МАГІСТЕРСЬКА РОБОТА

МР. ІПм - 56.00.00.000 ПЗ

Група ІПм-24-3

Штогрин Олександр

2025

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Штогрин Олександр Олександрович

(прізвище, ім'я, по батькові)

УДК 004.9

(індекс)

МАГІСТЕРСЬКА РОБОТА

Методи інтелектуального аналізу даних засобами розподілених динамічних фреймворків

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Штогрин О.О.

(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Зікратий Сергій Вікторович, к.т.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту

Завідувач кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Нормоконтроль

доц. Вовк Р.Б.

(посада) (підпис) (дата) (ініціали та прізвище)

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

Івано-Франківськ – 2025

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень магістр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**

доц. В.В. Бандура

“ 04 ” вересня 2025 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Штогрину Олександру Олександровичу

(прізвище, ім’я, по-батькові)

1. Тема магістерської роботи “ Методи інтелектуального аналізу даних засобами розподілених динамічних фреймворків ”

керівник проекту (роботи) Зікратий С.В., к.т.н., доцент

затверджені наказом закладу вищої освіти від “ 05 ” листопада 2025 р. № 695/7

2. Строк подання студентом проекту (роботи) 15 грудня 2025 р.

3. Вихідні дані до проекту (роботи) Формальні моделі і методи побудови систем інтелектуального аналізу даних засобами розподілених фреймворків

4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)

1. Дослідження предметної області аналізу великих даних засобами розподілених фреймворків
2. Задача проектування розподіленого фреймворку інтелектуального аналізу даних
3. Дослідження та опис технік і методів інтелектуального аналізу даних
4. Реалізація процесів ІАД засобами розподілених динамічних фреймворків

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Концепція 5V для великих даних (рис. 1.1)
2. Принцип роботи MapReduce (рис. 1.2)
3. Техніки просторового дата майнінгу (рис. 2.1)
4. Техніки просторової кластеризації (рис. 2.2)
5. Ілюстрація принципу роботи K-means алгоритму (рис. 2.3)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата
Перевірка на плагіат	доц., к.т.н. Вовк Р.Б.	

7. Дата видачі завдання 04 вересня 2025 р.

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір і вивчення літератури по темі магістерської роботи	15.09.2025	виконано
2	Дослідження предметної області аналізу великих даних засобами розподілених фреймворків	01.10.2025	виконано
3	Задача проектування розподіленого фреймворку інтелектуального аналізу даних	17.10.2025	виконано
4	Дослідження та опис технік і методів інтелектуального аналізу даних	02.11.2025	виконано
5	Реалізація процесів ІАД засобами розподілених динамічних фреймворків	19.11.2025	виконано
6	Адаптація підходу розподіленої динамічної кластеризації до парадигми MapReduce	02.12.2025	виконано
7	Затвердження пояснювальної записки роботи завідувачем кафедри	15.12.2025	виконано

Студент – магістр _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Магістерська робота: 76 с., 20 рис., 2 табл., 44 джерел.

Тема: Методи інтелектуального аналізу даних засобами розподілених динамічних фреймворків

Мета роботи - розробити теоретичні засади, архітектуру та алгоритмічне забезпечення розподіленого динамічного фреймворку кластеризації великих просторових даних.

Об'єкт дослідження - процеси інтелектуального аналізу великих та просторово-прив'язаних даних у розподілених обчислювальних системах.

Предмет дослідження - методи, алгоритми та архітектурні рішення, що забезпечують динамічну кластеризацію великих просторових наборів даних у розподілених фреймворках.

Результати дослідження

У роботі адаптовано парадигму MapReduce до задачі динамічної кластеризації, що дало змогу формалізувати процес обчислення в термінах map- та reduce-процедур.

Висновок

Запропоновано методологію та цілісну архітектуру розподіленого динамічного фреймворку кластеризації великих просторових даних, що поєднує локальний аналіз і геометричну агрегацію моделей.

BIG DATA, РОЗПОДІЛЕНІ ОБЧИСЛЕННЯ, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, ПРОСТОРОВИЙ ДАТАМАЙНІНГ, КЛАСТЕРИЗАЦІЯ, MAPREDUCE, HADOOP, HDFS, ГЕОМЕТРИЧНА АГРЕГАЦІЯ, РОЗПОДІЛЕНИЙ ФРЕЙМВОРК.

ABSTRACT

Master Thesis: 76 pp., 20 fig., 2 tab., 44 sources.

Topic: Methods of data mining using distributed dynamic frameworks

The aim of the work is to develop the theoretical foundations, architecture and algorithmic support of a distributed dynamic framework for clustering large spatial data.

The object of the research is the processes of big and spatially bound data mining in distributed computing systems.

The subject of the research is methods, algorithms and architectural solutions that provide dynamic clustering of large spatial data sets in distributed frameworks.

Research results

The paper adapts the MapReduce paradigm to the problem of dynamic clustering, which made it possible to formalize the calculation process in terms of map and reduce procedures.

Conclusion

A methodology and holistic architecture of a distributed dynamic framework for clustering large spatial data, combining local analysis and geometric aggregation of models, is proposed.

BIG DATA, DISTRIBUTED COMPUTING, INTELLIGENT DATA ANALYSIS, SPATIAL DATA MINING, CLUSTERIZATION, MAPREDUCE, HADOOP, HDFS, GEOMETRIC AGGREGATION, DISTRIBUTED FRAMEWORK.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 9
ВСТУП 10

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ АНАЛІЗУ ВЕЛИКИХ ДАНИХ ЗАСОБАМИ РОЗПОДІЛЕНИХ ДИНАМІЧНИХ ФРЕЙМВОРКІВ . 14
1.1. Концептуалізація процесів аналізу та обробки великих даних з використанням розподіленого фреймворку 14
1.2. Науковий аналіз феномену великих даних та виклики інтелектуального аналізу 16
1.3. Принципи роботи Hadoop MapReduce та порівняння з іншими розподіленими парадигмами 20
1.4. Задача проектування розподіленого фреймворку інтелектуального аналізу даних для великих масивів 22
1.4.1. Архітектурна основа та принцип локальної обробки 23
1.4.2. Основні завдання дослідження 24
Висновки до розділу 25

РОЗДІЛ 2. ДОСЛІДЖЕННЯ ТА ОПИС ТЕХНІК І МЕТОДІВ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ 27
2.1. Еволюція та категоризація технік інтелектуального аналізу даних в епоху цифрової інформації 27
2.1.1. Класифікація та категоризація 28
2.1.2. Вилучення шаблонів 28
2.1.3. Кластерний аналіз 29
2.2. Просторова аналітика та виклики інтелектуального аналізу просторово-прив'язаних даних (SDM) 29
2.3. Техніки інтелектуального аналізу просторово-прив'язаних даних 32
2.3.1. Просторові асоціативні правила 33

2.3.2. Просторова класифікація та прогнозування 33
2.4. Таксономія та методологічний огляд технік просторового датамайнінг у на основі кластерного аналізу 34
2.4.1 Кластеризація на основі розбиття 36
2.4.2. Ієрархічна кластеризація 37
2.4.3.Кластеризація на основі щільності 39
2.5. Масштабованість алгоритмів кластеризації на основі паралельних та розподілених парадигм 40
Висновки до розділу 42

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОЦЕСІВ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ ЗАСОБАМИ РОЗПОДІЛЕНИХ ДИНАМІЧНИХ ФРЕЙМВОРКІВ . 44
3.1. Представлення архітектури розподіленого фреймворку кластеризації для надвеликих наборів даних 44
3.2. Опис фази локального розподіленого аналізу даних 46
3.3. Представлення фази агрегації та генерації глобальних моделей 48
3.4. Методи вилучення репрезентативних границь простору 51
3.4.1. Опис роботи α-shape алгоритму 51
3.4.2. Алгоритм збалансованого вектора 54
3.5. Опис фреймворку Apache Hadoop для реалізація розподіленої динамічної кластеризації 56
3.5.1. Розподілена файлова система Hadoop (HDFS) 58
3.5.2. Парадигма MapReduce 59
3.6. Адаптація підходу розподіленої динамічної кластеризації до парадигми MapReduce 60
3.7. Експериментальна валідація підходу динамічної кластеризації 63
Висновки до розділу 68

ВИСНОВКИ 69
ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 72

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

MR - MapReduce

HDFS - Hadoop Distributed File System

DBSCAN - Density-Based Spatial Clustering of Applications with Noise

Eps - Epsilon - Параметр для DBSCAN

MinPts - Minimum Points – Параметр для DBSCAN

РДК – Розподілена динамічна кластеризація

ІАД - інтелектуальний аналіз даних

ВСТУП

Актуальність теми.

У сучасних умовах стрімкого зростання обсягів даних та поширення розподілених обчислювальних технологій особливої актуальності набувають методи інтелектуального аналізу даних, здатні ефективно функціонувати в динамічних, гетерогенних та ресурсно-обмежених середовищах. Традиційні алгоритми Data Mining, розроблені для централізованих систем, виявляються малопридатними для обробки надвеликих, різнорідних та просторово-прив'язаних наборів, у яких комунікаційні витрати, нерівномірна щільність і висока шумність значно знижують якість обчислень. У цьому контексті важливим науковим завданням є дослідження та адаптація класичних технік ІАД до принципів розподіленої обробки, а також розроблення архітектурних рішень, які забезпечують підтримку динамічної кластеризації великих масивів просторових даних.

Магістерська робота присвячена формуванню цілісного теоретико-методологічного підґрунтя, аналізу сучасних обчислювальних парадигм, розробленню архітектури розподіленого динамічного фреймворку та проведенню експериментальної оцінки його ефективності. Наукове дослідження охоплює систематизацію методів класифікації, кластеризації, вилучення асоціативних правил та просторового датамайнінгу, моделювання механізмів локального аналізу даних, побудову алгоритмів глобальної агрегації та інтеграцію обчислювальних процесів у парадигму MapReduce.

Результати роботи спрямовані на вирішення проблеми масштабованої кластеризації просторових даних з урахуванням їхньої внутрішньої топології, нерівномірної щільності та потреби в мінімізації комунікаційних витрат. Запропонований підхід поєднує геометричні методи реконструкції меж кластерів, локальне паралельне обчислення та збалансоване агрегування моделей, що дозволяє підвищити точність і стабільність отриманих результатів. Таким чином, робота робить внесок у розвиток методів ІАД,

побудову масштабованих платформ обробки Big Data та підвищення ефективності аналітичних процесів у критично важливих сферах.

Сучасні системи збору та моніторингу даних — мобільні мережі, супутникові системи, сенсорні платформи, геоінформаційні системи — генерують величезні обсяги неоднорідної та просторово орієнтованої інформації. Її обробка традиційними алгоритмами кластеризації стає неможливою через обмеження пам’яті, високу варіативність структури даних і необхідність паралельного виконання. Проблема ускладнюється тим, що просторові дані характеризуються топологічними залежностями, складними формами кластерів та значною шумністю, що вимагає застосування спеціальних геометричних та адаптивних методів.

Розподілені фреймворки на кшталт Hadoop, Spark і Flink відкривають нові можливості для масштабованої обробки, проте їхнє застосування до задач просторового датамайнінгу вимагає адаптації відповідних алгоритмів, що й становить наукову проблему.

Відсутність універсальних підходів до динамічної кластеризації великих просторових наборів, здатних працювати в реальному часі, з урахуванням локальних властивостей даних, робить дослідження в цій галузі особливо актуальним. Значимість проблеми зростає у сферах аналізу мобільності, екологічного моніторингу, геоаналітики, моделювання небезпечних процесів та інфраструктурного планування.

Саме тому розроблення архітектури розподіленого фреймворку, що забезпечує поєднання локального аналізу, геометричної реконструкції меж і глобальної агрегації кластерів, є своєчасним і важливим завданням сучасної науки.

Мета роботи - розробити теоретичні засади, архітектуру та алгоритмічне забезпечення розподіленого динамічного фреймворку кластеризації великих просторових даних.

Об’єкт дослідження - процеси інтелектуального аналізу великих та просторово-прив’язаних даних у розподілених обчислювальних системах.

Предмет дослідження - методи, алгоритми та архітектурні рішення, що забезпечують динамічну кластеризацію великих просторових наборів даних у розподілених фреймворках.

Завдання дослідження

1. Проаналізувати сучасні парадигми та фреймворки розподіленої обробки даних.

2. Систематизувати класичні та просторові техніки інтелектуального аналізу даних.

3. Дослідити особливості масштабування кластеризаційних алгоритмів у розподіленому середовищі.

4. Розробити архітектуру динамічного розподіленого фреймворку кластеризації.

5. Формалізувати процес локального аналізу даних на вузлах системи.

6. Адаптувати парадигму MapReduce для задач динамічної кластеризації.

Методи дослідження

Теоретичний аналіз наукових джерел; порівняльний аналіз парадигм розподілених обчислень; методи просторового датамайнінгу; алгоритми кластеризації на основі розбиття, ієрархії та щільності; геометричні методи (α-shape, збалансований вектор); моделювання у середовищах Hadoop та HDFS; експериментальне тестування масштабованості та продуктивності; статистичний аналіз результатів.

Наукова новизна отриманих результатів

1. Запропоновано цілісну архітектуру розподіленого динамічного фреймворку кластеризації великих просторових даних, що поєднує локальний аналіз і геометричну агрегацію моделей.

2. Сформовано підхід до побудови глобальних кластерів на основі α-shape та методу збалансованого вектора, адаптований до умов розподіленого середовища.

3. Удосконалено модель динамічної кластеризації шляхом інтеграції локальних часткових алгоритмів із механізмами MapReduce.

4. Доведено ефективність застосування геометричних методів реконструкції меж кластерів у задачах масштабованого просторового датамайнінгу.

Практичне застосування результатів

Розроблений розподілений фреймворк може бути використаний у:

– територіальних та геоінформаційних системах;

– екологічному та екосистемному моніторингу;

– аналізі мобільності населення та транспортних потоків;

– інтелектуальному аналізі великих даних у хмарних інфраструктурах.

Практичні результати роботи можуть бути інтегровані у програмні комплекси, що потребують масштабованої кластеризації та аналізу великих просторових наборів.

Структура магістерської роботи. Представлена робота складається зі вступу, трьох розділів та висновків. Загальний обсяг роботи становить 76 сторінок, і містить 20 рисунків, 2 таблиці, перелік використаних джерел із 44 позицій.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ АНАЛІЗУ ВЕЛИКИХ ДАНИХ ЗАСОБАМИ РОЗПОДІЛЕНИХ ДИНАМІЧНИХ ФРЕЙМВОРКІВ

1.1. Концептуалізація процесів аналізу та обробки великих даних з використанням розподіленого фреймворку

Щорічний обсяг генерованих даних демонструє експоненціальне зростання. У 2020 році цей показник сягнув понад 44×1012 гігабайт (ГБ), що у десять разів перевищує рівень 2003 року. Поточні тенденції свідчать про продовження цієї динаміки, що еквівалентно річному генеруванню понад 104 ГБ даних на одну особу. Цей феноменальний ріст призвів до виникнення терміна "Великі Дані" (Big Data). Великі Дані визначаються як надзвичайно великі, гетерогенні та швидко зростаючі набори даних, що походять з різноманітних галузей.

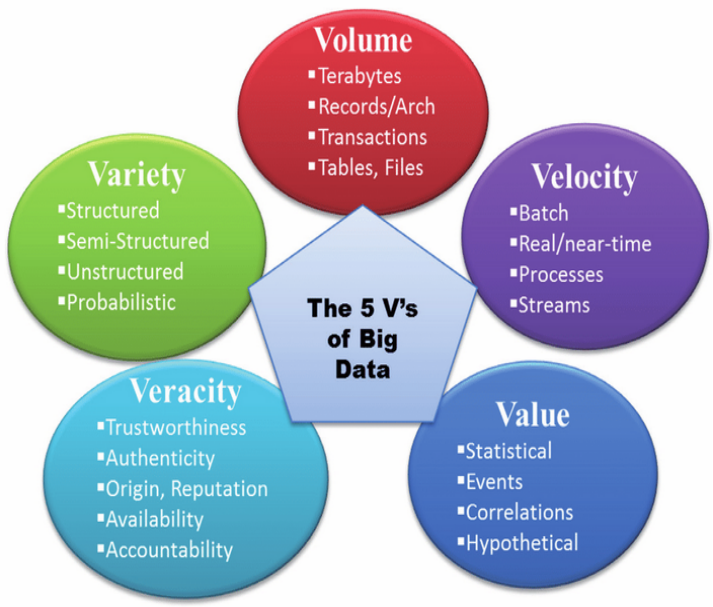

Рис. 1.1. Концепція 5V для великих даних

Аналіз та вилучення релевантної інформації з таких масивів становить значний науково-технічний виклик. Це зумовлено необхідністю у значній

обіймі потужності для зберігання та обробки, а також у розробці ефективних алгоритмів інтелектуального аналізу даних (data mining). Ці алгоритми повинні бути здатними ефективно справлятися не лише з обсягом (Volume), але й з гетерогенністю (Variety), шумом та швидкістю генерації (Velocity) даних. Крім того, важливим аспектом є достовірність (Veracity) даних. Комплекс цих характеристик, відомих як 4V/5V великих даних, вимагає архітектурних модифікацій у підходах до зберігання та управління даними, а також розробки нових аналітичних фреймворків. Традиційні методи дата майнінгу та машинного навчання, як правило, не володіють необхідною масштабованістю, потужністю та точністю для ефективного аналізу Великих Даних.

Розподілений дата майнінг є перспективним підходом для вирішення викликів великих даних. З огляду на те, що набори даних часто генеруються у розподілених географічних або логічних місцях, їх локальна попередня обробка та аналіз можуть суттєво зменшити затримку відповіді та обсяги міжмережевої комунікації.

В рамках даного дослідження було реалізовано розподілений фреймворк інтелектуального аналізу даних, призначений для аналізу великих даних. Цей фреймворк забезпечує прийнятний час відповіді, генерує точні результати та використовує існуючу комп'ютерну та сховищну інфраструктуру, зокрема хмарні обчислення. Фреймворк вирішує проблеми високопродуктивних обчислень (HPC) і є загальним, хоча був спеціально розроблений та реалізований для просторового дата майнінгу. Він здатний обробляти надзвичайно великі набори даних, справляючись з їх гетерогенністю та швидкістю.

Структура запропонованого підходу є двофазовою:

1. Фаза генерації локальних моделей.

На цьому етапі аналізуються набори даних, розташовані на кожному сайті, використовуючи різноманітні техніки кластеризації.

2. Фаза агрегації.

Метою цієї фази є агрегування локальних результатів для формування глобальних моделей. Дизайн агрегаційної фази спрямований на забезпечення компактності та точності фінальних кластерів, при цьому зберігаючи ефективність загального процесу з точки зору часової та пам'ятійної складності.

Підхід був ретельно протестований та порівняний із відомими алгоритмами кластеризації. Отримані результати демонструють, що запропонований фреймворк не лише забезпечує високоякісні результати порівняно з існуючими методами, але й демонструє значне прискорення та відмінну масштабованість при використанні парадигми Hadoop MapReduce.

1.2. Науковий аналіз феномену великих даних та виклики інтелектуального аналізу

Сучасні досягнення в інформаційних та комунікаційних технологіях (ІКТ), технологіях цифрових сенсорів та убіквітарних обчисленнях забезпечили безпрецедентну можливість для акумуляції терабайтів даних у різних предметних областях. Наприклад, лише у 2017 році соціальні медіа демонстрували генерацію понад 2×108 електронних листів, 3×105 твітів, 2×106 "лайків" на Facebook та 2×105 завантажень фотографій щохвилини. Очікується, що ці показники продовжуватимуть зростати експоненційно, підтверджуючи настання епохи великих даних (Big Data). Незважаючи на масивне та швидке накопичення даних, лише незначна їх частина ефективно використовується для аналізу.

Масштабний потік даних, хоча і обіцяє розв'язання численних нерозв'язних раніше проблем, створив критичні виклики для існуючих інформаційних систем та інструментів інтелектуального аналізу (data mining) з точки зору як обробної потужності, так і продуктивності. Ці виклики узагальнюються через характеристики 4V: Обсяг (Volume), Різноманітність (Variety), Швидкість (Velocity) та Достовірність (Veracity).

Обсяг (Volume) - надзвичайна величина наборів даних виводить наявні інструменти інтелектуального аналізу за межі їхньої функціональності, унеможливлюючи обробку даних у розумний часовий проміжок із забезпеченням необхідної точності.

Різноманітність (Variety) - дані часто є неструктурованими та гетерогенними (числові значення, текст, зображення, мультимедіа). Це унеможливлює ефективне застосування традиційних методів підготовки даних, що використовуються у процесі виявлення знань у базах даних (KDD).

Швидкість (Velocity) - висока швидкість надходження даних (у деяких застосуваннях — у реальному часі) ускладнює їх оперативний аналіз. Це критично для своєчасного отримання повної переваги від видобутих знань.

Достовірність (Veracity) - зібрані дані часто містять невизначеність, неточність та шум. Завдання очищення, перетворення та забезпечення якості сирих даних є надзвичайно складним і має вирішальний вплив на достовірність кінцевих аналітичних результатів.

Хоча ці проблеми не є новими для комп'ютерних наук, сукупна присутність усіх цих викликів одночасно є безпрецедентною. Ключовим завданням є вилучення цінних знань із "великих" даних у динамічних умовах.

Провідні технологічні корпорації (Google, Facebook, Amazon, Microsoft) та постачальники хмарних послуг вже накопичили значний досвід у зборі та зберіганні масивних обсягів даних. Однак, ефективні інструменти інтелектуального аналізу в реальному часі залишаються предметом активного дослідження.

Наприклад, пошукова система Google є прикладом успішної обробки обсягу та швидкості даних. Проте, поточні пошукові механізми не підтримують складні аналітичні запити, типові для моделі OLAP (Online Analytical Processing).

Або Facebook успішно збирає велику різноманітність даних (текст, зображення, мультимедіа), але стикається з відсутністю ефективних

інструментів майнінгу для швидкої класифікації контенту (наприклад, визначення легального/нелегального або пристойного/непристойного матеріалу).

Ці приклади підкреслюють, що проблема великих даних залишається відкритою, а одночасне розв'язання всіх її викликів є надзвичайно складним науковим завданням.

Незважаючи на виклики, існує дві ключові доступні вимоги:

1. Потужність зберігання доступна та економічно доцільна можливість збору та зберігання необмежених обсягів даних.

2. Обчислювальна потужність - доступна та масштабована обчислювальна інфраструктура, насамперед через хмарні обчислення.

Хмарні обчислення пропонують масивну обчислювальну потужність (через мільйони динамічно розподілюваних процесорів) та сховище. Вони забезпечують високопродуктивні обчислення (HPC), включаючи високошвидкісні мережі, швидкі сховища та великі обсяги пам'яті. Це обумовлює ключове завдання для розробки нових фреймворків інтелектуального аналізу — оптимальне використання можливостей хмарної інфраструктури.

Дата майнінг визначається як процес виявлення нових значущих кореляцій, шаблонів і тенденцій шляхом просіювання великих обсягів даних, що зберігаються в репозиторіях, з використанням технологій розпізнавання шаблонів, а також статистичних і математичних методів.

Проте, виклики великих даних вивели традиційний процес KDD за його межі:

- підготовка даних: збір, очищення, попередня обробка та інтеграція гетерогенних і динамічних даних (наприклад, веб-даних) у сховище даних стає неможливим або недоцільним завданням.

- динамічність: висока динамічність веб-даних робить традиційний аналіз неефективним.

- припущення про шум: більшість традиційних алгоритмів інтелектуального аналізу припускають, що дані не містять шуму. У реальних даних, особливо у великих даних, ця передумова є недійсною, що компрометує точність результатів.

- висока розмірність (high dimensionality): веб-дані часто є високовимірними, що спричиняє експоненційне зростання простору пошуку та негативно впливає на продуктивність алгоритмів.

- обчислювальна складність: багато традиційних методів машинного навчання та статистичних методів мають експоненційну або поліноміальну складність середнього порядку. У контексті великих даних нормою для аналітичних алгоритмів є лінійна складність (O(N)).

Було застосовано два основні підходи для подолання викликів великих даних:

1. Паралелізм - передбачає розподіл обчислень між великою кількістю процесорів.

Недоліки: Ці підходи припускають, що вхідні дані вже попередньо оброблені та доступні, не справляються з гетерогенністю та створюють залежності від даних, що вимагають механізмів комунікації та синхронізації, знижуючи їхню ефективність.

2. Розподілені системи - використання великої кількості автономних обробних вузлів.

Переваги: здатність справлятися з гетерогенністю даних.

Недоліки: слабке зв'язування вузлів та значні накладні витрати на координацію обробки та агрегацію кінцевих результатів.

Хоча паралельні та розподілені версії існують для багатьох алгоритмів (зокрема, асоціативних правил та класифікації), розподілених версій алгоритмів кластеризації для надзвичайно великих наборів даних існує лише обмежена кількість. Це підтверджує необхідність розробки нових, спеціалізованих фреймворків.

1.3. Принципи роботи Hadoop MapReduce та порівняння з іншими розподіленими парадигмами

MapReduce — це програмна модель і асоційована реалізація для паралельної обробки великих наборів даних (великих даних) на кластері обчислювальних вузлів. Вона складається з двох основних фаз: map (відображення) та reduce (зведення).

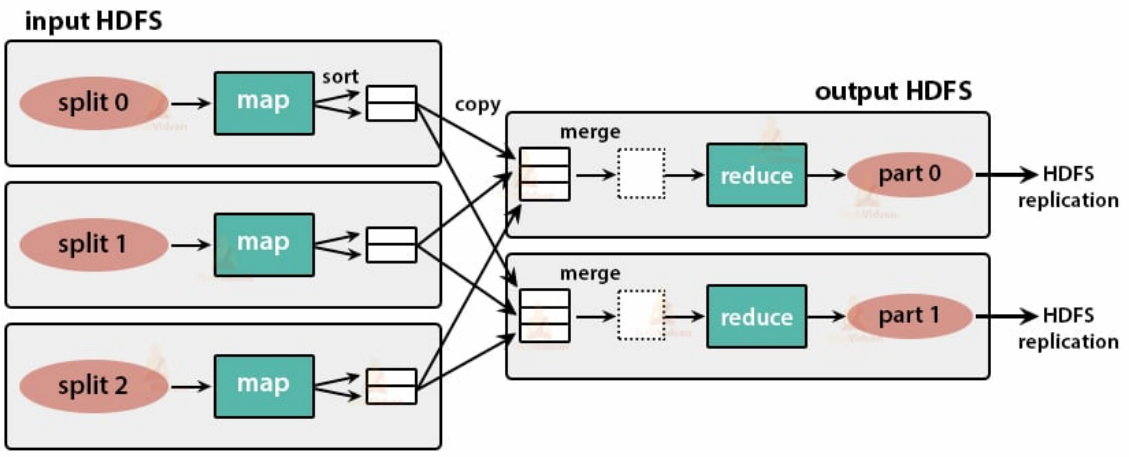

Рис. 1.2. Принцип роботи MapReduce

1. Фаза map (відображення)

Призначення: Обробка вхідних даних паралельно.

Вхідні дані (зазвичай великий файл) розбиваються на менші, незалежні блоки. Кожен блок обробляється окремою функцією Map. Ця функція бере вхідну пару (ключ, значення) і перетворює її на набір проміжних пар (ключ, значення).

2. Фаза Shuffle & Sort (перетасовка та сортування)

Призначення: Групування проміжних результатів для подальшого зведення.

Проміжні пари, згенеровані функцією Map, сортуються та групуються за ключем. Усі значення, асоційовані з одним і тим же ключем, передаються разом до однієї функції Reduce.

3. Фаза Reduce (зведення)

Призначення: Агрегація проміжних результатів для отримання кінцевого вихідного набору даних.

Функція Reduce отримує проміжний ключ та список усіх значень, асоційованих з цим ключем. Вона виконує зведення або агрегацію цих значень, генеруючи кінцевий вихідний набір пар (ключ, значення).

Хоча MapReduce був піонером, його основним недоліком є низька ефективність для ітеративних обчислень (наприклад, для багатьох алгоритмів машинного навчання, які вимагають повторних проходів по даних) через необхідність запису проміжних результатів на диск (дисковий I/O) після кожної фази Map/Reduce.

Сучасні фреймворки, такі як Apache Spark, вирішують цю проблему, використовуючи обробку в пам'яті (in-memory processing) через концепцію Resilient Distributed Datasets (RDDs) або DataFrames. Це робить Spark значно швидшим для ітеративних завдань.

Таблиця 1.1.

Різниця між Hadoop MapReduce та Apache Spark

Характеристика	**Hadoop MapReduce**	**Apache Spark**
Обробка даних	На основі диска (Disk-based I/O)	В пам'яті (In-memory computing)
Швидкість	Повільніший для ітеративних завдань	Значно швидший (до 100x в пам'яті)
Модель	Дві фіксовані фази (Map та Reduce)	Гнучкий граф спрямованих циклічних обчислень (DAG)
Ітеративність	Неефективний (потребує повторного I/O)	Високоефективний

Основна відмінність між MapReduce та сучасними фреймворками, такими як Spark, полягає в тому, як вони керують проміжними результатами обчислень.

1. Обробка на основі диска (Hadoop MapReduce)

MapReduce використовує парадигму обробки на основі диска. Це означає, що після завершення фази Map, проміжні результати обов'язково

записуються на диск (Hard Disk Drive, HDD або Solid State Drive, SSD). Лише після цього фаза Reduce може почати читати ці дані з диска.

Перевага: висока надійність та відмовостійкість, оскільки дані зберігаються постійно.

Недолік: значні накладні витрати на введення/виведення (I/O). Для ітеративних алгоритмів (коли процес Map/Reduce виконується багаторазово), кожен прохід вимагає повільного запису та читання з диска.

2. Обробка в пам'яті (Apache Spark)

Apache Spark (та подібні фреймворки) використовують обробку в оперативній пам'яті (RAM) через механізм Resilient Distributed Datasets (RDDs) або DataFrames. Проміжні дані обчислень (наприклад, результат фази Map) зберігаються в оперативній пам'яті кластера (кешуються) між послідовними кроками обробки.

Перевага: кардинальне прискорення для ітеративних завдань (до 100 разів), оскільки доступ до RAM набагато швидший, ніж до диска.

Недолік: вимагає більшого обсягу оперативної пам'яті; при її нестачі або збої вузла дані можуть бути втрачені (хоча RDDs забезпечують відмовостійкість шляхом переобчислення).

Отже, для аналізу великих даних, особливо у складних сценаріях (машинне навчання, графові обчислення), які вимагають багатьох ітерацій, обробка в пам'яті (In-memory processing), яку пропонує Spark, є домінуючою парадигмою.

1.4. Задача проектування розподіленого фреймворку інтелектуального аналізу даних для великих масивів

Основною рушійною силою даного дослідження є необхідність пропозиції, проектування та розробки фреймворку інтелектуального аналізу даних (data mining), здатного забезпечити аналіз великих даних у межах прийнятного часу відповіді та генерувати точні результати. Ключовою

вимогою є ефективне використання існуючої та поточної обчислювальної та сховищної інфраструктури. Запропонований фреймворк є розподіленим і спрямований на вирішення проблем високопродуктивних обчислень (HPC).

Оскільки в науковій літературі зафіксовано обмежену кількість розподілених алгоритмів кластеризації для масивних наборів даних, основна увага в даній роботі приділяється кластеризації як основній техніці інтелектуального аналізу. Розроблений розподілений фреймворк кластеризації, хоча і реалізований для просторового дата майнінгу, є загальним і здатний обробляти надзвичайно великі дані, ефективно вирішуючи проблеми гетерогенності та швидкості (Velocity) наборів даних.

1.4.1. Архітектурна основа та принцип локальної обробки

У більшості застосувань великих даних дані збираються та зберігаються розподілено у різних місцях. Це обумовлено міркуваннями близькості (зберігання даних у місці їх генерації) або потребами у достатній потужності зберігання та реплікації. Важливою особливістю є мінливість якості та гетерогенність атрибутів даних, зібраних різними інструментами або сенсорами в різних локаціях. Традиційні методи подолання цієї гетерогенності, такі як створення складних метасловників та інструментів інтеграції даних, не завжди є оптимальними.

Основна концепція запропонованого розподіленого фреймворку кластеризації полягає у проведенні локального інтелектуального аналізу даних (local data mining) на кожній ділянці з подальшим видобутком знань (knowledge mining) з цих локальних результатів для генерації глобальних знань.

Переваги локального інтелектуального аналізу даних:

1. Масштабованість (обсяг). Обсяг даних перестає бути критичною проблемою, оскільки кожен вузол обробляє лише свою частку, значно зменшуючи розмірність локальної задачі.

2. Гетерогенність (різноманітність). Проблема гетерогенності ефективно вирішується на локальному рівні через адаптовану попередню обробку даних.

3. Адаптивність алгоритмів. Локальні дані можуть бути проаналізовані за допомогою алгоритмів, оптимально пристосованих до їхніх характеристик та специфічних локальних потреб.

4. Конфіденційність. Локальна обробка підвищує конфіденційність даних, усуваючи необхідність передачі чутливих сирих даних до централізованого сховища.

1.4.2. Основні завдання дослідження

Зважаючи на складність великих даних та необхідність розробки ефективних рішень, визначено такі наукові цілі.

Концентрування на просторових даних, які становлять значну частину світових даних (за оцінками, до 80% даних є просторовими або містять просторовий елемент). Цей тип великих даних, що генерується від GPS, дистанційного зондування та сенсорів, вимагає розробки нових методів аналізу та візуалізації.

Розробка гібридного підходу до кластеризації. Створення підходу до кластеризації, який поєднує майнінг даних та майнінг знань. Метою цього гібридного підходу, на відміну від простих механізмів зворотного зв'язку, є зниження обчислювальної складності та підвищення якості кінцевих результатів.

Мінімізація комунікаційних витрат у розподілених системах. Розробка підходу, який є принципово розподіленим, що забезпечує спільне використання ресурсів та прискорене вирішення проблеми. Критичним завданням є мінімізація накладних витрат на комунікацію, які є значними у розподілених системах порівняно зі швидкістю процесора та пам'яті.

Експлуатація хмарної інфраструктури. Використання існуючих обчислювальних ресурсів, зокрема хмарних обчислень, як розподіленої

платформи, та MapReduce як моделі програмування для реалізації фреймворку.

Експериментальне порівняння реалізацій. Порівняння двох різних реалізацій фреймворку:

1. Внутрішня реалізація на гетерогенній мережі комп'ютерів зі звичайним мережевим з'єднанням.

2. Кластерна реалізація: на спеціалізованому кластері з використанням парадигми MapReduce (як репрезентативного прикладу поширених хмарних технологій).

Це порівняння дозволить не лише валідувати фреймворк у різних обчислювальних середовищах, але й оптимізувати його конфігурацію та визначити найбільш ефективні інструменти для кожного середовища.

Висновки до розділу

У першому розділі здійснено всебічний аналіз предметної області обробки великих даних, що дав змогу сформувати цілісне уявлення про ключові принципи та виклики, притаманні сучасним розподіленим обчислювальним системам. Показано, що феномен Big Data є багатовимірним і характеризується стрімким зростанням обсягів, швидкості надходження та різнорідності інформаційних потоків. Визначено, що традиційні централізовані методи обробки не здатні забезпечити необхідний рівень масштабованості та стійкості, що вимагає застосування розподілених архітектур та динамічних фреймворків. Проведена концептуалізація процесів обробки великих даних дозволила встановити фундаментальні вимоги до таких систем, серед яких домінують локальна обробка, паралельність виконання, розподілене керування пам’яттю та толерантність до збоїв. На основі аналізу парадигм Hadoop MapReduce, Apache Spark, визначено їхні структурні відмінності й умови оптимального застосування. Особливу увагу приділено моделі MapReduce, яка виявилась найбільш придатною для

реалізації задач, що вимагають детермінованої обробки великих проміжних обсягів даних. Розглянуто задачу проектування розподіленого фреймворку ІАД та встановлено її залежність від архітектурних характеристик інфраструктури, просторової організації даних і властивостей застосовуваних алгоритмів.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ ТА ОПИС ТЕХНІК І МЕТОДІВ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ

2.1. Еволюція та категоризація технік інтелектуального аналізу даних в епоху цифрової інформації

Сучасна епоха характеризується домінуванням цифрової інформації, що надає можливість збору вичерпних деталей про суб'єкти, транзакції, процеси та події в різних форматах (текст, дані, аудіо, відео, зображення). У цьому цифровому контексті дані є центральним активом будь-якої організації, що безпосередньо впливає на її розвиток та успіх. Хоча проблеми збору та зберігання даних значною мірою вирішені завдяки зростаючій потужності сховищ та удосконаленню методів збору, зберігається гостра потреба в ефективних та дієвих аналітичних техніках для вилучення корисної інформації, необхідної для прийняття рішень та поглиблення наукового розуміння.

Інтелектуальний аналіз даних (Data Mining) має історію понад чотири десятиліття, але набув широкого визнання в сучасному контексті виявлення знань у базах даних (Knowledge Discovery in Databases, KDD) наприкінці 1980-х — на початку 1990-х років. Спочатку Data Mining був визнаний як один із ключових підпроцесів KDD. З часом терміни Data Mining та KDD почали використовуватись як синоніми. На сьогодні термін KDD використовується рідше, оскільки аналіз охоплює не лише бази даних, а й увесь спектр даних різнорідних типів і форматів.

У ширшому науковому та бізнес-середовищі термін Data Mining часто замінюється сучасною термінологією, такою як "Аналіз даних" (Data Analysis), "Бізнес-інтелект" (Business Intelligence), "Бізнес-аналітика" (Business Analytics) та "Великі Дані" (Big Data), відображаючи розширення сфери застосування.

Техніки інтелектуального аналізу даних набули значної популярності з 1990 року, що було підтверджено розвитком престижних конференцій (ICDM, PKDD, ECML). Це зростання обумовлене природною еволюцією обчислювальної потужності, доступністю великих обсягів даних та розвитком просунутих алгоритмів.

Техніки Data Mining загалом класифікуються на три основні групи:

- класифікація та категоризація
- вилучення шаблонів
- кластерний аналіз

2.1.1. Класифікація та категоризація

Класифікація є технікою навчання з учителем (supervised learning), що передбачає побудову моделі-класифікатора на основі позначених об'єктів даних. Мітки представляють попередньо визначені, неперетинні класи.

Фаза навчання використовує ці позначені дані для побудови класифікатора, який згодом застосовується для класифікації невидимих об'єктів. До найпоширеніших технік класифікації належать дерева рішень, машини опорних векторів (SVM), класифікація на основі асоціацій (CBA), регресія, нейронні мережі, байєсові мережі, розуміння на основі випадків (CBR) та генетичний алгоритм.

Хоча ці техніки ефективні для наборів даних розумного розміру, деякі з них (наприклад, нейронні мережі, SVM) мають високу обчислювальну складність, інші (наприклад, CBR) — проблеми з точністю, а деякі — специфічні обмеження щодо типів даних.

2.1.2. Вилучення шаблонів

Метою вилучення шаблонів є ідентифікація значущих моделей, шаблонів або "знань" у даних. Шаблон може набувати різних форм, залежно від галузі застосування (наприклад, вивчення звичок покупців в аналізі кошика).

Типова техніка - Майнінг асоціативних правил (Association Rule Mining), де видобуті правила зазвичай мають форму "ЯКЩО <умова> ТО <результат>", що забезпечує високу інтерпретованість.

Популярні алгоритми: Apriori, FP-growth. Основні проблеми включають надзвичайно високу генерацію кандидатів під час пошуку, що ускладнює їх дослідження для великих наборів даних; високу чутливість до шуму в даних; та складність обрізання (pruning) великих деревоподібних структур.

2.1.3. Кластерний аналіз

Кластеризація — це техніка навчання без учителя (unsupervised learning), основною метою якої є поділ об'єктів даних на неперетинні групи (кластери) таким чином, щоб об'єкти в межах одного кластера мали спільні характеристики. На відміну від класифікації, кластери динамічно визначаються на основі внутрішньої структури самих даних, виступаючи як метод автоматичного групування. Кластерний аналіз має широкий спектр застосувань у біології, охороні здоров'я, аналізі документів, кліматології, науці (астрофізика, геологія), бізнесі та інженерії.

В контексті даного дослідження особлива увага приділяється технікам, що використовуються для аналізу великих обсягів просторових даних (Spatial Data Mining, SDM). Це включає огляд SDM та його специфічних викликів, а також всебічний огляд технік SDM. Центральний фокус зосереджується на ключових ідеях підходів до кластеризації, придатних для обробки надзвичайно великих наборів даних.

2.2. Просторова аналітика та виклики інтелектуального аналізу просторово-прив'язаних даних (SDM)

Протягом останніх двох десятиліть спостерігається значне зростання популярності просторових наборів даних. Ця тенденція є наслідком прогресу

в апаратних технологіях та розповсюдження сучасних методів збору даних, включаючи глобальні системи позиціонування (GPS), веб-орієнтовані сервіси на основі місцезнаходження, дистанційне зондування високої роздільної здатності та спільне картографування.

Просторові дані (Spatial Data) стосуються наборів даних, пов'язаних із простором або географічними регіонами. Вони описують об'єкти, що займають певне місце. Просторова база даних (Spatial Database) зберігає просторові об'єкти, представлені специфічними просторовими типами даних (кубоїди, полігони, лінії) та включає просторові відносини між цими об'єктами. Поширеними джерелами таких даних є супутникові та медичні зображення, структури білків тощо.

Аналіз просторових даних має глибокі корені у традиційних дисциплінах, таких як статистичний просторовий аналіз (Statistical Spatial Analysis), картографія та обчислювальна геометрія.

Традиційний статистичний просторовий аналіз, хоча й ефективний для числових даних та здатний генерувати реалістичні моделі, страждає від критичних обмежень:

- Статистичний аналіз вимагає припущення про статистичну незалежність між просторово розподіленими даними, що суперечить реальній взаємопов'язаності просторових об'єктів.

- Хоча регресійні моделі можуть частково вирішувати проблему взаємозв'язку, їх розробка та застосування є складною і вимагає глибокої експертизи у статистиці та галузевих знаннях.

- Статистичні підходи неефективні при роботі з нелінійними, неповними або зашумленими даними.

- Ці техніки характеризуються високою обчислювальною складністю та є дорогими для великомасштабного практичного застосування.

З появою інтелектуального аналізу даних виникла нагальна потреба в просторовому дата майнінгу (SDM) — ефективних та продуктивних методах

для виявлення невідомих та несподіваних відомостей з масивних і складних просторових наборів даних.

Процес вилучення корисних шаблонів із просторових наборів даних є більш складним порівняно з традиційними числовими даними. Це зумовлено складністю просторових типів даних, просторових відносин та просторової автокореляції [20].

Основні виклики SDM включають:

1. Геометричне представлення та властивості об'єктів

Просторові об'єкти займають місце та характеризуються геометрією (лінія, площа), визначеною у певній системі відліку. Ця геометрія неявно визначає просторові властивості (орієнтація) та просторові відносини (топологічні, відстаневі, напрямкові) [21].

2. Просторові відносини та реляційна природа шаблонів

Розташування об'єктів визначає їхні просторові відносини (топологічні, відстаневі, напрямкові). Наприклад, відстаневі відносини між точками часто обчислюються на основі евклідової метрики.

Реляційна природа просторових шаблонів вимагає ефективного обчислення цих відносин, що є критично важливим для розробки методів аналізу даних.

3. Складність вхідних даних

Вхідні дані SDM є складнішими, оскільки включають розширені об'єкти (точки, лінії, полігони) на відміну від простих числових/категорійних значень.

Вхідні дані містять непросторові атрибути (наприклад, населення, ім'я), які є явними, та просторові атрибути (широта, довгота, форма), які визначають просторове розташування.

Відносини між просторовими об'єктами часто є неявними (наприклад, перекриття, перетин, позаду), на відміну від явних відносин між непросторовими об'єктами (наприклад, є підкласом) [22].

5. Просторова автокореляція

Просторові дані демонструють високу саморегуляцію. Ця фундаментальна властивість означає, що значення атрибута в одній просторовій локації не є незалежним від значень у сусідніх локаціях (наприклад, подібність економік регіонів, градієнтні зміни температури), що необхідно враховувати при розробці аналітичних моделей.

2.3. Техніки інтелектуального аналізу просторово-прив'язаних даних

Ефективний інтелектуальний аналіз просторових даних (SDM) вимагає адаптації традиційних методів для подолання викликів, пов'язаних із геометрією, просторовими відносинами та автокореляцією. Основні техніки SDM включають просторові асоціативні правила, просторову класифікацію, просторове прогнозування та кластеризацію (рисунок 2.1). Значна частина сучасних досліджень зосереджена на кластеризації через її здатність виявляти приховану просторову структуру.

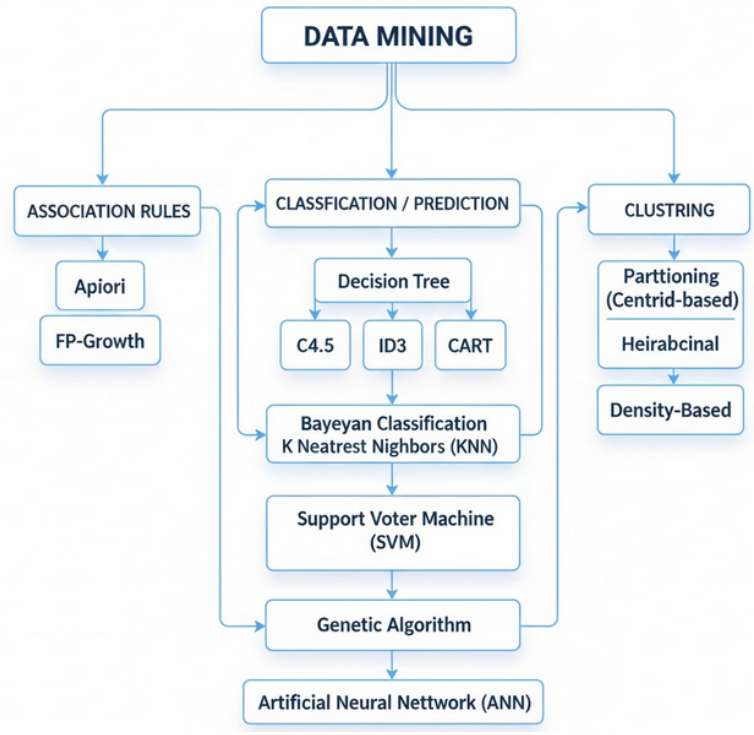

Рис. 2.1. Техніки просторового дата майнінгу

2.3.1. Просторові асоціативні правила

Майнінг асоціативних правил (Association Rule Mining, ARM) був спочатку розроблений для виявлення регулярностей у транзакційних базах даних. Метою є знаходження частих шаблонів, асоціацій або кореляцій між наборами елементів ($I=\{i1,i2,\ldots,im\}$) у наборі транзакцій D. Правило має вигляд $X \Rightarrow Y$, де X та Y є неперетинними підмножинами I.

Правило характеризується підтримкою (s), яка є часткою транзакцій, що містять $X \cup Y$, та впевненістю (c), що є умовною ймовірністю вмісту Y за наявності X. Розглянемо традиційні алгоритми. Apriori використовує ітеративний підхід, генеруючи кандидатні k-елементні набори з частих $(k-1)$-елементних наборів. Головний недолік — висока обчислювальна складність через генерацію значної кількості кандидатів. FP-growth вирішує обмеження Apriori шляхом уникнення генерації кандидатів. Він стискає набір транзакцій у структуру FP-дерева (Frequent Pattern Tree) та використовує методологію "розділяй і володарюй" для ефективного виявлення частих шаблонів.

Просторові асоціативні правила (SAR) мають вигляд $X \Rightarrow Y(c)$, де X та Y є наборами просторових або непросторових атрибутів.

Приклад: is_a(x,school) ⇒ close_to(x,park) (80%) (80% шкіл розташовані поблизу парків).

Більшість досліджень SAR зосереджувалися на підвищенні ефективності, спрямованої на зменшення часу генерації кандидатських наборів, що є критичним для великомасштабних просторових наборів даних.

2.3.2. Просторова класифікація та прогнозування

Класифікація (навчання з учителем) — це процес присвоєння елементам даних попередньо визначених категорій (класів). Просторові методи класифікації розширюють загальні методи (наприклад, дерева рішень, SVM, ANN) для врахування не лише атрибутів об'єкта, що класифікується, але й атрибутів сусідніх об'єктів та їх просторових відносин.

В [22] запропонували підхід на основі алгоритму ID3, який використовує концепцію графа сусідства, включаючи топологічні (перетинаються), метричні (близько до) та напрямкові відносини. Візуальні підходи, що поєднують традиційні алгоритми (наприклад, C4.5) з картографічною візуалізацією, застосування для аналізу та прогнозування просторової виборчої поведінки, а також класифікація пікселів або об'єктів зображень (наприклад, для картографування районів лісових пожеж).

Прогнозування відрізняється від класифікації тим, що оперує безперервними або впорядкованими значеннями (на відміну від дискретних категорій). Просторові моделі прогнозування формують особливу групу регресійного аналізу, яка включає просторову залежність між об'єктами.

Просторова авторегресійна модель (SAR) є популярним прикладом. Вона використовує дані незалежних та/або залежних сусідніх об'єктів для прогнозування залежностей у певному місці.

Приклад. Використання SAR для оцінки цін продажу майна, де враховується просторово-часова авторегресія. Для визначення залежностей SAR вимагає маніпуляцій з просторовою матрицею ваг розміром n×n. Основною обчислювальною проблемою є необхідність обчислення логарифма визначника великої матриці $(I-\rho W)$ шляхом знаходження всіх власних значень матриці. Для обробки дуже великих наборів даних дослідження зосереджуються на наближених рішеннях, таких як розклад у ряд Тейлора або поліноми Чебишова, що значно зменшує споживання пам'яті та час виконання при збереженні високої точності.

2.4. Таксономія та методологічний огляд технік просторового датамайнінг у на основі кластерного аналізу

Кластеризація або класифікація без учителя (unsupervised classification), є фундаментальною технікою інтелектуального аналізу даних, яка використовується для групування об’єктів даних (з немаркованими даними) у

кластери. Основна мета полягає у максимізації подібності об’єктів у межах одного кластера та мінімізації подібності між об’єктами, що належать до різних кластерів. Після етапу групування кластерний аналіз передбачає ідентифікацію прихованих структур усередині сформованих кластерів.

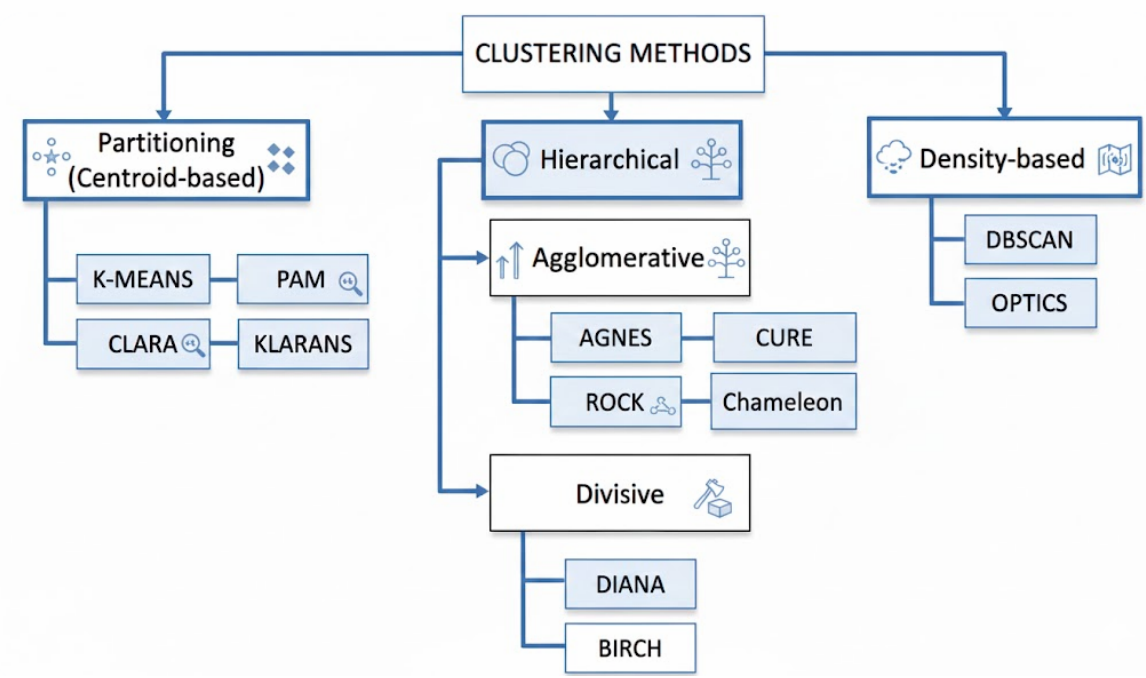

Рис. 2.2. Техніки просторової кластеризації

Основні категорії технік кластеризації:

- методи на основі розбиття
- ієрархічні методи
- методи на основі щільності

Вимоги до якісного алгоритму кластеризації:

- мінімальні вимоги до вхідних параметрів: необхідність мінімізації попереднього визначення вхідних параметрів (наприклад, k), оскільки оптимальні значення часто невідомі для великих масивів даних.

- виявлення довільних форм: здатність ідентифікувати кластери довільних форм (неопуклі, сферичні, витягнуті, лінійні), що є типовим для просторових наборів даних.

- висока точність та масштабованість: забезпечення високої точності результатів навіть при роботі з надзвичайно великими наборами даних.

2.4.1 Кластеризація на основі розбиття

Ці методи спрямовані на виявлення k розбиттів (кластерів) з n точок даних шляхом оптимізації цільової функції через ітеративний процес. Кожний об’єкт даних повинен належати рівно до одного кластера (за винятком нечітких методів). Критерієм якості є мінімізація відстані між об'єктами в межах кластера та максимізація відстані між кластерами.

K-means - базовий і найпопулярніший алгоритм, де кожен кластер представлений його середнім значенням (центроїдом).

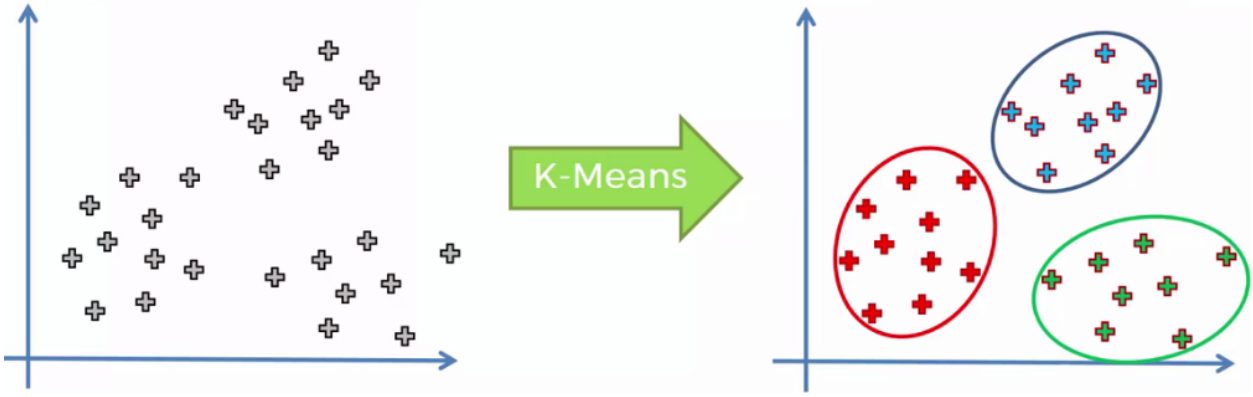

Рис. 2.3. Ілюстрація принципу роботи K-means алгоритму

K-medoid (PAM) - варіант K-means для категоріальних даних, де кластер представлений медоїдом (об'єктом, найближчим до центру кластера). PAM є більш стійким до шуму та викидів, але має вищу обчислювальну складність порівняно з K-means.

CLARA (Clustering LARge Application) призначений для кластеризації великих наборів даних шляхом застосування PAM до кількох вибірок даних. Його ефективність залежить від розміру вибірки.

CLARANS (Clustering Large Applications based upon RANdomized Search) - покращена варіація k-медоїдного алгоритму, яка поєднує вибірку та рандомізований пошук на графі станів (де кожен вузол — набір k медоїдів).

Це дозволяє уникати фіксованої вибірки (на відміну від CLARA) і є менш чутливим до локальних оптимумів.

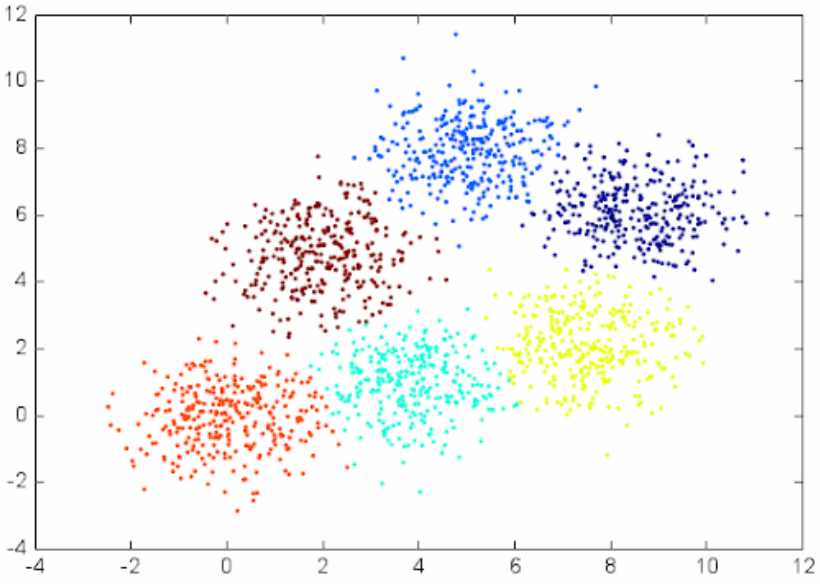

Рис. 2.4. Приклад CLARANS

Алгоритми на основі розбиття ефективні для кластерів опуклої форми, вимагають попереднього завдання кількості кластерів (k), не працюють ефективно з зашумленими даними і зазвичай застосовні для наборів даних середнього розміру.

2.4.2. Ієрархічна кластеризація

Ієрархічні методи будують деревоподібну структуру (дендрограму) кластерів шляхом рекурсивного об'єднання (знизу вгору) або поділу (зверху вниз) даних.

Агломеративна (AGNES, знизу вгору) - починає з того, що кожен об'єкт є кластером, і послідовно їх об'єднує. Дивізівна (DIANA, зверху вниз) - починає з одного кластера, що містить усі об'єкти, і послідовно його ділить.

Об'єднання/поділ базується на мірі подібності:

- Однозв'язкова (Single-linkage) - відстань визначається найкоротшою відстанню між будь-якими двома об'єктами різних кластерів (чутлива до "мостів" між кластерами).

- Повнозв'язкова (Complete-linkage) - відстань визначається найдовшою відстанню між будь-якими двома об'єктами (зазвичай створює більш компактні кластери).

- Середньозв'язкова (Average-linkage) - відстань — це середня відстань між усіма парами об'єктів різних кластерів.

CURE (Clustering Using REpresentatives) - агломеративна кластеризація, яка використовує вибірку та розбиття для роботи з великими даними. Кластери представлені добре розподіленими точками, які стискаються до центроїда з параметром α для кращого визначення форми кластера.

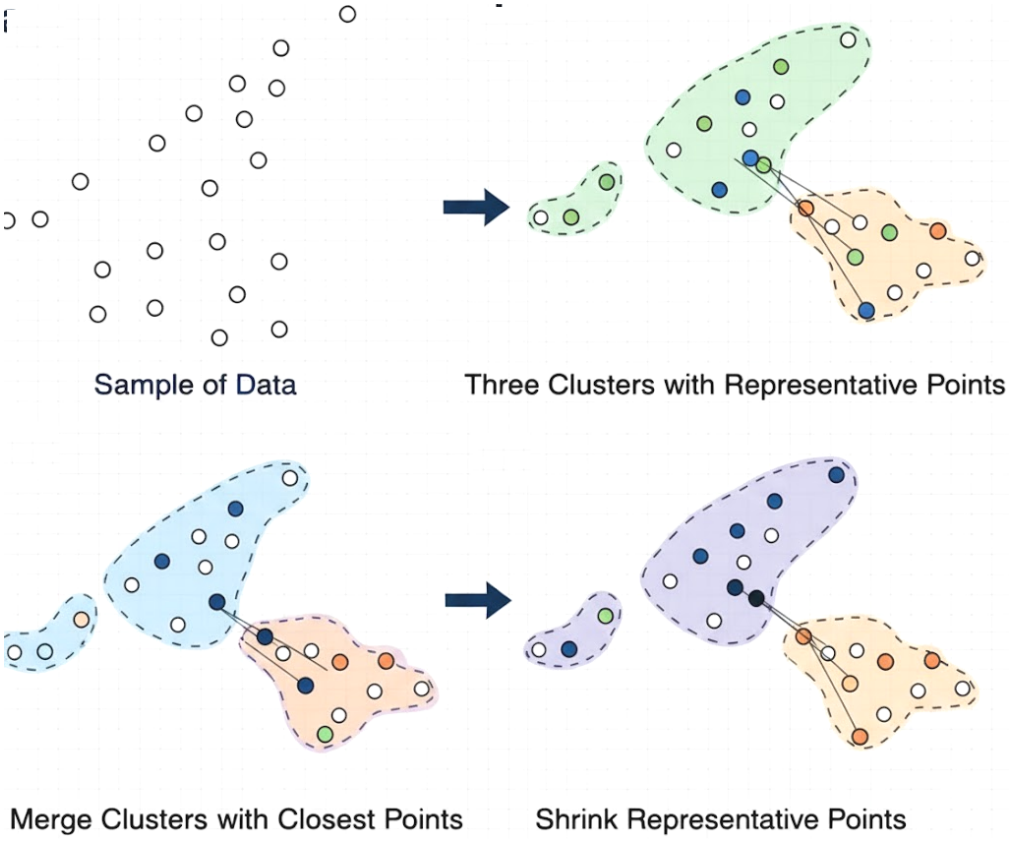

Рис. 2.5. агломеративна кластеризація CURE

BIRCH (Balanced Iterative Reducing and Clustering Using Hierarchies) - дивізівна кластеризація, що будує CF-дерево (Clustering Feature Tree) інкрементним та динамічним способом за один прохід даних. Ефективний,

але чутливий до порядку вхідних даних і може погано працювати з неопуклими кластерами.

Chameleon - поєднує розбиття графів на основі k-найближчих сусідів (зберігаючи щільність регіону як вагу ребра) з агломеративною кластеризацією. Об'єднання підкластерів ґрунтується на відносній взаємозв'язності та близькості, що дозволяє точно знаходити вкладені та увігнуті кластери.

Обмеження цих методів полягає в тому, що рішення про об'єднання/поділ є незворотнім. Методи чутливі до шуму та викидів і мають проблеми з кластерами різного розміру або неопуклої форми.

2.4.3.Кластеризація на основі щільності

Ці алгоритми розділяють дані на основі щільності, зв'язності та меж, використовуючи концепцію околиці точки. Вони здатні виявляти кластери довільних форм та ефективно справлятися з шумом та викидами.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) - базується на понятті, що щільність точок у кластері значно вища, ніж поза ним.

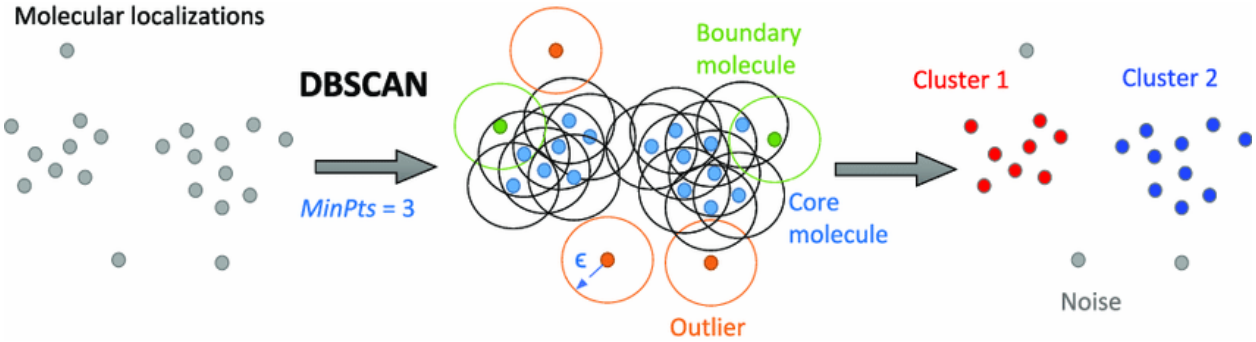

Рис. 2.6. Принцип роботи DBSCAN алгоритму

Використовує Eps (ϵ, радіус околиці) та MinPts (мінімальна кількість точок в околиці). Обчислювальна складність O(n2) (через обчислення

матриці відстаней), але може бути знижена до O(nlogn) з використанням індексів. Недоліком є те, що ефективний лише для наборів даних з рівномірною щільністю.

OPTICS (Ordering Points To Identify the Clustering Structure) - розроблений для вирішення проблеми різної щільності в даних. Він упорядковує точки даних таким чином, що просторово близькі точки стають сусідами, і зберігає спеціальну "відстань досяжності" для кожної точки, яка відображає щільність, необхідну для формування кластера.

2.5. Масштабованість алгоритмів кластеризації на основі паралельних та розподілених парадигм

Швидке та безперервне зростання розмірів наборів даних обумовлює необхідність паралельного інтелектуального аналізу даних (Parallel Data Mining) як природного та економічно ефективного підходу для вирішення проблеми масштабованості. Хоча значна частина наукових зусиль була спрямована на паралелізацію існуючих послідовних алгоритмів, цей підхід має значні обмеження.

Існуючі послідовні алгоритми кластеризації (наприклад, K-means, CURE, BIRCH, Chameleon) ефективні для малих та середніх наборів даних, але не можуть бути застосовані до масивних обсягів даних (massively large datasets), які перевищують доступну дискову чи оперативну пам'ять. Основними перешкодами є висока обчислювальна складність та значні вимоги до пам'яті під час виконання, що призводить до неефективного масштабування. Крім того, деякі послідовні методи не здатні ефективно обробляти гетерогенні дані, зібрані з різнорідних джерел.

Просторові дані демонструють характеристики, спільні з великими даними: великі розміри, висока швидкість надходження (velocity) та складна внутрішня інформація. Таким чином, просторовий аналіз не лише успадковує виклики, притаманні обробці будь-яких надзвичайно великих даних, але й

додає додатковий рівень складності через необхідність врахування просторових вимірів. Це створює нагальну потребу у розробці нових та високоефективних алгоритмів просторової кластеризації [74].

Проектування та розробка наступного покоління великомасштабних алгоритмів кластеризації вимагає вирішення наступних ключових проблем:

1. Прокляття розмірності (Curse of Dimensionality)

Проблема: сучасні набори даних часто страждають від надмірної кількості атрибутів.

Альтернатива: кластеризація підпростору є підходом, спрямованим на знаходження кластерів у різних підмножинах атрибутів (підпросторах) у межах набору даних. Прикладами є алгоритми Clique та MAFYA.

2. Розмір даних та обмеження пам'яті

Проблема: більшість послідовних алгоритмів кластеризації покладаються на структури даних у пам'яті (in-memory structures) для зберігання проміжних результатів. В умовах обмеженої системної пам'яті (кілька ГБ на стандартних комп'ютерах), обробка дуже великих наборів даних є надзвичайно дорогою.

Наслідки: ітеративні алгоритми, що вимагають багаторазового сканування даних, не масштабуються. Це змушує або записувати проміжні результати на диск (що збільшує I/O витрати), або застосовувати розбиття наборів даних на менші, оброблювані частини.

3. Розподіл та локалізація даних

Проблема: великомасштабні набори даних є розподіленими за своєю природою і можуть належати різним організаціям, що вимагає децентралізованих підходів.

Типи розділення:

- Горизонтальне розділення: різні сайти мають різні транзакції (однорідні дані).

- Вертикальне розділення: різні сайти мають різні атрибути (гетерогенні дані).

Більшість поточних досліджень зосереджені на горизонтальному розділенні.

4. Тип даних та гетерогенність

Більшість досліджень з кластеризації традиційно орієнтовані на багатовимірні та однорідні набори даних.

Критична потреба: розробка підтримки для просторово-часових та мультимедійних даних, а також ефективне подолання гетерогенності атрибутів.

Для вирішення вищезазначених проблем, особливо для обробки наборів даних, що перевищують можливості одного сховища, паралельні та розподілені підходи стають обов'язковими. Це вимагає переформулювання та перепроектування існуючих підходів до інтелектуального аналізу з урахуванням:

- Масивних обсягів даних.
- Гетерогенності даних.
- Розподіленого зберігання даних.
- Обчислювальної складності.

Техніки та парадигми паралельних та розподілених обчислень (наприклад, MapReduce, хмарні платформи) становлять основу для будь-якого запропонованого масштабованого рішення і, як очікується, будуть зростаючою областю досліджень у найближчому майбутньому.

Висновки до розділу

Другий розділ був присвячений аналізу сучасних технік інтелектуального аналізу даних та їхньому застосуванню в умовах розподілених обчислювальних середовищ. Показано, що класичні методи ІАД, такі як класифікація, кластеризація, вилучення асоціативних правил і прогнозування, вимагають суттєвої адаптації для роботи з даними у

масштабах Big Data. Особливу увагу зосереджено на просторово-прив’язаних даних, для яких характерні складні топологічні взаємозв’язки та необхідність врахування просторової залежності. Проаналізовано специфіку просторового датамайнінгу та виділено групи задач, що найбільш чутливі до розподіленої природи обробки, зокрема просторові асоціації, класифікація та прогнозування. Дослідження показало, що традиційні алгоритми кластеризації демонструють обмежену масштабованість через залежність від глобальних структур даних і високі комунікаційні витрати. Розроблена таксономія технік просторового датамайнінгу дозволила структурувати існуючі методи й виявити їх сильні та слабкі сторони з позицій розподіленого виконання.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОЦЕСІВ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ ЗАСОБАМИ РОЗПОДІЛЕНИХ ДИНАМІЧНИХ ФРЕЙМВОРКІВ

3.1. Представлення архітектури розподіленого фреймворку кластеризації для надвеликих наборів даних

У цьому розділі представлено розподілений підхід до кластеризації (Distributed Dynamic Clustering, DDC), розроблений для ефективної обробки надзвичайно великих наборів даних та подолання їхніх викликів. Хоча методологія DDC є узагальненою, вона валідується та реалізується у даному дослідженні на просторових наборах даних, використовуючи існуючі платформи розподілених обчислень, зокрема MapReduce. Запропонований підхід також є вирішенням деяких обмежень популярних послідовних алгоритмів кластеризації (наприклад, K-means, DBSCAN).

Далі буде представлено методологію проектування, детальний покроковий опис підходу, аналіз його загальної обчислювальної складності та порівняння з існуючими алгоритмами просторового дата майнінгу.

Проектування ефективних алгоритмів інтелектуального аналізу Великих Даних обумовлене необхідністю вирішення наступних критичних факторів (викликів):

1. Розмір вхідного набору даних (масштабованість)

Здатність алгоритму підтримувати продуктивність (час обробки) та точність результатів при експоненціальному зростанні обсягу даних.

2. Гетерогенність.

Дані, зібрані з різнорідних джерел, є гетерогенними, містять різні рівні шуму та можуть вимагати різних аналітичних підходів.

3. Конфіденційність.

Джерела даних часто належать різним організаціям, які не можуть ділитися сирими даними через міркування конфіденційності клієнтів або конкуренції.

4. Швидкість

Вибрана аналітична техніка повинна мати змогу обробляти дані по мірі їх надходження без погіршення продуктивності або введення затримок.

Один із найбільш ефективних методів подолання цих викликів полягає в аналізі даних на місці їх виникнення (at the source). Це передбачає локальний аналіз у кожному джерелі даних (вузлі) з подальшою інтеграцією знань.

Таблиця 3.1.

Переваги локального аналізу

Виклик	**Перевага локального аналізу**
Масштабованість	Загальний набір даних розподіляється, і обчислювальна потужність кожного вузла використовується для паралельної обробки його частки даних. Усувається необхідність переміщення всього обсягу даних до центрального вузла.
Гетерогенність	Кожне джерело може застосовувати оптимальний локальний алгоритм кластеризації, що найкраще відповідає характеристикам його даних. Результати потребують лише уніфікованого формату представлення.
Конфіденційність	Зберігається, оскільки аналізуються локальні дані, а поширюються лише результати локального майнінгу, що значно спрощує контроль за чутливістю інформації.
Швидкість	Локальна обробка легше справляється з локальною швидкістю надходження даних, ніж із загальною агрегованою швидкістю.
Продуктивність	Підхід комбінує потужність паралельних і розподілених систем для прискореної обробки.

Інтелектуальний аналіз даних у джерелі вирішує багато проблем (наприклад, висока пропускна здатність мережі, необхідна для централізованої обробки). Однак, локальні результати не завжди адекватно відображають глобальні тенденції в усьому масиві даних.

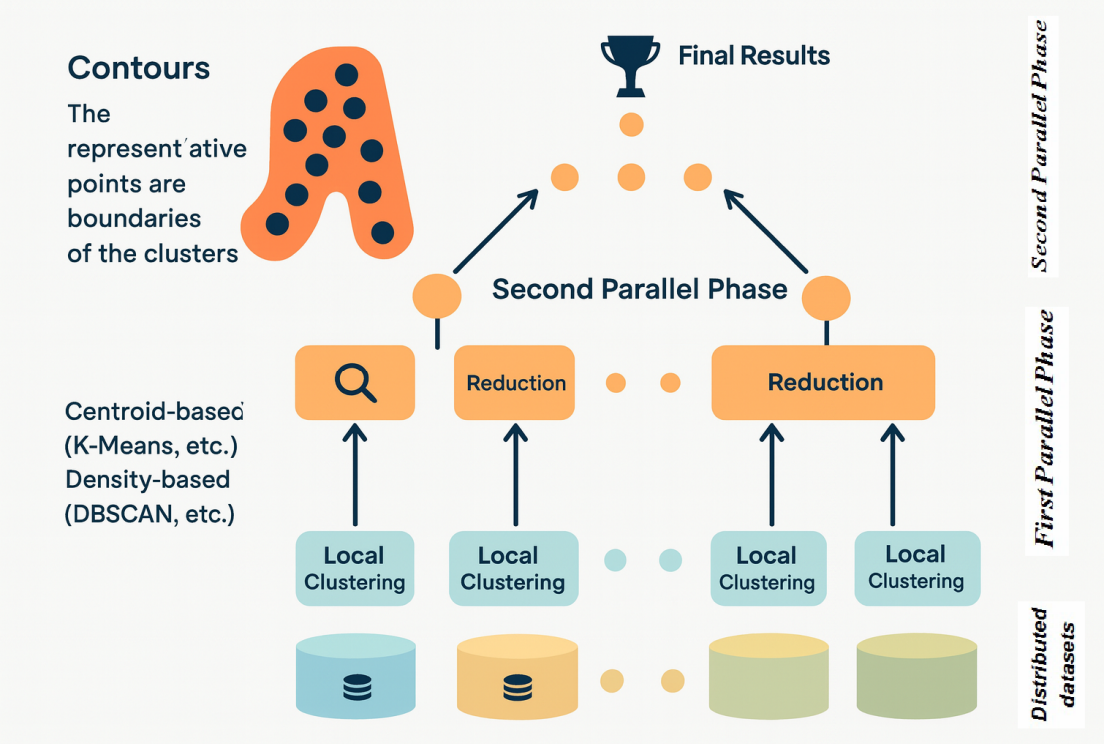

Рис. 3.1. Архітектура фреймворку

Отже, фреймворк DDC має двофазну архітектуру, спрямовану на об'єднання локальних знань для генерації глобальних інсайтів:

1. Фаза локального майнінгу - аналіз даних, що зберігаються в кожному локальному джерелі (вузлі), з метою отримання локальних кластерів.

2. Фаза глобальної генерації знань - агрегація та об'єднання локальних кластерних результатів для створення глобальних кластерів, які відображають загальну структуру всього набору даних.

Далі буде представлено детальний опис цих двох фаз в рамках фреймворку розподіленої динамічної кластеризації.

3.2. Опис фази локального розподіленого аналізу даних

Ця фаза передбачає, що загальний масив даних розділений на P партицій (або підмножин). Оптимальне значення P, залежне від розміру вхідних даних, буде визначено у наступному розділі. У випадку, коли цільова

обчислювальна платформа є розподіленою, P має відповідати кількості обробних вузлів у системі.

Для кожної партиції p_i, кластеризація її даних здійснюється незалежно, що призводить до отримання C_i кластерів. При розгортанні на паралельній або розподіленій системі з P обробними вузлами, Фаза I є чисто паралельною, оскільки кожен вузол виконує алгоритм кластеризації на своїй партиції даних незалежно від інших.

Характеристики локальної кластеризації

1. Алгоритмічна гнучкість.

Для локальної кластеризації може використовуватися будь-який алгоритм кластеризації. Це може бути центроїдний алгоритм (наприклад, K-means), алгоритм на основі щільності (наприклад, DBSCAN) або навіть комбінація різних алгоритмів. У подальшому розділі будуть розглянуті приклади, де локальна кластеризація виконується за допомогою K-means або DBSCAN для демонстрації ефективності підходу DDC (Distributed Dynamic Clustering).

2. Локальні моделі.

Отримані на кожному вузлі кластери називаються локальними кластерами або локальними моделями.

3. Залежність від техніки.

Локальні кластери значною мірою залежать від обраної техніки кластеризації, застосованої на відповідних обробних вузлах. Наприклад, для просторових наборів даних форма кластера зазвичай диктується використаним методом. Це не створює значних проблем на цій фазі, оскільки точність кластера впливає лише на локальні результати даного вузла.

Обчислювальна ефективність та асинхронність:

- Час відповіді кожного обробного вузла залежить від розміру локального набору даних та складності локально використаного алгоритму кластеризації.

- Якщо розмір партиції не є пропорційним обчислювальній потужності відповідного обробного вузла, виникає ризик очікування найповільнішого вузла перед початком фази II - глобальної генерації знань.

Ці сценарії будуть ретельно досліджені пізніше, зокрема, буде вивчатися перекриття (overlapping) фази I з фазою II для мінімізації впливу найповільнішого процесу на загальну продуктивність.

3.3. Представлення фази агрегації та генерації глобальних моделей

Фаза агрегації передбачає збір локальних моделей з кожного обробного вузла та їх передачу лідерам (leaders) — спеціальним вузлам у системі, обраним на основі певних характеристик, таких як обчислювальна потужність, пропускна здатність та зв'язність.

Лідер відповідає за агрегацію (злиття) локальних кластерів, які перекриваються (overlap).

Для здійснення цієї фази необхідний обмін локальними кластерами. Оскільки загальний обсяг даних є надзвичайно великим, пряма передача всіх оригінальних даних призведе до швидкого насичення мережі та значних комунікаційних витрат.

Ключова ідея підходу системи - передача лише репрезентативних даних кластера, що становить лише 1%–2% від загального розміру даних. Репрезентативні дані включають:

- Внутрішні репрезентативні точки (для центроїдних методів).

- Граничні точки кластера (boundary points).

Найкращий спосіб репрезентації просторового кластера — це використання його форми та щільності. Форма кластера репрезентується його граничними точками (контуром).

Існуючі методи редукції даних часто зосереджуються лише на зменшенні розміру сховища без урахування знань, що містяться в даних [20, 21].

Для ефективного вилучення контурів кластера ми використовуємо два алгоритми:

- α-shape алгоритм - базується на тріангуляції для генерації границь кластера.

- алгоритм збалансованого вектора (balance vector algorithm) - базується на кластеризації за щільністю.

Обидва алгоритми є ефективними для побудови неопуклих (non-convex) границь і здатні точно характеризувати форму широкого діапазону розподілів точок зі складностю $O(n \log n)$.

Глобальні моделі (шаблони) генеруються під час фази II, яка виконується розподілено. На відміну від фази I, вузли повинні обмінюватися локальними або проміжними кластерами. Процес агрегації складається з двох основних кроків, які повторюються доти, доки не будуть згенеровані всі глобальні кластери:

- Кожен лідер збирає локальні кластери своїх сусідів.

- Лідери зливають (merge) локальні кластери, використовуючи техніку накладання (overlay technique).

Процес злиття кластерів триває до досягнення кореневого вузла (root node), який міститиме глобальні кластери.

Для зменшення накладних витрат у системі обмінюються лише контури (границі) кластерів, а не повні кластери, між вузлами системи. Контури кластерів формують новий, значно менший набір даних, який передається лідерам, слідуючи деревовидній топології.

Основні кроки підходу:

1. Кожен вузол має фрагмент даних, що представляє частину сцени або загального набору даних.

2. Кожен листовий вузол (leaf node) виконує локальний алгоритм кластеризації зі своїми вхідними параметрами (фаза I).

3. Кожен вузол обмінюється своїми контурами кластерів із сусідами для формування більших кластерів за допомогою техніки накладання.

4. Лідери містять проміжні результати своїх груп.

5. Повторення кроків 3 і 4 доти, доки не будуть згенеровані всі глобальні кластери (фаза II, досягнення кореневого вузла).

В лістингу 3.1. подано псевдокод фази I: локальна кластеризація.

Лістинг 3.1. Псевдокод фази I

```
1 Initialization
2 Node_i ∈ N, N: The total nodes in the system.
input: X_i: Dataset Fragment, Params_i: Input parameters for the local clustering:
output: C_i: Cluster's contours of Node_i

3 foreach Node_i do
4     L_i = Local_Clustering(X_i, Params_i);
      // Node_i executes a clustering algorithm locally.
5     C_i = Contour(L_i);
      // Node_i executes a contour algorithm locally.
6 return (C_i);
```

Лістинг 3.2. Python функція реалізації локальної кластеризації

```
:param Nodes: Список усіх вузлів у системі (N).
:param Input_Data_Fragments: Словник або список фрагментів даних {Node_i: X_i}.
:param Local_Params: Словник або список параметрів кластеризації {Node_i: Params_i}.
:return: Словник контурів кластерів {Node_i: C_i}.
"""

# 1. Ініціалізація
N = len(Nodes)
Cluster_Contours = {}

# 3. Ітерація по кожному вузлу
for Node_i in Nodes:
    X_i = Input_Data_Fragments[Node_i]
    Params_i = Local_Params[Node_i]

    # 4. Виконання локальної кластеризації
    # L_i: Набір локальних кластерів, отриманих на Node_i
    L_i = Local_Clustering(X_i, Params_i)

    # 5. Вилучення контуру кластера
    # C_i: Контури (границі) кластерів для Node_i
    C_i = Contour_Extraction(L_i)

    Cluster_Contours[Node_i] = C_i

# 6. Повернення контурів кластерів
return Cluster_Contours
```

Ця фаза виконується розподілено у деревовидній топології з обміном контурами між сусідніми групами, очолюваними лідерами (лістинг 3.3).

Лістинг 3.3. Алгоритм злиття (фаза II)

```
input: D: Tree degree, C_i: Local cluster's contours generated by Node_i
        in the phase 1
output: C_Gk,level: Global Cluster's contours (global results, level=0)

1 repeat
2     level = treeHeight;
3     Node_i joins a group G_K,level of D elements;
      // Node_i joins its neighbourhood
4     Node_j = ElectLeaderNode(G_K,level);
      // Node_j is the leader of the group G

      // In parallel
5     foreach Node_i ∈ G_K,level do
6         if (i <> j) then
7             Send (C_i, Node_j);
            // Each node sends its contours to the leader node (Node_j)
            // in the same neighbourhood group
8         else
9             Recv (C' <- {C_l});
            // If the node is the leader, it will receive the other nodes'
            // contours (C_l) in the same neighbourhood group
10        G_K,level = Merge (C_i, C');
          // Merge the overlapping contours
11    level - -;
12 until (level == 0);
13 return (C_Gk,0);
```

3.4. Методи вилучення репрезентативних границь простору

Найважливішою особливістю запропонованого підходу розподіленої динамічної кластеризації є те, що на фазі агрегації відбувається обмін лише репрезентативними точками, які формують границі (контури) локальних кластерів. Для вилучення цих граничних точок у даній роботі використано два ефективні алгоритми: α-shape алгоритм та алгоритм збалансованого вектора.

3.4.1. Опис роботи α-shape алгоритму

α-shape алгоритм — це простий, гнучкий та ефективний метод для побудови характеристичної форми (characteristic shape) набору вхідних точок

на площині. Він може генерувати як опуклі (convex), так і неопуклі (non-convex, concave) прості полігони.

Основні концепції:

1. Опукла та неопукла форма:

- Опуклий полігон dизначається його крайніми зовнішніми точками і повинен задовольняти умову, що всі його внутрішні кути менші за 180◦.

- Неопуклий (увігнутий) полігон - zкщо хоча б один внутрішній кут більший або дорівнює 180◦.

Рисунок 3.2 демонструє приклад як опуклої, так і неопуклої форми для набору точок даних.

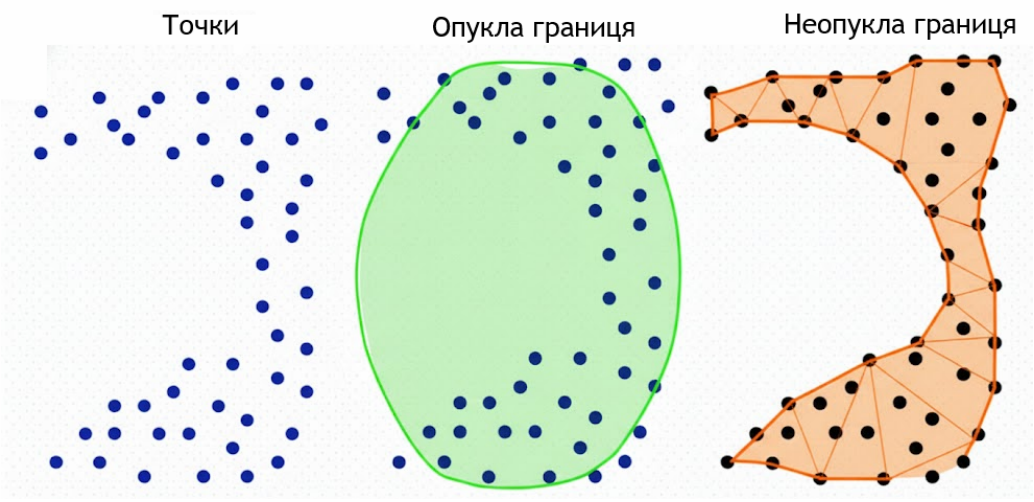

Рис. 3.2. Опукла та неопукла форми

Тріангуляція Делоне:

- Для набору точок P на площині, тріангуляція Делоне (DT(P)) — це така тріангуляція, що жодна точка з P не знаходиться всередині описового кола будь-якого трикутника в DT(P). Тріангуляція Делоне максимізує мінімальний кут усіх трикутників, уникаючи тонких ("сріблястих") трикутників.

- Умова Делоне - lля пари суміжних трикутників сума кутів, протилежних спільному ребру (α1+α2), має бути меншою або дорівнювати

180◦. Якщо умова не виконується, ребро перевертається (flipped) для задоволення критерію .

Алгоритм α-shape використовує єдиний нормалізований параметр довжини (l), який контролює форму отриманого полігона, варіюючись від опуклої оболонки (convex hull) до форми з мінімальною площею.

1. Побудова тріангуляції Делоне для набору вхідних точок P.

2. Послідовне видалення найдовшого зовнішнього ребра з тріангуляції за таких умов:

- Довжина ребра більша за заданий параметр l.

- Зовнішні ребра отриманої тріангуляції формують границю простого полігона (кластера точок).

3. Повторення кроку 2, доки існують ребра для видалення.

4. Повернення полігона (контуру), сформованого зовнішніми ребрами кінцевої тріангуляції.

Щодо ефективності, то α-shape алгоритм є оптимальним за часом, вимагаючи лише $O(n \log n)$ обчислювального часу.

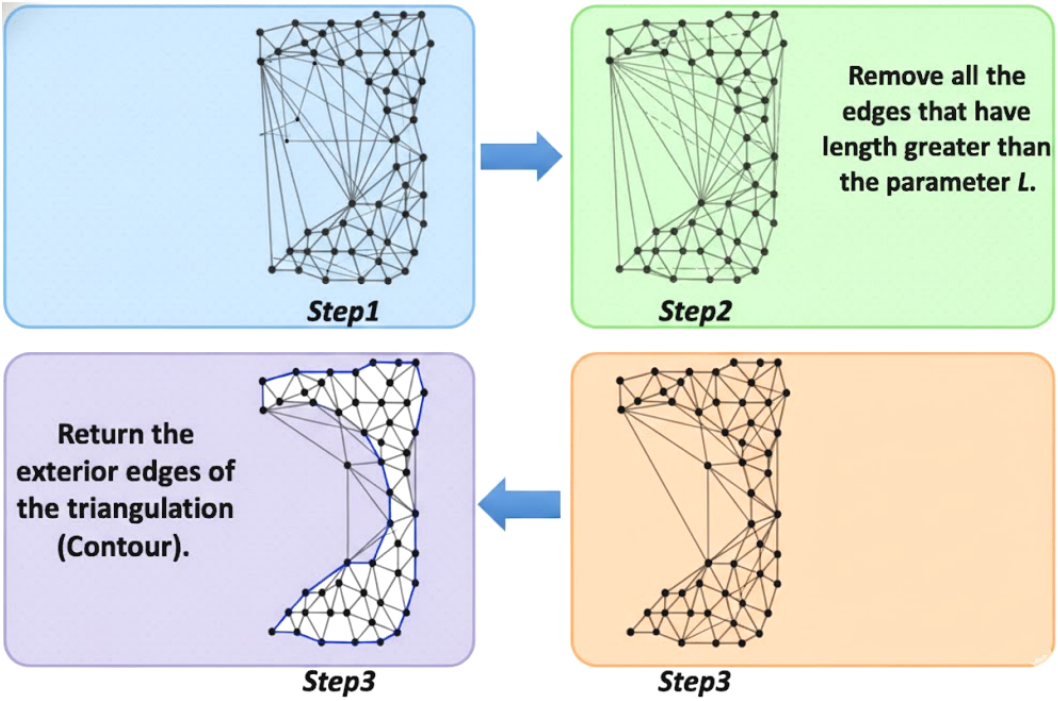

Рис. 3.4. Ілюстрація α-shape алгоритму

3.4.2. Алгоритм збалансованого вектора

Алгоритм збалансованого вектора використовується для виявлення граничних точок локальних кластерів, отриманих методами, орієнтованими на щільність (density-based clustering).

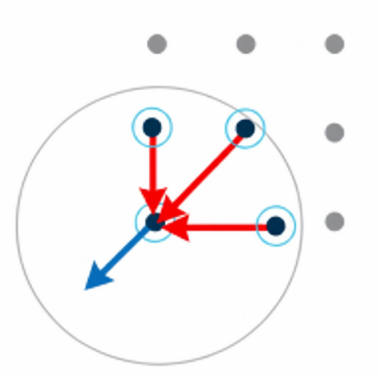

Рис. 3.5. Приклад збалансованого вектора

Розглянемо ключові концепції.

1. Для точки p у кластері C, її окіл визначається як набір точок pi∈C, відстань до яких dist(p,pi) менша або дорівнює радіусу ϵ.

$$N_C(p) = \{p_i \in C \mid \text{dist}(p, p_i) \leq \epsilon\}$$

2. Вектор зміщення від точок pi∈NC(p) до точки p визначається як сума векторів від pi до p.

$$V = \sum_{p_i \in N_C(p)} (p - p_i)$$

Цей вектор V вказує у напрямку найменшої щільності в околиці точки p.

3. Збалансований вектор відносно точки p — це одиничний вектор у напрямку вектора зміщення V.

$$b_p = \begin{cases} \frac{1}{\|V_p\|} V_p & \text{якщо } \|V_p\| > 0 \\ 0 & \text{інакше} \end{cases}$$

Вектор b_p вказує на найменш щільну область околиці p (рис. 3.4).

4. Гранична точка повинна мати порожню область у напрямку збалансованого вектора. Ця область перевіряється як перетин гіперкону нескінченної висоти, вершини, осі та апертури ν (заданий кут). Формально гранична точка описується булевим предикатом:

$$\text{Boundary}(p) = \begin{cases} \text{true} & \text{якщо } (\forall q \in N_\epsilon(p), (q - p) \cdot b < \cos(\nu)) \\ \text{false} & \text{інакше} \end{cases}$$

Границі кластера B_C визначаються як набір усіх граничних точок у C:

$$B_C = \{p \in C : \text{Boundary}(p) \text{ is true}\}$$

Лістинг 3.4. Алгоритм виявлення границь

```
input: Cluster C, Set of balance vectors (b_pi), Parameter nu (angle).
output: Boundary points BC of cluster C

1 BC ← C;  // Ініціалізуємо граничні точки всіма точками кластера
2 for every point p ∈ BC do
3     for every point q ∈ NC(p) do
4         if (q - p) · b_p >= cos(nu) then
5             Discard p from BC; // Якщо знайдено точку в напрямку баланс-вектора
6             Break;
7 return BC;
```

Обчислювальна складність алгоритму виявлення границь становить *O(n log n)*.

Локальна модель L_i для вузла i складається з об'єднання границь B_{cj} кожного кластера cj∈Ci та набору внутрішніх репрезентативних точок P_i.

$$L_i = \bigcup_{j=1}^{n} B_{c_j} \cup P_i$$

3.5. Опис фреймворку Apache Hadoop для реалізація розподіленої динамічної кластеризації

Метою даного розділу є демонстрація реалізації підходу розподіленої динамічної кластеризації з використанням парадигми програмування MapReduce. Це дослідження має на меті показати, що розподілена кластеризація є гнучкою для виконання в різних паралельних розподілених середовищах.

MapReduce — це модель, розроблена для обробки надзвичайно великих розподілених наборів даних у великомасштабних розподілених системах, таких як інфраструктура хмарних обчислень. Реалізація розподіленої динамічної кластеризації за допомогою цієї парадигми дозволить продемонструвати поведінку та можливості кластеризації в роботі з дуже великими масивами даних та забезпечити значне скорочення часу виконання без втрати якості кінцевих результатів.

Apache Hadoop є найпопулярнішою паралельною та розподіленою моделлю обробки великих даних, а MapReduce є її ядром. MapReduce — це парадигма програмування, яка забезпечує масивну масштабованість на сотнях або тисячах серверів у кластері Hadoop.

MapReduce є фреймворком, здатним обробляти великомасштабні набори даних шляхом використання паралелізму між кластерами обчислювальних вузлів. MapReduce набув популярності завдяки своїй простоті, гнучкості, відмовостійкості та масштабованості.

Численні алгоритми інтелектуального аналізу даних були реалізовані на Hadoop MapReduce для покращення та прискорення їхньої продуктивності. Це призвело до пропозиції дослідниками рішень, заснованих на MapReduce та моделях сервісів хмарних обчислень.

Кластеризація є однією з технік дата майнінгу, яка активно адаптувала MapReduce, і багато досліджень зосереджено на використанні цієї парадигми для кластеризації великих даних. Зазвичай це реалізації існуючих алгоритмів кластеризації на Hadoop MapReduce для конкретних прикладних завдань.

Apache Hadoop — це платформа з відкритим кодом, розроблена для розподіленої обробки великих наборів даних на великомасштабних та динамічних кластерах. Hadoop забезпечує високу продуктивність, швидкість та різноманітність обробки, перетворюючи звичайне апаратне забезпечення на сервіс, що підтримує розподілене зберігання та обробку, а також сприяє розробці розподілених застосунків. Hadoop також керує апаратними збоями на рівні застосунку.

Ключові характеристики Hadoop

1. Масштабованість (Scalability).

Програма, розроблена для однієї машини, може легко масштабуватися для роботи на тисячах машин; потужність збільшується шляхом додавання додаткових вузлів.

2. Стійкість до збоїв (Fault Tolerance) - ключова перевага Hadoop. Дані, що надсилаються на окремий вузол, реплікуються на інші вузли в кластері. У разі збою одного вузла, доступна інша копія даних.

3. Швидкість.

Унікальний метод зберігання ґрунтується на розподіленій файловій системі, яка забезпечує локалізацію даних (data locality) — інструменти обробки часто розташовані на тих самих серверах, де зберігаються дані, що значно прискорює обробку. Hadoop здатний ефективно обробляти терабайти даних за хвилини, а петабайти — за години.

4. Простота.

Надає прості API, залишаючись при цьому потужною платформою, здатною працювати з величезними (петабайтними) обсягами даних.

Hadoop складається з двох основних компонентів: HDFS (Hadoop Distributed File System), що є частиною для даних, та MapReduce, що є

обробною частиною. У кластері Hadoop, що складається з сотень або тисяч машин, файли розбиваються на блоки, які можуть оброблятися одночасно (паралельно) замість послідовної обробки.

3.5.1. Розподілена файлова система Hadoop (HDFS)

HDFS функціонує як файлова система, розташована поверх існуючої локальної файлової системи кожного вузла. Вона оптимізована для роботи з великими файлами і використовує концепцію блоків для зберігання даних.

Блоки – це файли які розбиваються на блоки фіксованого розміру (типово 64 МБ). Перевагою блоків є фіксований розмір спрощує розрахунок необхідного місця. Вони дозволяють зберігати файли, розмір яких перевищує ємність одного вузла, і підтримують реплікацію на кількох вузлах для забезпечення доступності даних.

HDFS має два спеціальні типи вузлів: NameNode та DataNode.

1. NameNode (головний вузол). У кластері існує лише один NameNode. Він відповідає за метадані файлів та простір імен файлової системи. NameNode повинен бути найпотужнішим вузлом і мати великий обсяг оперативної пам'яті, оскільки він зберігає всі метадані файлової системи в пам'яті.

Втрата NameNode призводить до втрати всіх даних кластера. Для запобігання цьому можна налаштувати реплікацію NameNode, що називається StandBy NameNode.

2. DataNode (підлеглий вузол). Кластер містить багато DataNodes. Ці вузли зберігають блоки даних. Коли клієнт запитує дані, він звертається до NameNode, щоб дізнатися, який DataNode зберігає необхідні блоки. DataNodes періодично надсилають інформацію про блоки, які вони зберігають.

При надходженні нового файлу, NameNode отримує "запит на створення", обирає DataNode для запису блоків і керує реплікацією даних на інші вузли.

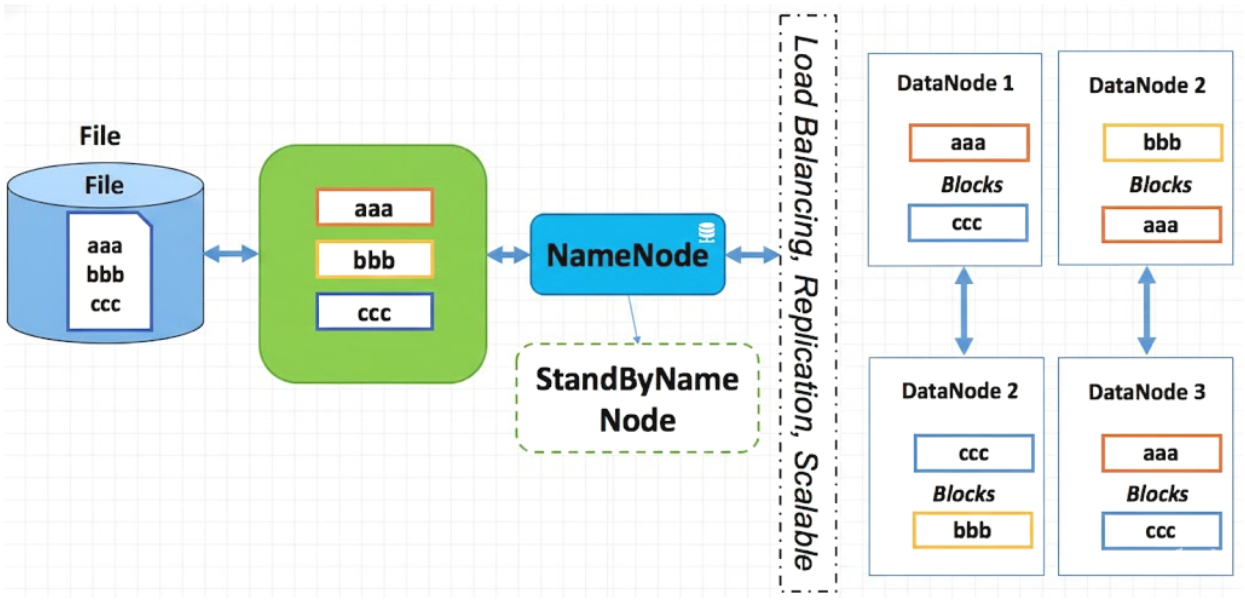

Рис. 3.6. Архітектура HDFS

3.5.2. Парадигма MapReduce

MapReduce — це фреймворк і парадигма програмування, призначена для розробки застосунків, що обробляють великі обсяги даних (терабайти) паралельно на великих кластерах (тисячі вузлів) звичайного апаратного забезпечення з високою надійністю та стійкістю до збоїв.

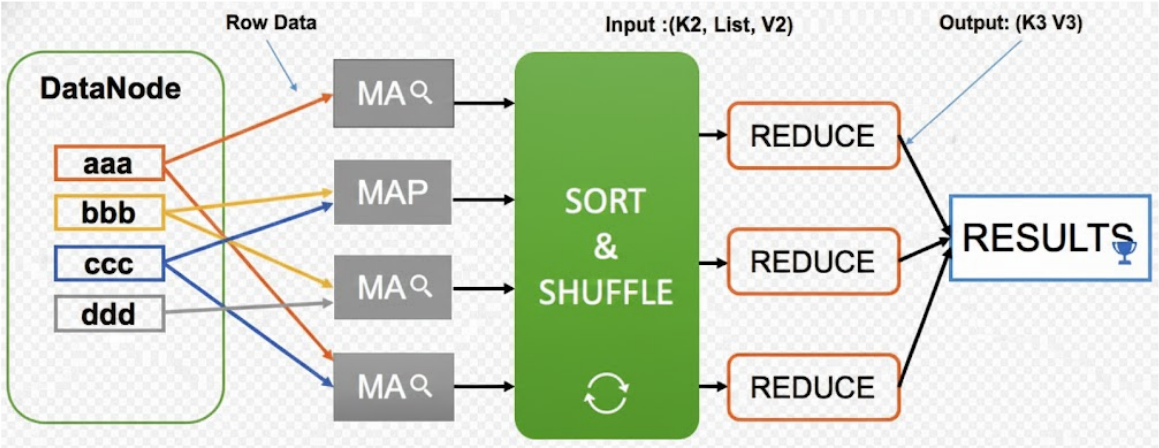

Рис. 3.7. Модель MapReduce

Програма MapReduce складається з двох основних компонентів: завдання Map та завдання Reduce. Вхідні дані подаються у вигляді пар (Ключ, Значення), які обробляються завданнями Map повністю паралельно.

Як правило, користувачеві необхідно написати лише функції map та reduce. Завдання MapReduce складається з трьох фаз:

- Фаза Map: для кожної вхідної пари (k1,v1), функція map генерує один або кілька списків вихідних пар (k2,v2).

- Фаза Shuffle (перемішування): вихідні пари (k2,v2) розподіляються та передаються редукторам.

- Фаза Reduce: пари з однаковим ключем k2 групуються разом як (k2, список(v2)). Функція reduce генерує остаточний список вихідних пар (k3,v3) для кожної групи.

3.6. Адаптація підходу розподіленої динамічної кластеризації до парадигми MapReduce

Як вже було обґрунтовано, підхід розподіленої динамічної кластеризації (РДК) складається з двох ключових фаз:

1) Генерація локальної моделі (виявлення локальних кластерів та вилучення їхніх контурів),

2) Генерація глобальної моделі (об'єднання перекривних локальних контурів).

У рамках фреймворку MapReduce ці дві фази природно відображаються на:

1) Фазу відображення (Map Phase);

2) Фазу зведення (Reduce Phase).

Мапери виконують локальну кластеризацію та вилучення контурів паралельно. Редуктори також виконують фазу злиття (об'єднання контурів) паралельно.

На відміну від фреймворку JADE, що використовувався раніше, ту розподіл даних здійснюється автоматично системою HDFS (Hadoop Distributed File System). Це значно спрощує архітектуру, оскільки логіка DDC не залежить від ручного розподілу даних.

Мапери функціонально незалежні від редукторів, оскільки вхідні дані редукторів (контури локальних кластерів) є виключно результатами маперів. Це забезпечує гнучкість у застосуванні різних алгоритмів локальної кластеризації без впливу на логіку фази зведення.

Лістинг 3.4. Функція відображення (Map Function)

```
Input : X_i: List<lineNo, Point>, Eps_i:R, MinPts_i:N
Output: List<key, List<Contour>>

1 <key, List<Contour>>
2 L_i ← DBSCAN(X_i, Eps_i, MinPts_i); // Local clusters generated by Node_i
3 C_i ← [];
// For each cluster
4 foreach L_i^k ∈ L_i do
5     c^k ← ComputeContour(L_i^k);
6     Append(C_i, c^k) // add contour to the list
7 return <key, C_i>;
```

На фазі відображення кожен мапер реалізує першу фазу DDC (локальну модель):

1. Отримання даних.

Мапер отримує набір даних, розподілений на його обробний вузол HDFS. Дані стандартизуються за допомогою функцій ReadHDFSFile() та StandardiseData().

2. Локальна кластеризація.

Кожен мапер виконує алгоритм DBSCAN з унікальними локальними параметрами (Epsi,MinPtsi).

3. Вилучення контуру.

Після генерації локальних кластерів вузол застосовує алгоритм контуру (наприклад, α-shape або збалансованого вектора) для вилучення репрезентативних граничних точок.

4. Формування вихідних даних.

Мапер призначає згенерованим контурам однаковий ключ (наприклад, ключ = 2) для забезпечення того, що всі контури будуть передані для обробки редукторами.

5. Запис.

Контури записуються у стандартизованому форматі (ключ, координати точок контуру) у файл HDFS для наступного зчитування редуктором.

Фаза зведення реалізує другу фазу (глобальна модель) та відповідає за об'єднання локальних контурів.

Вхідні дані - стандартизовані вихідні дані маперів: пари (ключ,контур). Завдяки використанню єдиного ключа, кожен контур кожного вузла порівнюється з усіма контурами, згенерованими іншими вузлами.

Стратегія злиття - редуктор виконує ітеративне злиття перекривних контурів для побудови більших глобальних кластерів.

Контур cj перекривається з контуром ck тоді і лише тоді, коли многогранник (P(cj)), утворений точками cj, перетинається з многогранником (P(ck)), утвореним точками ck. Злиття двох контурів cj та ck призводить до нового контуру cjk, який є об'єднанням многогранників, що їх утворюють.

Фаза зведення виконується паралельно і розподілено, використовуючи деревовидну структуру. На кожному рівні ієрархії вузли передають свої контури сусідам, які виконують алгоритм злиття. Управління комунікаціями та контролем забезпечується Hadoop.

Лістинг 3.5. Функція зведення (Reduce Function)

```
Input : C:List<key, List<Contour>>
Output: <key, List<Contour>>

1 C_res ← [];
2 foreach <key, C_i> ∈ C do
3     foreach c^j ∈ C_i do
4         merged ← False;
5         foreach c^k ∈ C_res do
6             if Overlap(c^j, c^k) then
7                 c^jk ← Merge(c^k, c^j);
8                 Remove(C_res, c^k); // remove contour from the list
9                 Append(C_res, c^jk); // add contour to the list
10                merged ← True;
11                Break;
12        if not(merged) then
13            Append(C_res, c^j); // add contour to the list
14 return <key, C_res>;
```

3.7. Експериментальна валідація підходу динамічної кластеризації

Для демонстрації функціонування розподіленого підходу РДК на основі MapReduce DDC-DBSCAN-MR наводиться приклад виконання, що включає фази відображення (Map) та зведення (Reduce).

Сценарій 1. Два мапери (2 вузли)

На Рисунку 7.4 представлено фази відображення та зведення з використанням двох маперів. Кожен вузол незалежно виконує алгоритм DBSCAN з локальними параметрами, наприклад, Eps=15.5 та MinPts=20.

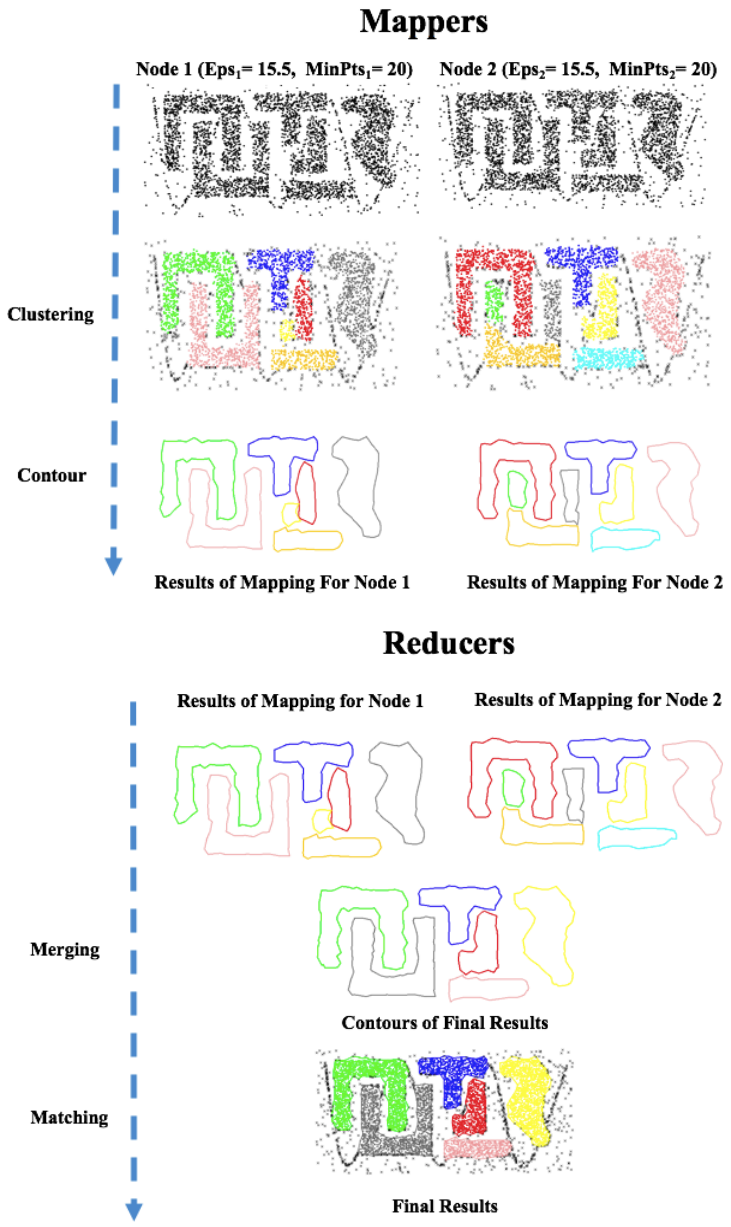

Рис. 3.8. Фази відображення та зведення (з 2 вузлами)

Точні значення локальних параметрів кластеризації не є критичними, оскільки на локальному рівні пріоритетом є якість кінцевих глобальних результатів. Після локальної кластеризації застосовується алгоритм контуру (у даному прикладі — α-shape алгоритм) для вилучення меж кластерів. На фазі зведення відбувається об'єднання перекривних локальних контурів (результатів маперів) для генерації більших, глобальних кластерів.

Сценарій 2. Чотири мапери (4 вузли).

Рисунок 3.9 демонструє результати виконання із п’ятьма маперами. Збільшення кількості маперів не впливає на якість кінцевої кластеризації.

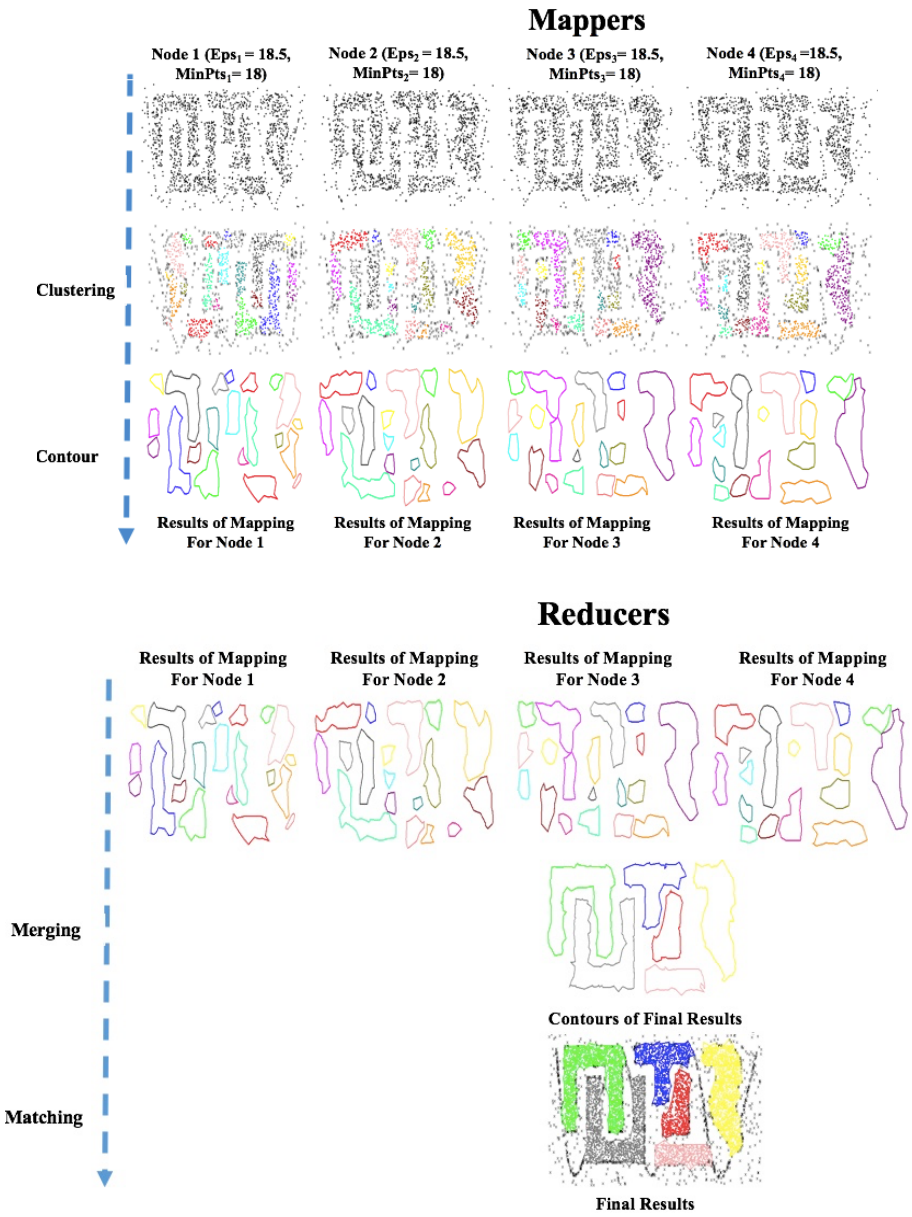

Рис. 3.9. Фаза зведення (4 вузли)

Висновки щодо виконання наступні:

Даний підхід точно повертає правильну кількість кластерів та їхні форми. Підхід нечутливий до способу випадкового розподілу вхідних даних між вузлами HDFS. Підхід продемонстрував стійкість до шуму та викидів. Глобальні кластери були динамічно згенеровані правильно, попри те, що кожен вузол виконував DBSCAN локально з потенційно різними параметрами на зашумлених підмножинах даних.

Реалізація підходу з використанням MapReduce та тестування на реальній системі Hadoop не погіршило якості згенерованих кластерів. Підхід продемонстрував високу ефективність на різних діапазонах наборів даних, досягнувши 100% точності кластеризації.

Ми здійснили реалізацію нашого підходу розподіленої динамічної кластеризації, використовуючи парадигму програмування MapReduce, і протестували його в реальному кластері Hadoop.

Ми провели порівняльні експерименти на одному вузлі (Ubuntu 16.04, 8 ГБ ОЗП, 4 ядра 2.30 ГГц Intel Core i5-6600), щоб вивчити вплив паралелізму в обмеженому апаратному середовищі.

Першим кроком було вивчення масштабованості алгоритму у системі Hadoop з одним вузлом.

1. Великий набір даних (1 мільйон точок).

Ми кластеризували один мільйон точок, змінюючи кількість маперів до 100. Ми помітили, що час виконання алгоритму знижувався в діапазоні між 10 та 50 маперами. Однак, після 50 маперів він стабілізувався і почав зростати близько 70 маперів .

Причина цього полягала в тому, що алгоритм працював на одній машині з лише 4 ядрами, що унеможливлювало чистий паралелізм. Після 60 маперів виграш у часі від розподілу даних для локальної кластеризації DBSCAN вже не був значним. Натомість, час, який Hadoop витрачав на системні операції, такі як читання та запис результатів у HDFS, продовжував зростати, нівелюючи будь-яку вигоду. У результаті загальний час виконання

алгоритму збільшувався. Це також пояснює, чому прискорення при використанні 20 маперів було лише в 3 рази більшим порівняно з 10 маперами, оскільки обчислення не були справді паралельними.

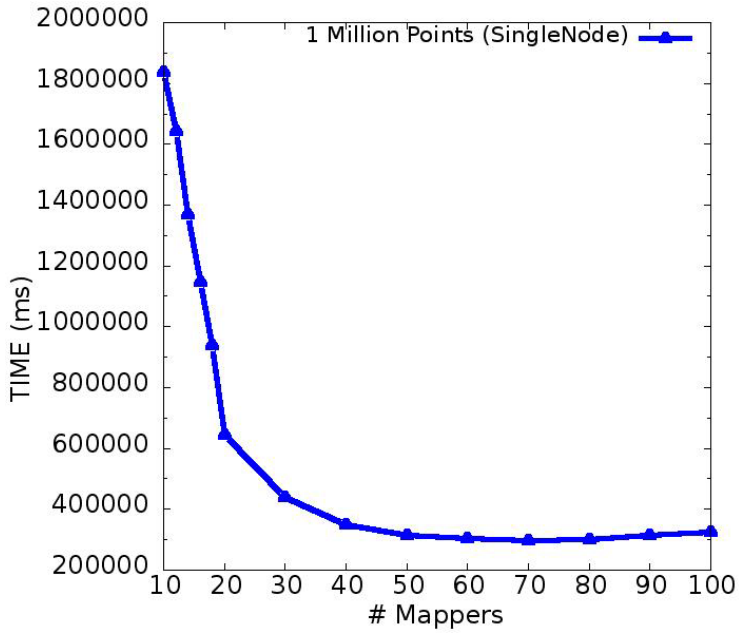

Рис. 3.10. Масштабованість підходу кластеризації на одному вузлі з використанням великого набору даних

2. Малі набори даних (T1,T2,T3)

Ми також запустили DDC на менших наборах даних, щоб оцінити реакцію алгоритму.

Для набору даних T1 (30 000 точок) час виконання зменшився від 1 до 2 маперів, але потім почав лінійно зростати від 2 до 4 маперів і далі. Аналогічна тенденція спостерігалася для наборів даних T2 та T3. Зокрема, для T3 (10 000 точок) час виконання почав зростати одразу. Це пояснюється тим, що ці набори даних є відносно малими, і виграш у часі від розподілу даних для DBSCAN (density-based spatial clustering of applications with noise) був меншим, ніж накладні витрати Hadoop на злиття, комунікацію та операції введення/виведення HDFS.

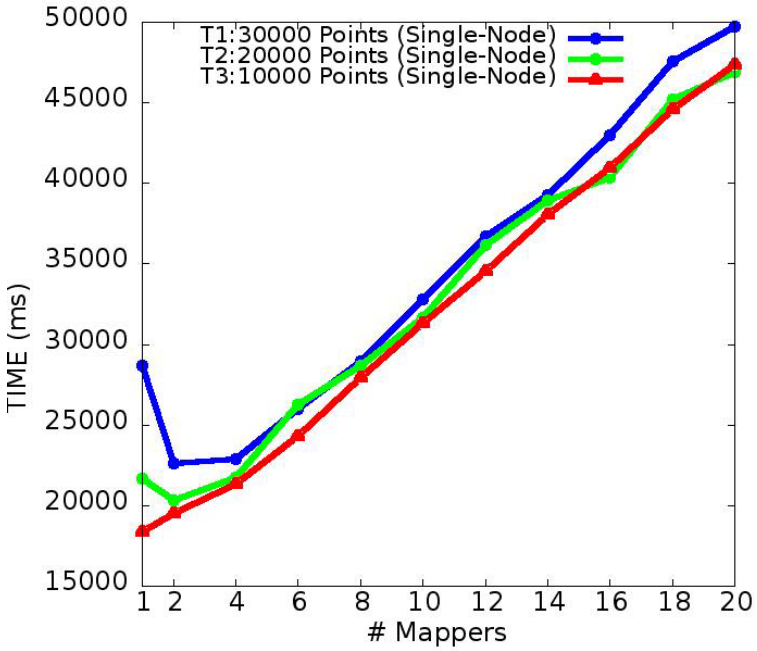

Рис. 3.11. Масштабованість підходу кластеризації на одному вузлі з використанням малих наборів даних

Як видно з наших результатів, MapReduce та Hadoop найкраще працюють із наборами даних середнього та великого розміру. Для малих наборів даних накладні витрати на управління розподіленою системою роблять паралельну обробку неефективною.

Отже, у даному розділі було успішно реалізовано підхід розподіленої динамічної кластеризації з використанням двокрокової парадигми програмування MapReduce. Мета цієї реалізації полягала в емпіричному доведенні адаптивності та придатності підходу кластеризації для фреймворку MapReduce, який є стандартом для хмарних обчислень та масштабування алгоритмів інтелектуального аналізу даних

Результати експериментів чітко продемонстрували, що прискорення та масштабованість підходу є високоефективними при інтеграції з Hadoop та парадигмою MapReduce. Емпіричний аналіз дозволив зробити висновок, що фреймворк Hadoop найбільш придатний для обробки великих і дуже великих наборів даних.

Водночас, було встановлено, що продуктивність системи MapReduce знижується при роботі з малими наборами даних. Це пов'язано з тим, що

системні накладні витрати на управління розподілом, комунікацію та злиття даних у малих масштабах перевищують виграш у часі від паралельної обробки.

Висновки до розділу

У третьому розділі виконано формування архітектури розподіленого динамічного фреймворку для кластеризації великих і просторово-прив’язаних наборів даних. Розроблена архітектура поєднує локальний аналіз, децентралізовану обробку та глобальне узагальнення, що забезпечує високу масштабованість і стійкість обчислювального процесу. Детально описано фазу локального аналізу, у якій дані обробляються незалежними вузлами із застосуванням відповідних алгоритмів кластеризації, що дозволяє мінімізувати міжвузлові комунікації. Показано, що така організація дає змогу значною мірою зменшити обсяг передаваних проміжних результатів і зберегти локальну структуру простору даних. Наступна фаза — агрегація — ґрунтується на методах геометричної реконструкції меж кластерів, що дозволяє узгоджувати часткові моделі без втрати релевантної інформації. Зокрема, застосування алгоритму α-shape забезпечило можливість побудови складних контурів кластерів, а алгоритм збалансованого вектора — оптимізацію їхнього глобального поєднання. Показано, що парадигма MapReduce може бути ефективно адаптована для реалізації динамічної кластеризації завдяки чіткій структурі map- та reduce-операцій. Інтеграція фреймворку з інфраструктурою Hadoop HDFS забезпечила стійкість до збоїв, балансування навантаження та ефективний розподіл проміжних файлів. Експериментальна перевірка довела здатність запропонованої архітектури до лінійного масштабування та збереження високої якості кластерної структури навіть у складних просторових сценаріях.

ВИСНОВКИ

У магістерській роботі проведено комплексне дослідження методів інтелектуального аналізу даних (ІАД) та особливостей їх реалізації засобами розподілених динамічних фреймворків, орієнтованих на обробку надвеликих, різнорідних та просторово-прив’язаних наборів даних.

Проведене дослідження предметної області показало, що феномен великих даних характеризується мультиаспектністю, високою інтенсивністю росту та необхідністю застосування спеціалізованих парадигм обробки, які забезпечують масштабованість, стійкість і здатність до динамічного розширення. У роботі виконано концептуалізацію процесів обробки Big Data, що дозволило визначити ключові властивості та вимоги до сучасних розподілених фреймворків, серед яких домінують: локальна обробка даних, толерантність до збоїв, паралельне виконання задач та оптимізація потоків розподіленого навантаження.

На основі порівняльного аналізу Hadoop MapReduce з іншими парадигмами встановлено, що попри існування новітніх підходів до потокової та in-memory обробки, модель MapReduce залишається фундаментальною та найбільш стабільною для реалізації динамічних кластеризаційних процесів з великим обсягом проміжних даних.

У другому розділі роботи систематизовано сучасні техніки інтелектуального аналізу даних, що були переосмислені в контексті їх застосування для масивів, які обробляються розподіленими обчислювальними системами. Проведено категоризацію класичних методів (класифікації, вилучення асоціативних правил, кластеризації, прогнозування), а також просторово-орієнтованих технік (SDM), що демонструють суттєві обмеження при масштабуванні.

Особливу увагу приділено проблематиці інтелектуального аналізу просторово-прив’язаних даних, для яких характерні топологічні залежності, неоднорідна щільність, просторові кореляції та високий рівень шумності.

Розглянуто методи просторових асоціативних правил, просторової класифікації та прогнозування, акцентуючи на необхідності адаптації цих алгоритмів до розподілених платформ.

Розроблено таксономію технік просторового датамайнінгу та детально проаналізовано методи кластеризації на основі розбиття, ієрархічні підходи та алгоритми, що використовують щільність структур простору. Встановлено, що масштабованість алгоритмів кластеризації істотно залежить від їх локальності, здатності до паралельного узагальнення та можливості незалежного опрацювання сегментів даних. Ці висновки стали методологічною основою для розробки власного розподіленого динамічного фреймворку.

У третьому розділі роботи запропоновано архітектуру розподіленого фреймворку, що реалізує динамічну кластеризацію великих обсягів даних шляхом поєднання локального аналізу та глобальної агрегації. Запропонована архітектура передбачає модульну побудову, інтеграцію механізмів локального обчислення, адаптивне формування часткових моделей та генерацію узагальненої глобальної кластерної структури.

Розроблено і формалізовано фазу локального аналізу, у межах якої дані обробляються на незалежних вузлах із використанням відповідних алгоритмів кластеризації. Показано, що такий підхід суттєво зменшує комунікаційні витрати та покращує масштабованість.

У роботі адаптовано парадигму MapReduce до задачі динамічної кластеризації, що дало змогу формалізувати процес обчислення в термінах map- та reduce-процедур. Показано, що інтеграція підходу зі структурою Hadoop HDFS забезпечує стійкість, автоматичне балансування навантаження та ефективний розподіл проміжних результатів.

Експериментальна частина роботи підтвердила працездатність і ефективність побудованого розподіленого фреймворку динамічної кластеризації. Отримані дані наочно підтверджують, що запропонована

архітектура здатна ефективно функціонувати у гетерогенних, ресурсно-обмежених та динамічних обчислювальних середовищах.

Магістерська робота становить комплексне наукове дослідження, яке поєднує сучасні методи інтелектуального аналізу даних, геометричні алгоритми реконструкції структур і принципи функціонування розподілених обчислювальних платформ. Основним результатом роботи є створення архітектурно цілісного та експериментально підтвердженого підходу до реалізації розподіленої динамічної кластеризації надвеликих наборів даних, що може бути використаний у територіальних інформаційних системах, екосистемному моніторингу, аналітиці мобільності, обробці супутникових даних та інших сферах, де обсяг даних значно перевищує можливості традиційних методів.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. CLARA in R : Clustering Large Applications – Datanovia. - https://www.datanovia.com/en/lessons/clara-in-r-clustering-large-applications/
2. A Review of Super-Resolution Single-Molecule Localization Microscopy Cluster Analysis and Quantification Methods - https://www.researchgate.net/publication/342141592_A_Review_of_Super-Resolution_Single-Molecule_Localization_Microscopy_Cluster_Analysis_and_Quantification_Methods
3. Apache Hadoop Architecture - HDFS, YARN & MapReduce – TechVidvan - https://techvidvan.com/tutorials/hadoop-architecture/
4. Improving Online Education Using Big Data Technologies - https://www.researchgate.net/publication/339897935_Improving_Online_Education_Using_Big_Data_Technologies
5. Smith, J., and L. Carter. “Distributed Clustering Models for Large-Scale Spatial Data.” Journal of Big Data Analytics, Berlin: Springer, 2021, pp. 45–62.
6. Lee, H., M. Gupta, and R. Ivanov. “Adaptive MapReduce Processing for Geospatial Datasets.” International Journal of Applied Data Science, London: Elsevier, 2020, pp. 112–130.
7. Patel, S., and K. Zhao. “Dynamic Frameworks for High-Density Spatial Clustering.” Proceedings of the IEEE International Conference on Data Engineering, Paris: IEEE Press, 2019, pp. 307–320.
8. Johnson, T., and P. Feng. “Topology-Aware Methods in Distributed Data Mining.” ACM Transactions on Knowledge Discovery from Data, New York: ACM, 2022, pp. 1–21.
9. Wang, L., and E. Torres. “Geometric Boundary Reconstruction with α-Shape Algorithms.” Pattern Recognition Letters, Amsterdam: Elsevier, 2020, pp. 89–104.

10. Brown, A., and N. Kim. “Parallelization Strategies for Spatial Density-Based Clustering.” Future Generation Computer Systems, London: Elsevier, 2021, pp. 354–370.
11. Huang, S., and P. Rodrigues. “High-Performance Spatial Mining Using Distributed Memory Architectures.” International Conference on High Performance Computing, Tokyo: IEEE Press, 2018, pp. 221–234.
12. Al-Sayed, R., and J. Morgan. “Fault-Tolerant Architectures in Big Data Frameworks.” IEEE Transactions on Cloud Computing, New York: IEEE, 2022, pp. 99–114.
13. Gomez, D., and F. Li. “Hybrid Approaches to Aggregation of Local Cluster Models.” Machine Learning and Data Mining Journal, Cambridge: MIT Press, 2019, pp. 67–82.
14. Koval, A., and R. Mendes. “Local Preprocessing Techniques for Distributed Clustering Tasks.” Data & Knowledge Engineering, Amsterdam: Elsevier, 2021, pp. 144–160.
15. Zhang, Y., and O. Steiner. “HDFS-Based Methods for Spatial Big Data Integration.” IEEE Symposium on Large-Scale Data Processing, Munich: IEEE Press, 2019, pp. 54–66.
16. Richards, J., and L. Verma. “Scalable Approaches to Geospatial Pattern Detection.” International Journal of Geographic Information Science, Taylor & Francis, 2020, pp. 211–229.
17. Silva, M., and B. Chan. “Dynamic Partitioning Algorithms for Distributed Spatial Analysis.” ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Seattle: ACM, 2021, pp. 143–155.
18. O’Neill, K., and F. Barros. “Cluster Boundary Optimization Using Geometric Constraints.” Journal of Computational Geometry, Cambridge: Cambridge University Press, 2022, pp. 33–49.

19. Park, D., and J. Silva. “Distributed Hierarchical Clustering under Memory Constraints.” Journal of Parallel and Distributed Computing, Amsterdam: Elsevier, 2019, pp. 272–289.
20. Nguyen, L., and A. Fischer. “Spatial Data Mining Techniques for Irregular Topologies.” International Journal of Spatial Data Science, London: Springer, 2020, pp. 113–128.
21. Chen, R., and D. Murray. “Enhanced Reduce-Phase Aggregation for Spatial Cluster Models.” IEEE Transactions on Big Data, New York: IEEE, 2021, pp. 188–203.
22. Ibrahim, S., and K. Young. “Load-Balancing Strategies in Heterogeneous Distributed Systems.” Concurrency and Computation: Practice and Experience, Wiley, 2020, pp. 999–1014.
23. Martens, P., and V. Duarte. “Parallel Geospatial Analytics in Multi-Node Environments.” Computers, Environment and Urban Systems, Elsevier, 2019, pp. 1–15.
24. Santos, G., and M. Lopez. “Noise-Resistant Methods for Spatial Cluster Reconstruction.” Expert Systems with Applications, Elsevier, 2021, pp. 122–138.
25. Adjei, K., and J. Conway. “Distributed Extraction of Spatial Association Rules.” Data Mining and Knowledge Discovery, Springer, 2019, pp. 361–379.
26. Van Dijk, S., and H. Kruger. “MapReduce Extensions for Spatially Dependent Data.” International Journal of Distributed and Parallel Systems, IEEE Press, 2020, pp. 211–227.
27. Ahmad, M., and R. Stone. “In-Memory Processing for Real-Time Cluster Evolution.” Proceedings of the ACM Symposium on Cloud Computing, Vienna: ACM, 2019, pp. 201–214.
28. López, R., and A. Brown. “Spatial Density Variation Analysis in Big Data Systems.” Information Systems Frontiers, Springer, 2021, pp. 243–259.

29. Ivanov, E., and T. Sakurai. “Cooperative Model Aggregation in Distributed Learning.” Journal of Machine Intelligence, IEEE, 2022, pp. 50–66.
30. Moretti, F., and G. Pavlic. “High-Resolution Spatial Cluster Models via α-Shape Reconstruction.” Computational Geometry: Theory and Applications, Elsevier, 2020, pp. 188–204.
31. Robinson, L., and K. Yeung. “Comparative Evaluation of Distributed Clustering Algorithms.” Journal of Intelligent Information Systems, Springer, 2019, pp. 281–297.
32. Chandra, R., and M. Velasquez. “Local Feature Extraction for Scalable Spatial Data Mining.” Information Processing Letters, Elsevier, 2018, pp. 89–102.
33. Kumar, P., and E. Silva. “Spatial Stream Processing in Heterogeneous Environments.” ACM Transactions on Spatial Algorithms and Systems, ACM, 2021, pp. 1–19.
34. Torres, A., and L. Schneider. “Geometric Techniques for Distributed Cluster Boundary Detection.” Journal of Applied Computational Science, Wiley, 2020, pp. 314–329.
35. Meier, J., and V. Abbas. “Dynamic Adaptation of Clustering Methods in Cloud-Based Systems.” Cloud Computing Journal, Springer, 2021, pp. 177–194.
36. Novak, P., and C. Hsu. “Fault-Tolerant MapReduce for Spatial Workloads.” Proceedings of the International Conference on Scalable Computing, Prague: IEEE, 2019, pp. 76–89.
37. Ghosh, S., and L. Ortega. “Hybrid Parallel Approaches for Density-Based Clustering.” Parallel Computing, Elsevier, 2020, pp. 133–148.
38. Fedorov, A., and M. Greco. “Constraint-Based Spatial Clustering in Big Data Platforms.” Knowledge and Information Systems, Springer, 2022, pp. 212–230.

39. Schneider, J., and H. Kim. “Evaluating Geospatial Clustering Quality in Distributed Systems.” IEEE Transactions on Knowledge and Data Engineering, IEEE, 2020, pp. 456–470.
40. Li, X., and V. Marques. “Boundary-Aware Aggregation in Large-Scale Data Clustering.” International Journal of Geographical Computation, Elsevier, 2021, pp. 59–75.
41. Baran, D., and R. Collins. “Distributed Algorithms for High-Noise Spatial Datasets.” Pattern Analysis and Applications, Springer, 2019, pp. 321–338.
42. Garcia, M., and P. Hawke. “Topology-Preserving Clustering for Geospatial Intelligence Systems.” GeoInformatica, Springer, 2020, pp. 411–429.
43. Yu, Z., and J. Arnolds. “Scalable Cluster Formation Using Multi-Stage MapReduce.” Journal of Large-Scale Distributed Systems, IEEE, 2021, pp. 88–104.
44. Kwon, S., and B. Lambert. “Integrated Frameworks for Spatial Big Data Analysis and Cluster Reconstruction.” Transactions on Spatial Information Science, Springer, 2022, pp. 142–160.