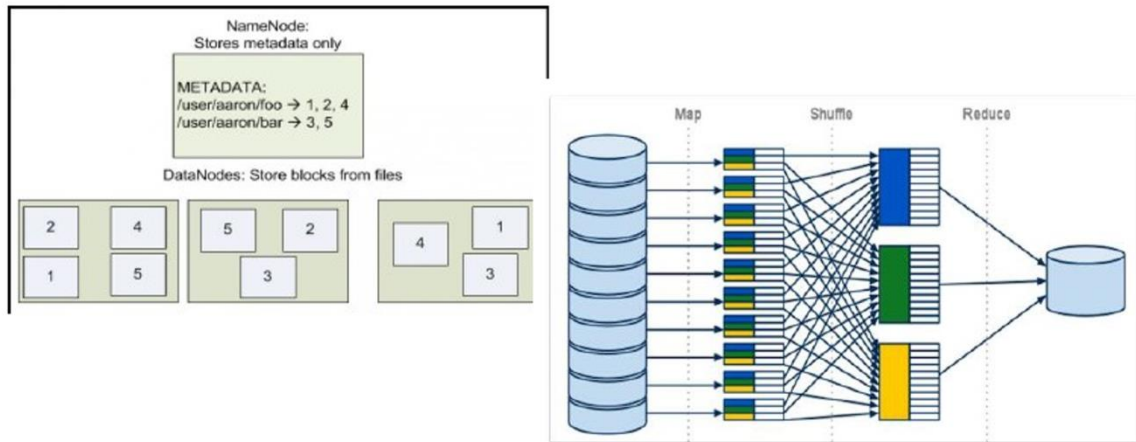
Вибір середовища розробки

Критерій порівняння середовищ розробки	Середовище розробки	
	Eclipse	IntelliJ IDEA
ПідтримкаHadoop	Так	Так
Розробка застосунків із розподіленими обчисленнями	Так	Так
Засоби рефакторингу	Так	Так
Тестування коду	Так	Так
Безкоштовність	Безкоштовне	Платне

6

Рисунок В.6 – Слайд 6

Використання структури HDFS



7

Рисунок В.7 – Слайд 7



8

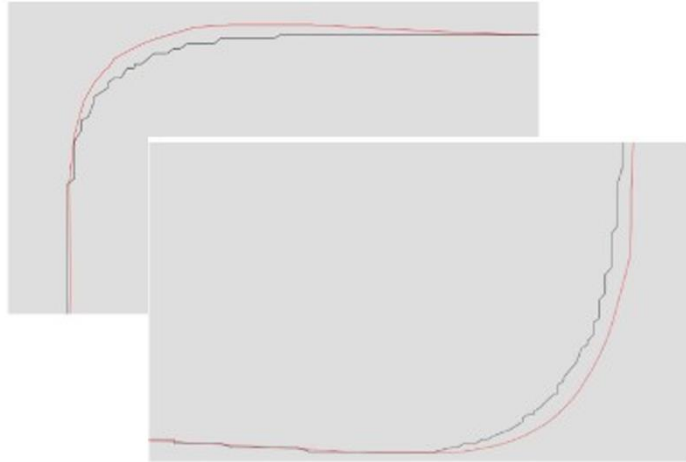
Архітектура розроблюваної системи

Рисунок В.8 – Слайд 8

Фільтрація та корекція вхідних даних

$$\hat{x}_k = \hat{x}_{k-1} + v_{k-1}(t_k - t_{k-1}) + a_{k-1} \frac{(t_k - t_{k-1})^2}{2}$$

Застосування фільтра
Калмана для обробки
геоданих



9

Рисунок В.9 – Слайд 9

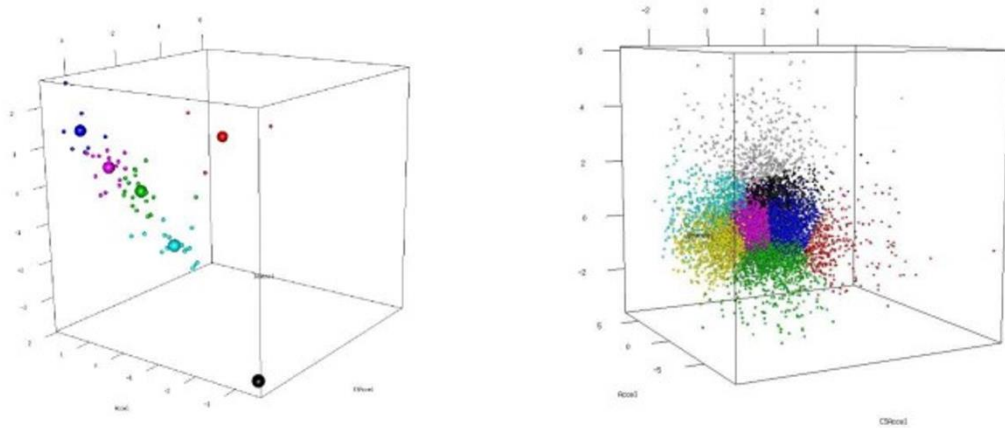
Кластеризація вхідних даних

№ кластеру	Доцентрове прискорення	Прискорення	Гальмування	Кількість водіїв
1	-0.2063358	-1.2018407	-1.1488741	13
2	-0.1827992	1.4781924	1.2344404	24
3	-0.2179239	0.4181136	0.4666877	1
4	-0.6739149	-3.4083534	-3.7190770	14
5	3.9781538	0.2282023	0.6838466	9
6	-0.1151139	-0.2914570	-0.3506687	3

10

Рисунок В.10 – Слайд 10

Графічне представлення отриманих кластерів



11

Рисунок В.11 – Слайд 11

Результати тестування

Була розроблена система, що дозволяє виділяти характеристики водіння і поїздок на підставі даних, одержуваних з систем геопозиціонування. Розроблена система задовольняє всім поставленим вимогам, а саме:

- система відповідає принципу горизонтального масштабування;
- система вміє працювати з різними схемами даних у вихідних файлах;
- система вміє працювати незалежно від похибок у вихідних даних. Для цих цілей був застосований алгоритм корекції даних, заснований на фільтрі Калмана. Коректність роботи алгоритму була перевірена за допомогою накладення отриманих траєкторій на географічну карту;
- система розбиває поїздки на ділянки з однаковим характером руху і обчислює динамічні характеристики для отриманих ділянок.

12

Рисунок В.12 – Слайд 12

Висновки

В ході виконання дипломної кваліфікаційної роботи магістра було проведено аналіз і порівняння існуючих рішень для обробки великих даних, отриманих з транспортних засобів.

Новизна роботи полягає в тому, що запропоновано метод, що може працювати незалежно від похибок у вихідних даних. Для цих цілей був застосований алгоритм корекції даних, заснований на фільтрі Калмана.

Практичне значення роботи полягає в тому, що розроблено програмне забезпечення для реалізації обробки великих даних, отриманих з транспортних засобів.

13

Рисунок В.13 – Слайд 13