НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМ. ІГОРЯ СІКОРСЬКОГО»

ННК «Інститут прикладного системного аналізу»
(повна назва інституту/факультету)

Системного проектування
(повна назва кафедри)

«На правах рукопису»
УДК 004.75

«До захисту допущено»
Завідувач кафедри
_____ А.І. Петренко
(підпис) (ініціали, прізвище)
“___”_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 8.05010103 Системне проектування
(код і назва)

на тему: Побудова моделей обробки великих масивів даних з використанням технології MapReduce.

Виконав: студент VI курсу, групи ДА-52м
(шифр групи)

Яременко Вадим Сергійович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник проф., д.т.н., Рогоза В.С. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант Розроблення стартап-проекту проф., д.т.н., Рогоза В.С. _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент проф., д.т.н. Аушева Н.М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2017 року

Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет (інститут) ННК "Інститут прикладного системного аналізу"
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти Другий (Магістерський)
(перший (бакалаврський), другий (магістерський) або спеціаліста)

Спеціальність 8.05010103 Системне проектування
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І. Петренко
(підпис) (ініціали, прізвище)

«___»_____2017 р.

ЗАВДАННЯ
на дипломний проект (роботу) студенту
Яременко Вадиму Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) «Побудова моделей обробки великих масивів даних з використанням технології MapReduce»

керівник проекту (роботи) Рогоза В.С., д.т.н., проф.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «20» березня 2017 р. №32-ст

2. Термін подання студентом дисертації 01.06.2017

3. Об'єкт дослідження Великі масиви даних

4. Предмет дослідження Моделі для вирішення задач кластеризації та пошуку частих предметних наборів, які основані на технології MapReduce

5. Перелік завдань, які потрібно розробити
   1. Аналіз технології MapReduce та виділення рекомендацій по її використанню
   2. Створення моделі для вирішення задачі кластеризації з використанням технології MapReduce
   3. Створення моделі для вирішення задачі пошуку частих предметних наборів з використанням технології MapReduce
   4. Спроектувати програми реалізації запропонованих моделей з використанням мови UML та реалізувати їх програми на мові Java
   5. Провести тестування розроблених моделей
   6. Оформити роботу на основі отриманих результатів

6. Орієнтовний перелік публікацій

1. Yaremenko V. An approach for data clustering CURE algorithm implementation using the MapReduce technology / Vadym Yaremenko. // 19-th International Conference SAIT 2017 Kyiv, Ukraine, May 22 – 25, 2017. – 2017. – P. 204.
2. Yaremenko V. Distributed data clustering CURE algorithm approbation using Hadoop MapReduce / Vadym Yaremenko. // International Scietific Journal «Internauka». – 2017. – №6. – P. 81–83.

7. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Реалізація стартап-проекту	Рогоза В.С., проф.		

8. Дата видачі завдання 01.02.2017

Календарний план

№ з/п	Назва етапів виконання дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання	01.02.2017	
2	Збір інформації та аналіз літератури	15.02.2017	
3	Розробка моделей для вирішення задач обробки великих даних	28.02.2017	
4	Розробка програм, що реалізують запропоновані моделі	15.04.2017	
5	Тестування моделей	30.04.2017	
6	Оформлення дипломної роботи	31.05.2017	
7	Отримання допуску до захисту та подача роботи в ДЕК	08.06.2017	

Студент _____ Яременко В.С.
(підпис) (ініціали, прізвище)

Керівник проекту (роботи) _____ Рогоза В.С.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ

виконану на тему: Побудова моделей обробки великих масивів даних з використанням технології MapReduce

студентом: Яременко Вадимом Сергійовичем

Робота виконана на 124 сторінках, містить 43 ілюстрацій, 36 таблиць. При підготовці використовувалась література з 53 джерел.

Актуальність теми

На сьогоднішній день обсяг даних стає занадто великим для того, щоб була можлива їх обробка традиційними алгоритмами. Сучасні програми інтелектуального аналізу даних, які часто називають великими даними, вимагають від нас швидкого керування величезними обсягами даних. Однією із технологій для вирішення задач обробки великих даних є MapReduce.

Тому дослідження даної технології, а також побудова нових моделей, що базуються на ній, є актуальним напрямком досліджень саме на сьогоднішній день, у час, коли кількість даних постійно зростає, а єдиного підходу до рішення задач інтелектуального аналізу не існує.

Мета та задачі дослідження

Метою даної роботи є дослідження технології MapReduce та її використання для вирішення задач обробки великих масивів даних на прикладі задач кластеризації та пошуку частих предметних наборів. Результатом проведених досліджень є практична частина роботи, що становить собою створення моделей для обробки великих даних та їх апробація з використанням сучасних програмних засобів.

Рішення поставлених завдань та досягнуті результати

У даній роботі були запропоновані дві моделі обробки великих масивів даних. В основі першої моделі, для вирішення задачі кластеризації, лежить алгоритм CURE, який використовує репрезентативну вибірку кластеру і не

зберігає усі його точки. В основі другої моделі, для вирішення задачі пошуку частих предметних наборів, лежить алгоритм SON, який за два проходи по даним знаходить часті набори, що задовільняють заданим користувачем параметрам.

Апробація моделей була проведена на локальному комп'ютері та на сервері кафедри Системного проектування, що складався з трьох вузлів. З отриманих результатів можна зробити висновок, що розроблені моделі доцільно використовувати для даних більших за 500 МБ, а також важливо, що зі збільшенням кількості вузлів, швидкість обробки зростає.

Об’єкт досліджень

Великі масиви даних.

Предмет досліджень

Моделі для вирішення задач кластеризації та пошуку частих предметних наборів з використанням технології MapReduce

Методи досліджень

Для вирішення проблеми в даній роботі використовуються методи аналізу і синтезу, системного аналізу, порівняння, логічного узагальнення результатів, проектування логічних структур даних.

Наукова новизна

Наукова новизна роботи полягає у створенні нових моделей для вирішення задач кластеризації та пошуку частих предметних наборів з використанням технології MapReduce, а також у апробації цих моделей на практиці – порівнянні швидкості обробки даних з програмами, які не використовують розподілені технології обробки даних.

Практичне значення одержаних результатів

Отримані результати можуть використовуватись у майбутніх дослідженнях за напрямком створення моделей обробки великих даних, враховуючи переваги та недоліки даних результатів. Також завдяки двом

науковим публікаціям англійською мовою та їх розміщенню в мережі інтернет, результати дослідження будуть доступні також за межами України.

Також розроблені програми можуть бути використані як основа для створення лабораторного практикуму з дисципліни «Інтелектуальний аналіз даних», що викладається на кафедрі Системного проектування.

Апробації результатів дисертації

Результати досліджень оприлюднені на 19-й Міжнародній науково-технічній конференції SAIT 2017, а також – у міжнародному науковому журналі «Інтернаука», випуск №6 2017 року.

Публікації

Yaremenko V. An approach for data clustering CURE algorithm implementation using the MapReduce technology / Vadym Yaremenko. // 19-th International Conference SAIT 2017 Kyiv, Ukraine, May 22 – 25, 2017. – 2017. – P. 204.

Yaremenko V. Distributed data clustering CURE algorithm approbation using Hadoop MapReduce / Vadym Yaremenko. // International Scietific Journal «Internauka». – 2017. – №6. – P. 81–83.

Ключові слова

Інтелектуальний аналіз даних, великі дані, розподілені обчислення, кластеризація, часті предметні набори, Hadoop, MapReduce.

РЕФЕРАТ

НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ

выполненную на тему: Построение моделей обработки больших массивов даннхы с использованием технологии MapReduce

студентом: Яременко Вадимом Сергеевичем

Работа выполнена на 124 страницах, содержит 43 иллюстраций, 36 таблиц. При подготовке использовалась литература из 53 источников.

Актуальность темы

На сегодняшний день объем данных становится слишком большим для того, чтобы была возможна их обработка традиционными методами. Современные программы интеллектуального анализа данных, которые часто называют большими данным, требуют от нас быстрого управления огромными объемами данных. Одной из технологий для решения задач обработки больших данных является MapReduce.

Поэтому исследования данной технологии, а также построение новых моделей, основанных на ней, является актуальным направлением исследований именно на сегодняшний день, в то время, когда количество данных постоянно растет, а единого подхода к решению задач интеллектуального анализа не существует.

Цель и задачи исследования

Целью данной работы является исследование технологии MapReduce и ее использования для решения задач обработки больших массивов данных на примере задач кластеризации и поиска частых предметных наборов. Результатом проведенных исследований является практическая часть работы, представляет собой создание моделей для обработки больших данных и их апробация с использованием современных программных средств.

Решение поставленных задач и достигнутые результаты

В данной работе были предложены две модели обработки больших массивов данных. В основе первой модели, для решения задачи кластеризации, лежит алгоритм CURE, который использует репрезентативную выборку кластера и не хранит все его точки. В основе второй модели, для решения задачи поиска частых предметных наборов, лежит алгоритм SON, который за два прохода по данным находит частые наборы, которые удовлетворяют заданным пользователем параметрам.

Апробация моделей была проведена на локальном компьютере и на сервере кафедры Системного проектирования, состоял из трех узлов. Из полученных результатов можно сделать вывод, что разработанные модели целесообразно использовать для данных превышающих 500 МБ, а также важно, что с увеличением количества узлов, скорость обработки возрастает.

Объект исследования

Большие массивы данных.

Предмет исследования

Модели для решения задач кластеризации и поиска частых предметных наборов с использованием технологии MapReduce.

Методы исследования

Для решения проблемы в данной работе используются методы анализа и синтеза, системного анализа, сравнения, логического обобщения результатов, проектирования логических структур данных.

Научная новизна

Научная новизна работы заключается в создании новых моделей для решения задач кластеризации и поиска частых предметных наборов с использованием технологии MapReduce, а также в апробации этих моделей на практике - сравнению скорости обработки данных с программами, которые не используют распределены технологии обработки данных.

Практическае значение результатов

Полученные результаты могут использоваться в будущих исследованиях по направлению создания моделей обработки больших данных, учитывая преимущества и недостатки данных результатов. Также благодаря двум научным публикациям на английском языке и их размещению в сети интернет, результаты исследования будут доступны также за пределами Украины.

Также разработаны программы могут быть использованы как основа для создания лабораторного практикума по дисциплине «Интеллектуальный анализ данных», который преподается на кафедре системного проектирования.

Аппробация результатов диссертации

Результаты исследования опубликованы на 19-й Международной научно-технической конференции SAIT 2017, а также – в международном научном журнале «Интернаука», выпуск №6 2017 года.

Публикации

Yaremenko V. An approach for data clustering CURE algorithm implementation using the MapReduce technology / Vadym Yaremenko. // 19-th International Conference SAIT 2017 Kyiv, Ukraine, May 22 – 25, 2017. – 2017. – P. 204.

Yaremenko V. Distributed data clustering CURE algorithm approbation using Hadoop MapReduce / Vadym Yaremenko. // International Scietific Journal «Internauka». – 2017. – №6. – P. 81–83.

Ключевые слова

Интеллектуальный анализ данных, большие данные, распределенные вычисления, кластеризация, частые предметные наборы, Hadoop, MapReduce.

ABSTRACT

ON MASTER'S THESIS

on topic: Design of models for big data processing using the MapReduce technology

student: Vadym S. Yaremenko

Work carried out on 124 pages containing 43 figures, 36 tables. The paper was written with references to 53 different sources.

Topicality

Today the volume of data is too large to be processed using traditional algorithms. Modern applications of data mining, often called big data, require us to quickly control the huge amounts of data. One of the technologies for solving large data processing is MapReduce.

Therefore, the study of the technology and construction of new models based on it is relevant area of research today, at a time when the amount of data is growing, and a common approach to data mining problem solving does not exist.

Purpose

The aim of this work is to study MapReduce technology and its use to solve problems processing large amounts of data for clustering and frequent itemsets problem solving. The result of the research is the practical part of the work is the creation of models for processing large data, which was tested using modern technologies.

Solution

In this abstract are proposed two models for processing large data sets. The basis of the first model to solve the problem of clustering algorithm CURE is, using a representative sample of the cluster or store all of its terms. The basis of the second model for solving the problem of finding frequent itemsets is SON algorithm, which passes through two MapReduce steps and finds frequent itemsets that satisfy user-specified parameters.

Testing of models was performed on the local computer and a server on the System Design Department, consisting of three clusters. From the results we can conclude that the developed model should be used for data larger than 500 MB, and it is important that the increasing number of nodes increases processing speed.

The object of research

Big data.

The subject of research

Models for solving clustering and frequent itemsets search using the MapReduce technology.

Research methods

To solve the problem in this research such methods were used: analysis and synthesis, system analysis, comparison, logical generalization of the results, design logical data structures.

Scientific novelty

Scientific novelty lies in creating new models for solving clustering and finding frequent subject sets using MapReduce technology, as well as in testing these models in practice - processing speed compared with programs that do not use distributed data processing technology.

The practical value of research

The results can be used in future studies focus on creating models handle large data, including advantages and disadvantages of these results. Also thanks to two scientific publications in English, which are placed on the Internet, so results will be available also outside Ukraine.

Also developed applications can be used as a basis for the creation of laboratory works on discipline "Data mining" at the Department of System Design.

Research results approbation

Research results presented at the 19th International Scientific Conference SAIT 2017, as well as in the international scientific journal "Internauka", issue №6 2017.

Publications

Yaremenko V. An approach for data clustering CURE algorithm implementation using the MapReduce technology / Vadym Yaremenko. // 19-th International Conference SAIT 2017 Kyiv, Ukraine, May 22 – 25, 2017. – 2017. – P. 204.

Yaremenko V. Distributed data clustering CURE algorithm approbation using Hadoop MapReduce / Vadym Yaremenko. // International Scietific Journal «Internauka». – 2017. – №6. – P. 81–83.

Keywords

Data mining, Bid Data, Distributed calculations, clustering, frequent itemsets, Hadoop, MapReduce.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

Hadoop MapReduce – фреймворк для простого створення програм вирішення задач обробки великих даних

HDFS – Hadoop distributed file system

MapReduce – технологія для розподіленої обробки великих даних

CURE – Clustering using representatives

SON – Savasere, Omiecinski, and Navathe algoritm

SQL – Structured Query Language

ВСТУП

З появою обчислювальної техніки обсяг даних, з якими може працювати людини, значно збільшився. Ще понад 30 років тому вченими стали розроблятися алгоритми, що мають на меті спрощення роботи з даними, виявлення нових, раніше невідомих знань, що зберігаються в даних.

Однак сьогодні обсяг збережених даних стає занадто великим для того, щоб була можлива їх обробка традиційними алгоритмами. Дослідникам доводиться йти на різні хитрощі, такі як робота з наявними даними по частинах, використання апріорних знань про наявні даних.

Таким чином, робота з великими даними доступна тільки кваліфікованим фахівцям, що породжує необхідність формалізації нових методів, використовуваних дослідниками, створення нових алгоритмів і програмних засобів, що використовують міць створених раніше інструментів, для роботи з даними великих обсягів і розмірностей [2017, https://www-01.ibm.com/software/data/infosphere/hadoop/mapreduce/].

Сучасні програми інтелектуального аналізу даних, які часто називають великими даними, вимагають від нас швидкого керування величезними обсягами даних. У багатьох з цих програм дані представлені у такому вигляді, що існують можливості для використання технологій паралельних обчислень. Для вирішення даних проблем був створений спеціальний стек програмного забезпечення.

Головне місце в цьому стеку програмного забезпечення займає технологія під назвою MapReduce. Реалізації MapReduce дозволяє виконувати багато з найбільш поширених розрахунків великих даних, які повинні виконуватися на обчислювальних кластерах ефективно і враховуючи можливі апаратні збої. Технологія MapReduce швидко розвивається і поширюється. Сьогодні вона є загальною для програм, які були створені з систем програмування більш високого рівня, ніж SQL.

Системи, основані на MapReduce, розроблені таким чином, щоб паралельність була реалізована не за рахунок суперкомп'ютера, а за рахунок обчислювальних кластерів – наборів стандартних апаратних засобів, в тому числі звичайних процесорів (обчислювальних вузлів), з'єднаних за допомогою кабелів Ethernet або недорогих комутаторів. Стек програмного забезпечення починається з новою формою файлової системи, яку називають «розподілена файлова система». Такі файлові системи також забезпечують реплікацію даних або резервування для захисту від збоїв, які відбуваються, коли дані розподілені по тисячам недорогих обчислювальних вузлів.

Більшість обчислень робляться на одному процесорі, з його основної пам'яті, кеш-пам'яті, а також локального диску (на обчислювальному вузлі). У минулому програми для паралельної обробки даних, наприклад, для наукових розрахунків, були зроблені на спеціальних паралельних комп'ютерах з великою кількістю процесорів і спеціалізованим обладнанням. Проте домінування великих веб-служб створило передумови для того, щоб великі розрахунки робились в установках з тисячами обчислювальних вузлів, які працюють більш-менш незалежно один від одного. У цих установках обчислювальні вузли є товаром, що значно знижує витрати в порівнянні зі спеціальними паралельними машинами.

Ці нові обчислювальні засоби призвели до нового покоління систем програмування. Такі системи користуються перевагами паралельності і в той же час уникають проблем з надійністю, які виникають, коли обчислювальний вузол має апаратне забезпечення з тисячами незалежних компонентів, кожен з яких може вийти з ладу в будь-який час.

Нова паралельно-обчислювальна архітектура, яку іноді називають кластери для обчислень, організована наступним чином: обчислювальні вузли зберігаються на стелажах, приблизно по 8-64 на стійці. Вузли на одній стійці з'єднані мережею, зазвичай Gigabit Ethernet. Там може бути багато стійок обчислювальних вузлів, а також стійки з'єднані іншим рівнем мережі або комутатора. Смуга пропускання між стійкою зв'язку дещо більше, ніж Ethernet,

але з огляду на кількість пар вузлів, які, можливо, повинні взаємодіяти між стійками, ця пропускна здатність може бути істотною.

Проблема збору даних, їх інтеграції та інтелектувального аналізу гостро стоїть на даний час. Саме тому питання використання технологій масивно-паралельних СКБД, рішень класу бізнес-аналізу (Business Intelligence), нетрадиційних СКБД NoSQL та альтернативних рішень для побудови систем, що забезпечують розподілену обробку даних, є дуже актуальним [Leskovec J., 2014., 22 p]. Серед таких рішень особливою популярністю користується фреймворк Hadoop MapReduce. На сьогоднішній день існує багато задач Data Mining, деякими з яких є кластеризація та пошук частих предметних наборів.

Кластеризація, як перший етап аналізу даних, здійснюється на підставі певних методів, які мають відображення в алгоритмах розбиття груп об'єктів. Виділення груп дозволяє спростити роботу з даними, після кластеризації застосовуються інші методи, для кожної групи будується окрема модель.

Інша задача полягає у виявленні максимальних послідовностей серед всіх послідовностей, що мають підтримку вище заданого порогу. Кожна така максимальна послідовність і є послідовним шаблон. Послідовності, що задовольняють обмеження мінімальної підтримки є частими послідовностями.

Для кожної із задач у випадку великого набору вхідних даних існує проблема, що не всі дані можливо одночасно розмістити в оперативній пам'яті. Тому розробка алгоритмів, що дозволяють вирішувати ці задачі, завантажуючи в один момент часу лише частину даних, є важливим напрямком досліджень.

Метою даної роботи є дослідження технології MapReduce та її використання для вирішення задач обробки великих масивів даних на прикладі задач кластеризації та пошуку частих предметних наборів. Результатом проведених досліджень є практична частина роботи, що становить собою створення моделей для обробки великих даних та їх апробація з використанням сучасних програмних засобів.

1 ОГЛЯД ТЕХНОЛОГІЇ MAPREDUCE ТА ФРЕЙМВОРКІВ ДЛЯ ЇЇ РЕАЛІЗАЦІЇ

1.1 Загальний огляд MapReduce

MapReduce є технологією для розподілених обчислень, яка була реалізована в декількох системах, в тому числі внутрішньої реалізації компанії Google (просто називається MapReduce) і популярної системи з відкритим кодом Hadoop, який може бути отриманий разом з файловою системою HDFS [http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html] з фонду Apache. Можна використовувати реалізацію MapReduce для управління багатьма великомасштабними обчисленнями таким чином, щоб апаратні несправності не були проблемою. Все, що потрібно розробнику, це написати дві функції, які називаються Map та Reduce, в той час як система самостійно управляє паралельним виконанням, координацією завдань, які виконуються, а також контролює ситуації, коли одне з цих завдань не в змозі бути виконаним. Схема обчислень зображена на рис. 1. Обчислення MapReduce виконується наступним чином:

1. Частина даних разом з функцією Map відправляються на один з обчислювальних вузлів на виконання. Виходом цієї програми з одного кластеру є пари ключ-значення.
2. Ці пари ключ-значення з кожного завдання на збираються на головному контролері і упорядковуються відповідно до ключу.
3. Функція Reduce виконує обробку даних, упорядкованих до ключу. Спосіб обробки визначає код, написаний користувачем для функції Reduce.

Це факт, що іноді з ладу виходять компоненти мережі, і чим більше обчислювальних вузлів і з'єднань має система, тим більша ймовірність, що щось в системі не буде працювати в будь-який момент часу.

Деякі важливі розрахунки можуть зайняти кілька хвилин або навіть години на тисячах обчислювальних вузлів. Якби нам довелося переривати і перезапускати обчислення кожен раз, коли один компонент виходить з ладу, то обчислення не могли б завершитися успішно. Вирішення цієї проблеми має дві форми:

1. Файли повинні мати резервні копії. Якби ми не дублювати файл на декількох обчислювальних вузлах, а потім, якщо один вузол вийшов з ладу, то всі його файли були б недоступні, поки вузол не буде полагоджений. Якби ми не створювали резервні копії файлів з усіх вузлів, і відбувся збій диска, то файли були б втрачені назавжди.
2. Розрахунки повинні бути розділені на завдання, такі, що якщо одна задача не може виконати до кінця, він може бути перезапущений, не зачіпаючи інші завдання.

Для того, щоб використовувати кластерні обчислення, файли повинні виглядати трохи інакше, ніж в звичайних файлових систем, які знаходяться на окремих комп'ютерах. Ця нова файлова система, яку часто називають розподіленою файловою системою або DFS (хоча цей термін має також інші значення в минулому), як правило, використовується наступним чином [Leskovec J., 2014].

Файли розділені на частини, які, як правило, розміром у 64 МБ. Ці частини реплікуються, наприклад, в три рази, в три різних обчислювальних вузли. Крім того, одна частина повинна бути розташована на різних стійках, тому ми не втратимо всі копії через збій в стійці. Зазвичай, такі параметри, як розмір блоку і ступінь реплікації задаються користувачем. Для того, щоб знайти частини файлу, існує ще один невеликий файл, який називається головний вузол. Головний вузол самостійно реплікується, і каталог для файлової системи знає, де знайти його копію. Сам каталог може бути відтворений, і всі учасники, що використовують DFS мають, де знаходиться копія каталогу.

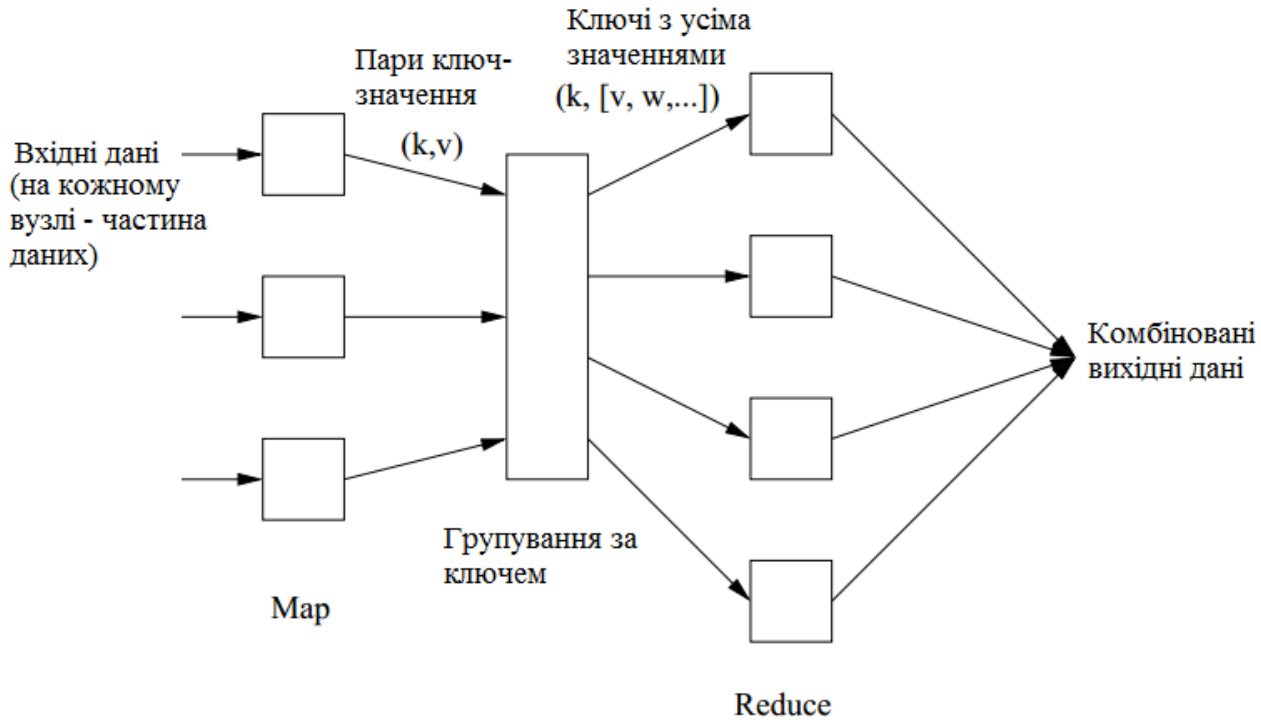

Рис. 1.1 – Схема обчислень з використанням MapReduce

Вхідні файли для завдання Map складаються з елементів, які можуть бути будь-якого вигляду: масив, документ, тощо. Частина являє собою сукупність елементів, і жоден з елементів не зберігаються в двох частинах даних. Технічно, всі входи до завдання Map і виходи з Reduce завдань мають вигляд пар ключ-значення, але зазвичай ключі вхідних елементів не мають значення, і їх можна ігнорувати. Ця форма входів і виходів мотивована бажанням дозволити композиції з декількох процесів MapReduce. Функція Map приймає вхідний елемент як аргумент і виробляє нуль або більше пар ключ-значення. Типи ключів і значень кожен вузол генерує довільно. Крім того, «ключі» не повинні бути унікальними [Leskovec J., 2014]. Найчастіше за одне завдання функція Map може призвести кілька пар ключ-значення з тим же ключем, навіть з того ж елемента. Необхідно зазначити, що одна задача Map, як правило, обробляти велику кількість документів – всі документи знаходяться одній або декількох частинах даних.

Як тільки завдання Map успішно завершилось, пари ключа-значення групуються по ключу, а значення, пов'язані з кожним ключем формуються в список значень. Угруповання здійснюється за допомогою системи, незалежно

від того, що роблять Map і Reduce завдання. Процес головного контролера знає, скільки буде Reduce завдань. Користувач зазвичай вказує системі MapReduce, що має бути певна кількість завдань (позначимо *r*), і тоді головний контролер вибирає хеш-функцію, яка застосовується до ключів і виробляє ряд значень від 0 до r – 1. Кожен ключ, який виводиться за допомогою завдання Map хешується і його пари ключ-значення поміщаються в одному з r локальних файлів. Кожен файл призначений для одного з Reduce завдань [Leskovec J., 2014]. Щоб виконати угруповання по ключу і розподілити файли до Reduce завдань, головний контролер об'єднує файли з кожного завдання Map, які призначені для конкретного Reduce завдання і подає об'єднаний файл в цей процес як послідовність пар ключ-спискок значень.

Аргументом функції Reduce є пара, що складається з ключа і його списку відповідних значень. Виходом функції Reduce є послідовність з нуля або більше пар ключ-значення. Ці пари ключ-значення можуть іншого ж типу, як і послані від завдань Map, але часто вони мають однаковий тип. Завдання Reduce отримує один або кілька ключів і пов'язані з ними списки значень. Тобто, Reduce виконує одне або декілька завдань. Виходи з усіх Reduce завдань об'єднуються в один файл. Іноді функція Reduce обладає властивостями асоціативності і комутативності. Тобто, значення можуть бути об'єднані в будь-якому порядку, а результат буде незмінним [Leskovec J., 2014].

На рисунку 2 зображено схему, яка містить процеси, завдання та файли, та їх взаємодію між собою. Користуючись бібліотеками, наданими системою Hadoop MapReduce, користувач програми має розветвління процесів, а саме – головний процес та деяку кількість робочих процесів на різних обчислювальні вузлах. Як правило, на вузлах обробляється завдання Map, або згортка, але не обидва одночасно.

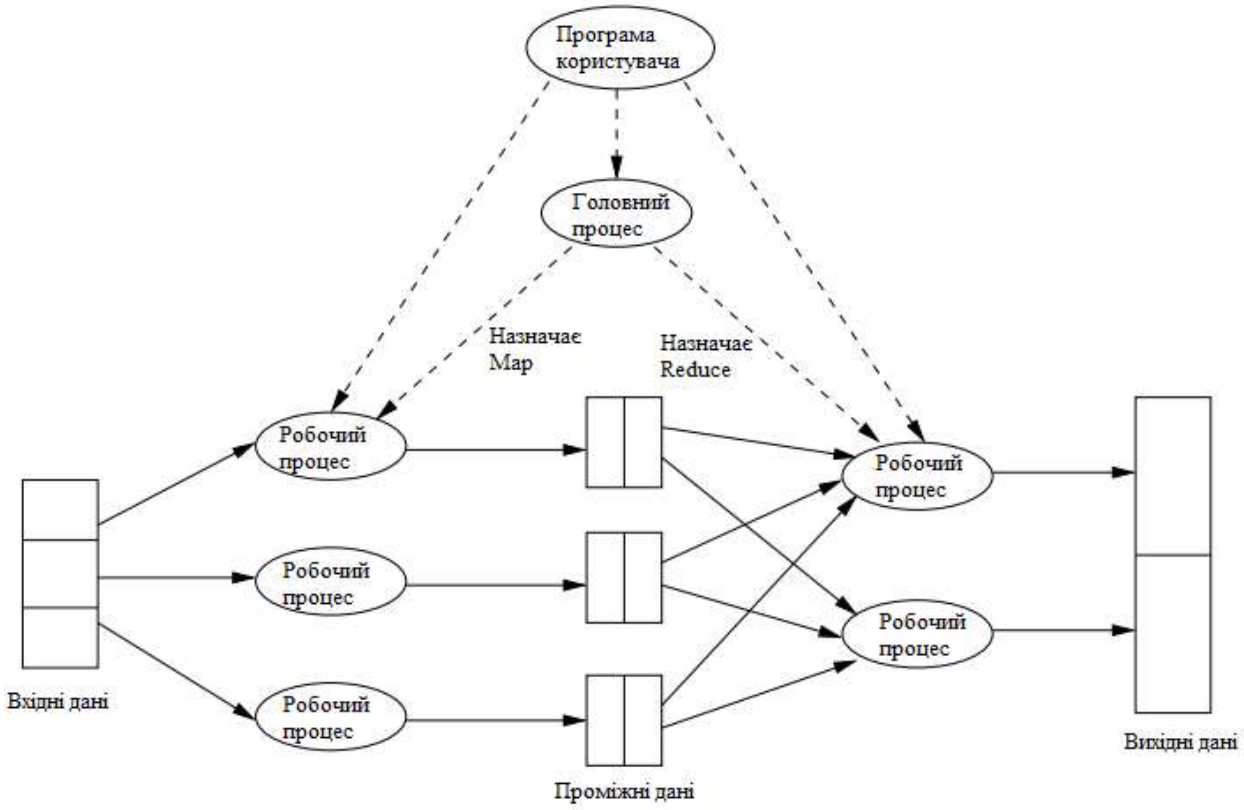

Рис. 1.2 – Загальний огляд запуску програми MapReduce

Головний вузол має багато обов'язків. Одним з них є створення деякого числа завдань Map і деякого числа завдань Reduce. Ці цифри обираються в програмі користувачем. Ці завдання будуть призначені на робочі процеси за допомогою головного вузла. Розумно створити одне Map завдання для кожного фрагмента вхідного файлу, але тоді необхідно буде створити менше Reduce завдань. Причина обмеження числа завдань є те, що необхідно для кожного завдання Map створити проміжний файл на вхід до завдання Reduce, і якщо є занадто багато Map завдань, кількість проміжних файлів буде значно більша за кількість Reduce завдань. Тому необхідно перед тим, як задати кількість завдань Map проаналізувати можливості завдань Reduce [Leskovec J., 2014].

Головний процес відстежує стан кожного завдання Map і Reduce (очікування виконання на конкретному вузлі). Робочий процес підпорядковується головному, коли він закінчує завдання, тоді нове завдання може бути надане цьому робочому процесу.

Кожне завданню Map задається одна або кілька частин вхідного файлу(ів) і після цього на ньому виконується код, написаний користувачем. Завдання Map створює файл для кожного Reduce завдання на локальному диску кластеру, який виконує завдання. Майстру повідомляється про місце і розмірів кожного з цих файлів. Коли завдання Reduce призначається майстром до робочого процесу, то завданню даються всі необхідні файли, які формують його вхід. Завдання Reduce виконує код, написаний користувачем, і записує свій результат в файл, який є частиною розподіленої файлової системи.

Найгірше, що може статися, що обчислювальний вузол, на якому працює головний процес, перестає працювати. У цьому випадку вся MapReduce робота повинна бути перезапущено. Але тільки цей один вузол може завдати шкоду всьому процесу; інші відмови будуть управлятися цим вузлом, і робота MapReduce буде завершена без великих затримок.

Припустимо, що обчислювальний вузол, на якому виконується завдання Map перестає працювати. Ця зупинка буде виявлена головним вузлом, тому що він періодично перевіряє стан робочих процесів. Всі завдання Map, які були призначені на цей вузол повинні будуть бути перероблені, навіть якщо вони були завершені. Причина переробки завершених завдань Map є те, що їх результат збережений саме на цьому вузлі. Головний вузол встановлює статус кожної з цих задач Map як готовий до виконання на іншому вузлі [Leskovec J., 2014].

Випадок з відмовою в вузлі Reduce обробляти простіше. Головний вузол просто перериває виконання на цьому вузлі та назначає виконання на іншому вузлі, як тільки той стає доступним.

Також варто розглянути технологію MapReduce на простому прикладі. Припустимо, що є п'ять файлів, і кожен файл містить два стовпці (ключ і значення з точки зору Hadoop), які представляють місто і відповідну температуру, записану в цьому місті на різні дні вимірювань. Звичайно, ми зробили цей приклад дуже простий, але він є наглядний. Реальне застосування не буде таке просте, як цей приклад. Як мінімум – у реальному випадку будуть

мільйони або навіть мільярди рядків, більше того, вони не будуть так відформатовані, як в цьому прикладі. У будь-якому випадку, в цьому прикладі, місто є ключом і температура – значенням [2017, https://www-01.ibm.com/software/data/infosphere/hadoop/mapreduce/].

Таблиця 1.1 – Температурні значення у містах

Місто	Температура
Торонто	20
Вітбі	25
Нью-Йорк	22
Рим	32
Торонто	4
Рим	33
Нью-Йорк	18

З усіх даних, які ми зібрали, ми хочемо знайти максимальну температуру для кожного міста через всі файли дані (варто звернути увагу, що кожен файл може мати одне і те саме місто представлене кілька разів). Використовуючи механізми MapReduce, ми можемо розбити це на п'ять Map завдань, де кожен Map працює з одним із п'яти файлів і завдання Map обробляє весь файл і повертає максимальну температуру для кожного міста. Наприклад, результати отримані з одного Map завдання для наведених вище даних буде виглядати як показано у таблиці 1.2 [2017, https://www-01.ibm.com/software/data/infosphere/hadoop/mapreduce/].

Таблиця 1.2 – Максимальні температури кожного міста

Місто	Температура
Торонто	20
Вітбі	25
Нью-Йорк	22
Рим	33

Варто припустимо, що інші чотири Map завдання (які працюють з чотирма ішиним файлами) дали наступні проміжні результати.

Таблиця 1.3 – Проміжні результати над іншими файлами

Номер файлу	Місто	Температура
1	Торонто	18
1	Вітбі	27
1	Нью-Йорк	32
1	Рим	37
2	Торонто	32
2	Вітбі	20
2	Нью-Йорк	33
2	Рим	38
3	Торонто	22
3	Вітбі	19
3	Нью-Йорк	20
3	Рим	31
4	Торонто	31
4	Вітбі	22
4	Нью-Йорк	19
4	Рим	30

Всі ці п'ять вихідних потоків будуть подані на вхід завдання Reduce, які поєднують в собі результати і кінцевий результат встановлюється таким, як зазначено в таблиці 1.4. Іншою аналогією може бути така: те, які шляхи перепису населення існували в римські часи, коли бюро перепису розсилало своїх людей в кожне місто в імперії. Кожен з цих людей в кожному місті мав порахувати кількість людей, а потім повернути ці результати в столицю.

Таблиця 1.4 – Кінцевий результат завдання MapReduce

Місто	Температура
Торонто	32
Вітбі	27
Нью-Йорк	33
Рим	38

Результати від кожного міста були зведені до одного відліку (суми по всім містам), щоб визначити загальну чисельність населення імперії. Тобто, в якості завдання Map треба було порахувати кількість людей в одному місті, а в якості завдання Reduce – додати значення, отримані по кожному місту. Цей процес, очевидно, швидший за відравлення однієї людини по всім містам імперії [2017, https://www-01.ibm.com/software/data/infosphere/hadoop/mapreduce/].

В даному пункті був представлений загальний огляд технології MapReduce та приклад її використання на практиці. У будь-якому випадку, перед використанням технології необхідно розуміти для яких саме задач вона призначена та чи доцільно її використовувати. В наступних пунктах будуть розглянуті альтернативи цієї технології та рекомендаціїї до її використання.

1.2 Альтернативи технології MapReduce

Традиційним рішенням для використання в аналітичних системах і сховищах даних з обсягами даних від сотень гігабайт до сотень терабайт були реляційні СУБД. На противагу рішенню в аналізі даних СУБД з великою кількістю дисків – накопичувачів, MapReduce виграє на апаратному рівні. Причина в особливості роботи накопичувачів на жорстких дисках, а саме те, що час пошуку даних прискорюється повільніше, аніж швидкість їх передачі. Процес пошуку характеризується затримкою дискових операцій, в той час як швидкість передачі даних залежить від пропускної здатності диска. Якщо

модель доступу до даних переважає над пошуком, то процес займе більше часу, щоб зчитати або записати великі частини набору даних, аніж час, затрачений на прохід через їх, який управляється швидкістю передачі даних.

У багатьох відносинах MapReduce можна розглядати як доповнення до СУБД . MapReduce добре підходить для задач, в яких необхідно проаналізувати весь набір даних в пакетному представленні. Перевагу СУБД слід надати при роботі з одиночними запитами або оновленнями, в яких набір даних був проіндексований для доставки з низькою затримкою на пошук і часом оновлення для відносно невеликих об'ємів даних . З іншого боку, для оновлення невеликої частини записів в базі даних ефективно використовуються традиційні B-Tree (структури даних в реляційних базах даних, які обмежені по швидкості виконання пошуку). Проте при роботі з більшості баз даних, B-Tree виявились менш ефективними, ніж MapReduce, який використовує сортування – злиття (Sort-Merge) для перебудови бази даних. Тобто, СУБД MapReduce підходить для додатків, де дані записуються один раз, а зчитуються багато разів, в той час як реляційні бази даних, ефективні для обробки наборів даних, які постійно оновлюються.

Реляційні дані часто нормалізуються для забезпечення своєї цілісності і видалення надлишковості. В MapReduce зчитування записів є нелокальною операцією, що робить використання нормалізації проблематичним.

Log веб-сервера є вдалим прикладом великої кількості ненормованих записів – це обґрунтовує зручність аналізу log – файлів будь-якого типу за допомогою MapReduce. MapReduce є моделлю з лінійною масштабованістю [Порівняльний аналіз технології паралельного обчислення великих масивів даних MapReduce, Second International Conference "Cluster Computing", 2013]. Важливо, що для написаних програмістом функцій map та reduce не важливі особливості кластера, на якому здійснюється обробка чи розмір даних, які опрацьовуються. Вони можуть залишатися незмінними як для невеликого набору даних чи для масиву. Особливість в тому, що рішення подвоїти розмір вхідних даних призведе до уповільнення виконання завдання вдвічі (рис.3 а). В

той же час подвоєння розміру кластера поверне попередню швидкість, що не властиво для SQL запитів (рис.3 б). Дані графіки отримані на основі представлених результатів SRA International, Inc. в ході дослідження ефективності технології MapReduce[Tom W, O'Reilly, 2009]. Конфігурації кластерів також важливі для аналізу результатів і приведені нижче. Cluster 1: 2 ГГц Intel Xeon Processor L5508/4GB RAM/2x160GB HD;Cluster 2: 2 ГГц Intel Xeon Processor L5508/4GB RAM/2x160GB HD;Cluster 3: 3,2 ГГц Intel Xeon Processor W5580/16 GB RAM/2x320GB HD.

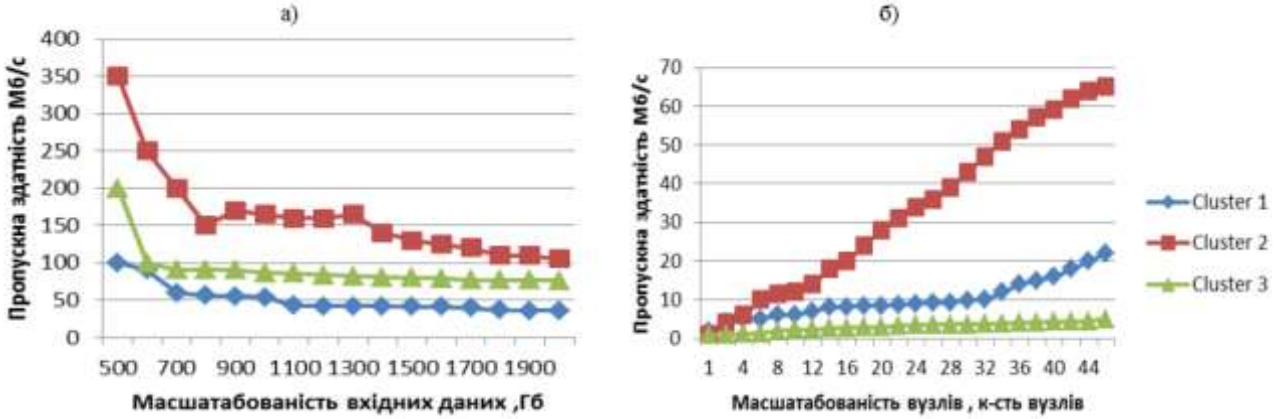

Рис. 1.3 – Графіки залежності пропускної здатності для кластерів різної конфігурації від масштабованості (а) вхідних даних (б) кількості вузлів в кластері

Однак, відмінності між реляційними базами даних і MapReduce системи стираються – як і в реляційних базах даних почали враховувати ідеї з MapReduce (наприклад, бази даних Greenplum, засновані на доопрацьованій PostgreSQL для бази даних з масивно-паралельною архітектурою), так і, з іншого боку, на основі MapReduce розробили мови запитів високого рівня (такі, як Pig і Hive) [Порівняльний аналіз технології паралельного обчислення великих масивів даних MapReduce, Second International Conference "Cluster Computing", 2013].

Високопродуктивні обчислення (HPC) і співтовариство Grid Computing здійснювали великомасштабні обробки даних протягом багатьох років, використовуючи такі API, як Message Passing Interface (MPI). Такий підхід

ефективно працює переважно з інтенсивними обчисленнями, але виникають проблеми, коли вузли повинні отримати доступ до великих обсягів даних (для розмірів від сотні гігабайт), так як пропускна здатність мережі є слабким місцем і обчислювання на вузлах переходить в стан простою. MapReduce розподіляє дані так, щоб вони знаходились поряд з вузлом, який здійснює обчислення, з метою забезпечити швидкий доступ, оскільки доступ до даних є локальним. Всі MapReduce-реалізації строго притримуються цієї властивості явно моделюючи топологію мережі, так як пропускна здатність мережі є найціннішим ресурсом в середовищі дата – центру.

Використання MPI дозволяє програмісту контролювати процес, але натомість вимагає від нього явної обробки механіки потоку даних доступних через сокети та низькорівневі підпрограми C. MapReduce працює тільки на високому рівні [Jon Z., 2010, http://blog.cloudera.com/blog/2010/12/a-profile-of-hadoop-mapreduce-computing-efficiency-continued/]: програміст обдумає функції пари , але при цьому потік даних не є явним. Проблематичною являється координація процесів для великомасштабних розподілених обчислень, особливо, коли мова йде про обробку часткової відмови. За такої відмови продовжує виконуватися загальне обчислення. За рахунок того, що MapReduce має нерозподілену архітектуру, програміст обробляє невиконаний процес та перенаправляє виконання на робочі машини тоді, коли не виконано map чи reduce завдання . Таким чином, з точки зору програміста, порядок, в якому повинна явно контролювати своєю реєстрацією наведення і відновлення. В такому випадку, основний контроль реалізує програміст, що ускладнює написання MPI програм.

В той же час MapReduce програє MPI програмам, коли мова йде про ітеративні обчислення. Приведені графіки отримані на основі представлених даних в ході наукового експерименту департаменту комп’ютерних наук Indiana University. Для аналізу продуктивності MapReduce та MPI програм в контексті цього експерименту розглядаються наступні задачі: кластеризація за допомогою K-means алгоритму та перемноження матриць. Перемноження

матриць ілюструє неефективність здійснення ітеративних обчислення MapReduce програмами (рис.4 а). Пояснюється даний факт відсутністю можливості зберігати статичні дані в пам’яті між викликами. Для кожної ітерації при перемноженні матриць A та B, всі map-завдання отримують на виході: (I) колонки блоку матриці B, і (II), блок рядків матриці; на виході маємо рядок результуючої матриці C. Блок стовпців пов'язаний з конкретним map-завданням, яке фіксується протягом обчислень, в той час як блок рядків змінюються в кожній ітерації. В моделі програмування в Hadoop (типова модель MapReduce), немає можливості зберігати статичні дані між викликами. Таким чином, він завантажує обидва блоки стовпчика і рядка блоку в кожній ітерації обчислень, що знижує продуктивність в порівнянні с MPI-програмами [Борис Т., Second International Conference "Cluster Computing" CC 2013 (Ukraine, Lviv, June 3-5, 2013), 54-57].

K-means кластеризація є наглядним прикладом задач з декількома ітеративними обчисленнями, які виконуються одночасно для загального обчислення. З цієї цілю обрано алгоритм кластеризації колекції 2D точок даних. Для виконання 16 ітерацій для 5.12E+06 точок виконання MPI програми проходить на два порядки швидше (рис.4 б).

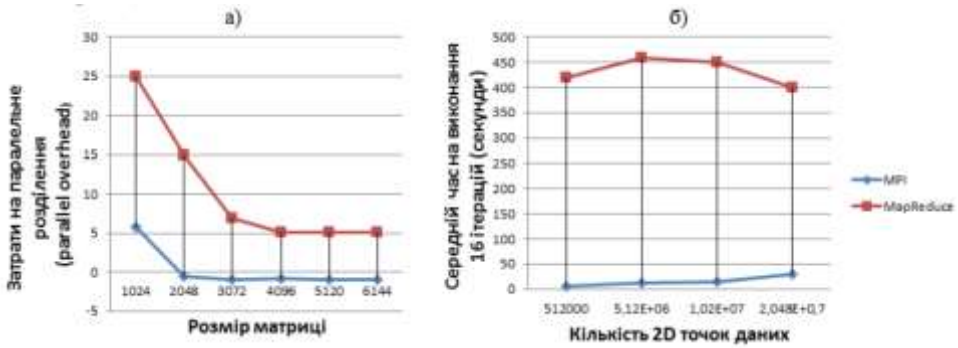

Рис. 1.4 – Затрати на паралельне розподілення програми MapReduce та MPI для перемноження матриць (б) Продуктивність у MapReduce та MPI для K-means кластерних обчислень

Іншим аналогом до MapReduce є Spark – фреймворк, призначений для розподіленої обробки даних з використанням спеціалізованих примітивів, що дозволяють проводити рекурентну обробку в даних в оперативній пам’яті.

Такий підхід дозволяє збільшити швидкість пакетної обробки даних у 100 разів порівняно з MapReduce, в якому відбувається запис даних на диск. Така швидкість, а також можливість багатократного доступу до занесених в пам'ять даних робить Spark привабливим для розробників machine-learning алгоритмів [http://spark.apache.org/docs]. Проте, Spark більше підходить для обробки потокових даних, аніж тих, які вже існують у файловій системі [https://hortonworks.com/apache/spark/]. Також слід зазначити, що Spark оброблює дані швидше за MapReduce, проте це коштує оперативної пам'яті, яку Spark потребує значно більше [Jon Z., 2010, http://blog.cloudera.com/blog/2010/12/a-profile-of-hadoop-mapreduce-computing-efficiency-continued/].

Згідно з визначеними критеріями був здійснений порівняльний аналіз технологій обчислення великих масивів даних результати якого підсумовуються в таблиці 1.5 Представлення проблеми в формі Map-Reduce дозволяє відносно легко розпаралелювати обчислення, направляти дані до процесорів і балансувати навантаження між ними. Деталі всіх цих питань можуть бути приховані від користувача, а можливості паралелізму на рівні задач та рівні команди легко ідентифікуються. MapReduce створена з розрахунку на використання кластерної апаратної архітектури для вирішення задачі паралельної роботи з Великими Даними. Застосування даного підходу обробки даних на противагу традиційним рішенням обґрунтовано такими перевагами як висока продуктивність та можливість опису обробки зрозумілим кодом. Як показав проведений аналіз, можливості коду MapReduce набагато ширші за SQL, навіть без використання спеціалізованих рішень [Борис Т., НТУУ «КПІ», 2013, 54-57].

Представлений огляд даних технологій розподілених обчислень обґрунтовує вибір саме технології MapReduce для здійснення обробки великих масивів даних у формі не ітеративних алгоритмів, де вузли потребують невеликого обміну інформацією (не ітеративні та не залежні). Масивно-паралельні СУБД виграють у роботі з структурованими даними, які часто

записуються, а також за рахунок інтегрованості системи. Надати перевагу MPI системам варто при необхідності контролю процесу в тонкій деталізації та інтенсивних обчисленнях задач, в той час, як MapReduce добре зарекомендував себе в обробці задач з великим об’ємом даних. Також Spark варто застосовувати у випадках, коли необхідний ітеративний доступ до даних, або до попередніх результатів [2017, https://hortonworks.com/apache/spark].

Apache Spark – це швидкий і універсальний кластер обчислювальної системи. Він надає API-інтерфейси високого рівня на Java, Scala, Python і R, а також є оптимізованим двигуном, який підтримує загальні графіки виконання. Цікавим моментом є те, що існують декілька рішень для запуску Hadoop MapReduce з використанням Python [http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/] та [https://blog.matthewrathbone.com/2016/02/09/python-tutorial.html].Він також підтримує багатий набір інструментів вищого рівня, в тому числі Spark SQL для SQL і структурованої обробки даних, MLlib для машинного навчання, Graphx для обробки графів, і Spark Streaming. На високому рівні, кожен додаток Spark складається з керуючої програми, яка запускає основну функцію користувача і виконує різні паралельні операції в кластері. Основна абстракція Spark являє собою пружний розподілений набір даних (ПРНД), який в свою чергу являє собою сукупність елементів розподілених між вузлами кластера, що можуть працювати паралельно. ПРНД створюється починаючи з файлу в файлової системі Hadoop (або будь-який інший Hadoop підтримуваній файловій системі), або в колекціях Scala, і перетворюючи їх. Користувачі також можуть попросити Spark зберігати ПРНД в пам'яті, що дозволяє ефективне повторне використання у паралельних операціях. Також ПРНД має властивість автоматичного відновлення після збоїв [2017, https://hortonworks.com/apache/spark]. Нижче приведений порівняльний аналіз технологій розподілених обчислень [Борис Т., НТУУ «КПІ», 2013, 54-57].

Таблиця 1.5 – Порівняльна характеристика використання MPI, СУБД, MapReduce та Spark

Властивості	**Технології**			
	MapReduce	**Spark**	**Grid**	**СУБД**
Розмір даних	Петабайти	Петабайти	Петабайти	Гігабайти
Пакетний доступ	Так	Так	Так	Так
Масштабованість	Лінійна	Лінійна	Лінійна	Нелінійна
Балансування навантаження	Автоматично (бібліотека MapReduce)	Автоматично (бібліотека Spark)	Програмно (самостійна реалізація)	Так
Локальна оптимізація	Так	Так	Ні	Частково
Оновлення даних	Запис один раз, зчитування багато раз	Запис один раз, зчитування (у більшості випадках) один раз	Записувати та зчитувати багато раз	Переваги в роботі з одиничними запитами, які постійно оновлюютьс я
Паралельне розподілення ресурсів	Автоматично	Автоматично	Програмно	Залежить від архітектури (автоматичн о або програмно)
Тип обчислення	Неітеративні алгоритми	Ітеративні алгоритми	Інтенсивні обчислення	-//-
Робота з неструктурованим и даними	Так	Так	Так	Ні, необхідна нормалізація

Вимоги до спеціаліста	Програміст без досвіду роботи з паралельними та розподіленими системами	Програміст без досвіду роботи з паралельними та розподіленими системами	Вимагає від спеціаліста явної обробки механіки потоку даних доступних через сокети та низькорівневі підпрограми C	Аналогічні до Map операції , попри те що деякі СКБД підтримують визначенні користувачем функції (UDFs), складно представити в виді SQL

Отже, існує декілька альтернатив до MapReduce, які мають свої переваги та недоліки. Технологію Spark важко назвати конкуруючою до MapReduce через те, що вона була створена на базі другої та у ній були враховані певні недоліки MapReduce. Але не дивлячись на це, дана технологія більш доцільна у випадках, коли дані надходять у режимі реального часу, аніж збережені заздалегідь. Також варто зазначити, що Spark завантажує дані в оперативну пам’ять у той час, як MapReduce виконує постійний процес завантаження даних в оперативну пам’ять та запис у файлову систему. Це є основним уповільнювачем MapReduce. Відповідно, тепер необхідно зазначити у яких випадках краще використовувати саме MapReduce.

1.3 Застосування MapReduce у прикладних задачах

Для початку варто розглянути декілька можливостей використання MapReduce для знайомих задач. Першою такою задачею є множення матриці на

вектор. Допустимо, ми маємо матрицю *M* розміру *n x n*, чий елемент в рядку *i* та стовпчику *j* буде позначений як m_{ij}. Також позначемо вектор *v* довжини *n*, чий елемент *j* позначений як v_j. Тоді матрично-векторним добутком буде вектор *x* довжини *n*, чий елемент x_i розраховується за формулою:

$$x_i = \sum_{j=1}^{n} m_{ij} v_j$$

Якщо *n* = 100, непотрібно MapReduce для цього розрахунку. Але цей вид розрахунку лежить в основі ранжування веб-сторінок, що відбувається в пошукових системах, і порядок числа *n* є десятки мільярдів. Припустимо, що *n* велике, але не настільки велике, що вектор *v* не може бути розміщений в основній пам'яті і, таким чином, бути доступними для кожного Map завдання.

Матриця *M* і вектор *v* будуть зберігатися у розподіленій файловій системі. Будемо вважати, що координати рядків стовпців кожного матричного елемента будуть доступні або з їх позиції в файлі, або тому, що вони зберігатимуться з явними координатами, такими як (I, J, M_{ij}). Також припускаємо, що положення елемента v_j в векторі *v* буде виявлятись аналогічним чином [Leskovec J., 2014].

Функція Map написана для застосування до одного елементу з *M*. Однак, якщо *v* ще не завантажене в основну пам'ять на обчислювальному вузлі, що виконує завдання Map, то *v* необхідно спочатку завантажити, і тоді вектор буде доступним для всіх застосування функції Map, що виконується на цьому вузлі. Кожне завдання Map буде працювати з частиною матриці *M*. З кожного матричного елемента M_{ij} вироблятиметься пара ключ-значення $(i, m_{ij}v_j)$. Таким чином, всі члени суми, які складають компонент x_i в матрично-векторному добутку отримають той самий ключ.

У цей час функція Reduce просто сумує всі значення, які відповідають ключу *i*. Відповідно, результатом цієї функції буде пара (i, x_i), що і є шуканим результатом.

Важливим є частковий випадок даної задачі, коли вектор *v* не може бути розміщений у пам'яті. Таким чином, в якості альтернативи, можна розділити

матрицю на вертикальні смуги однакової ширини і розділити вектор на рівну кількість горизонтальних смуг, однією і тією ж висоті. Наша мета полягає в тому, щоб використовувати такого розміру смуги, щоб частина вектору могла бути розміщена в основний пам'яті на обчислювальному вузлі.

Аналогічно до множення матриці на вектор можливо використовувати MapReduce і для інших схожих операцій: множення матриці на матрицю, операції реляційної алгебри, пошук, проекція, об'єднання, перетин та різниця, тощо.

Варто пам'ятати, що MapReduce створений для розподілених обчислень і найбільша ефективність досягається тоді, коли дані, які розміщуються на різних вузлах, є незалежними між собою. Однак можуть виникнути ситуації, коли операції відображення породжують інші операції відображення, так що є певний зв'язок між ними, в результаті чого не так ефективно використовувати паралельність. Як правило, не має бути багато між вузлової взаємодії. Крім того, паралелізм може бути обмежений кількістю робочих вузлів, які мають копію даних. Якщо є п'ять вузлів, яким необхідно отримати доступу до того ж файлу даних, але у існує лише три екземпляри цих даних, тоді два вузла повинні вивантажити дані з іншого вузла. Це призводить до зниження паралелізму і зниження продуктивності. Це вірно і для фази Map і фази Reduce. З іншого боку, три копії даних дозволяють використовувати три додатки для доступу до тих же даних, на відміну від серійних додатків, де є тільки одна копія даних [Layton J., 2012, http://www.enterprisestorageforum.com/storage-management/why-use-mapreduce.html]. Існують думки, що використання MapReduce є оптимальним рішенням для бізнес-середовища [http://www.tutorialspoint.com/articles/advantages-of-hadoop-mapreduce-programming].

Варто також привести декілька життєвих прикладів застосування технології MapReduce:

1) Підрахунок частоти доступу до URL-адреси; функція Map обробляє журнал веб-запитів і генерує пару <*URL, 1*>, функція Reduce додає дані отримані для кожного ключа (*URL*) і генерує пару <*URL, count*>*;*
2) Побудова графу переходів веб-сторінок; функція Map для кожного посилання зі сторінки *source* на сторінку *target* генерує пару <*target, source*>, у той час, як функція Reduce для кожного ключа target генерує пару <*target, list(sources)*>;
3) Пошук головного терміну в документі: цей процес означає пошук слова, яке найбільш часто зустрічається в документі або наборі документів, відповідно, функція Map генерує пару <*назва документу, термін*>, у той час, як функція Reduce рахує кількість однакових термінів для одного документу та залишає лише той, який зустрічається найчастіше [Dean J., Google, Inc., 2004].

Основна проблема даних на основі документів – це неструктурований формат, який може зажадати додаткової обробки. Багато різних записів можуть містити аналогічні дані. Збір і узгодження цієї інформації з метою спрощення її обробки залежить від етапів підготовки і застосування MapReduce.

В системі, заснованої на MapReduce, на етапі перетворення вихідні дані нормалізуються – наводяться до стандартної форми. Цей крок може бути відносно простим (визначення ключових полів або точок даних) або складним (аналіз і обробка інформації для створення вибірки даних). В процесі перетворення дані наводяться до стандартизованого формату, який можна використовувати в якості бази.

Запити до цих даних часто бувають складними – навіть при використанні спеціалізованих інструментів. Ідеальний підхід до інтелектуального аналізу даних полягає в використанні етапу MapReduce в рамках підготовки даних [Brown M., 2013, IBM DeveloperWorks].

Наприклад, при виконанні інтелектуального аналізу даних методом асоціації або кластеризації на першому етапі найкраще побудувати відповідну статистичну модель, яку згодом можна буде застосовувати для виявлення і

вилучення необхідної інформації. Використовуйте етап MapReduce для вилучення і обчислення цієї статистичної інформації з її подальшим введенням в іншу частину процесу інтелектуального аналізу даних, що веде до створення структури, у якій на початковому етапі з великого обсягу даних, використовуючи MapReduce дістаються необхідний, менший набір. А після цього над отриманими даними використовують мову SQL.

Наприклад, за один прохід можна взяти вихідну інформацію з документальної бази даних і виконати операцію MapReduce для отримання короткого огляду цієї інформації по датах. Хорошим прикладом послідовного процесу є регенеріріроваріе інформації та комбінування результатів з матрицею рішень (створюється на другому етапі обробки MapReduce) з подальшим додатковим спрощенням в послідовну структуру. На етапі обробки MapReduce потрібно, щоб весь набір даних підтримував окремі кроки обробки даних.

Інтелектуальний аналіз даних – це не тільки виконання деяких складних запитів до даних, що зберігаються в базі даних. Незалежно від того, чи використовуєте ви SQL, бази даних на основі документів, такі як Hadoop, або прості неструктуровані файли, необхідно працювати з даними, форматувати або реструктурувати їх. Потрібно визначити формат інформації, на якому буде грунтуватися ваш метод і аналіз. Потім, коли інформація знаходиться в потрібному форматі, можна застосовувати різні методи (окремо або в сукупності), які не залежать від необхідної базової структури даних або набору даних. Існують напрацювання по застосуванню MapReduce для кластеризації даних, наприклад [Мансурова М., 2013, Second International Conference "Cluster Computing" , с.54-57], [Zhao W., CloudCom 2009, pp. 674-679] чи [Thakare A., International Journal of Innovative Science, Engineering & Technology, 2015], але ще немає єдиного рішення всіх можливих задач, тому розвиток даного напрямку є актуальним. Також, на сьогоднішній день інтелектуальний аналіз стає невід'ємною частиною бізнесу та корпоративних середовищ [Гаврилова А., Інформаційні технології та системи управління, с. 101][Браун М., 2013, https://www.ibm.com/developerworks/ru/library/ba-datamining/] через необхідність

отримання швидких рішень при наявності величезної кількості необроблених даних.

Відповідно, використання MapReduce для інтелектуального аналізу даних є досить важливим напрямком досліджень. За допомогою MapReduce можливо розбивати одну велику задачу на декілька менших та простіших [2012, http://www.enterprisestorageforum.com/storage-management/why-use-mapreduce.html]. Головне в інтелектуальному аналізі даних за допомогою методу map / reduce – гарантувати, що збирається необхідна інформація, і потрібні поля даних для отримання бажаних відомостей. У методі map / reduce вирішальну роль грає формат map. Ми виводимо ключ і відповідне значення. Значення будуть потрібні на етапі скорочення. Але правильно вибрати значення є дуже важливим рішенням. При обробці тексту значенням може бути результат тематичного аналізу рядків. При аналізі складних даних іноді об'єднують кілька елементів даних; наприклад, при аналізі комерційної інформації можна скомбінувати унікального покупця, товар і місце покупки. Наприклад, при аналізі інформації про продажі відомості про покупців можуть перебувати в одній базі даних, про продажі – в інший, а про товари – в третій. Застосовуючи метод map / reduce до даних з продажу з експортом в іншу базу даних, комбінуючи її з даними про товари і потім знову скорочуючи з наступним об'єднанням з інформацією про покупців, можна згенерувати складні дані, що надходять з різних джерел [Leskovec J., 2014.].

1.4 Огляд фреймворку Hadoop MapReduce

Apache Hadoop є одним з найбільш популярних інструментів для обробки великих даних. Протягом декількох останніх років багато компаній стали використовувати його у виробничому середовищі. Хоча Hadoop вважається надійним, масштабованим і економічно ефективним рішенням, його постійно удосконалює численне співтовариство розробників. В результаті версія 2.0 пропонує кілька революційних функціональних можливостей, включаючи Yet

Another Resource Negotiator (YARN, MapReduce v.2.0), HDFS Federation і NameNode високої готовності, які роблять кластер Hadoop більш ефективним, продуктивним і надійним. У даній статті ви познайомитеся з перевагами, які надає YARN в порівнянні з попередньою версією рівня розподіленої обробки в Hadoop [http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html].

В Apache Hadoop MapReduce 2.0 входить механізм YARN, який відокремлює компоненти управління ресурсами від компонентів обробки. Архітектура YARN не зводиться до MapReduce. У даній статті описується YARN і його переваги перед попереднім рівнем розподіленої обробки в Hadoop. Дізнайтеся, як поліпшити роботу кластерів, використовуючи масштабованість, ефективність і гнучкість YARN.

Apache Hadoop – це програмна інфраструктура з відкритим вихідним кодом, що встановлюється в кластері стандартних машин для спільного розподіленого зберігання і обробки великих обсягів даних. Спочатку Hadoop складалася з двох основних компонентів, які дозволяють реалізувати виконання програм у вигляді завдань MapReduce: розподіленої файлової системи Hadoop (HDFS) і механізму розподілених обчислень.

Також Hadoop надає інфраструктуру ПО для виконання завдань MapReduce у вигляді серій завдань map і reduce. Дуже важливо, що інфраструктура Hadoop бере на себе всі складні аспекти розподіленої обробки: паралелізм, планування, управління ресурсами, взаємодія між машинами, обробку відмов ПЗ і обладнання та багато іншого. Ця абстракція як ніколи раніше спростила реалізацію розподілених додатків, що обробляють терабайти даних на сотнях (або навіть тисячах) машин, навіть для розробників, які не мають досвіду в області розподілених систем.

Хоча було кілька реалізацій моделі MapReduce з відкритим вихідним кодом, найбільш популярною швидко стала Hadoop MapReduce. Крім того, Hadoop є одним з найбільш захоплюючих проектів з відкритим вихідним кодом на планеті, оскільки володіє декількома чудовими функціональними

можливостями, такими як високорівнева програмний інтерфейс, майже лінійна масштабованість, виконання на стандартному обладнанні і відмовостійкість. Він успішно використовується сотнями (а може бути і тисячами) компаній і в даний час є стандартом для високорівневих масштабованих зберігання і обробки [http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html].

Координація всіх завдань, що виконуються в кластері, що включає інструкції процесам TaskTracker по запуску завдань map і reduce, моніторинг виконання цих завдань, повторний запуск завдань після збоїв, випереджаюче виконання повільних завдань, обчислення сумарних значень лічильників завдання і багато іншого.

Велике число обов'язків, доручених одному процесу, призводить до серйозних проблем масштабованості, особливо в великому кластері, де процесу JobTracker доводиться постійно відстежувати тисячі процесів TaskTracker, сотні завдань і десятки тисяч завдань map і reduce. Наведений нижче малюнок ілюструє проблему. У свою чергу процеси TaskTracker зазвичай виконують всього близько десятка завдань, які призначені їм сильно завантаженим процесом JobTracker.

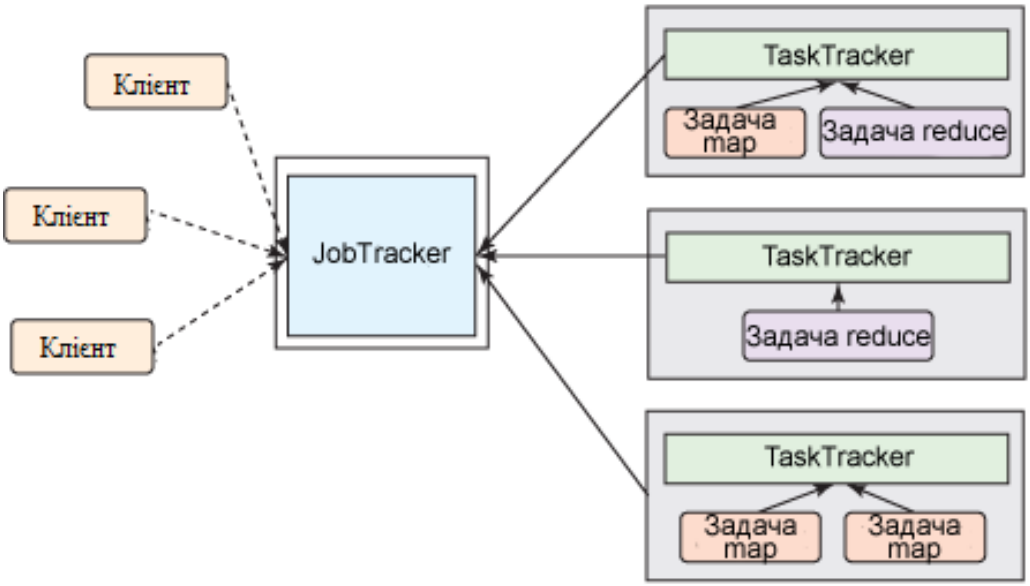

Рис. 1.5 – Класична версія Hadoop MapReduce

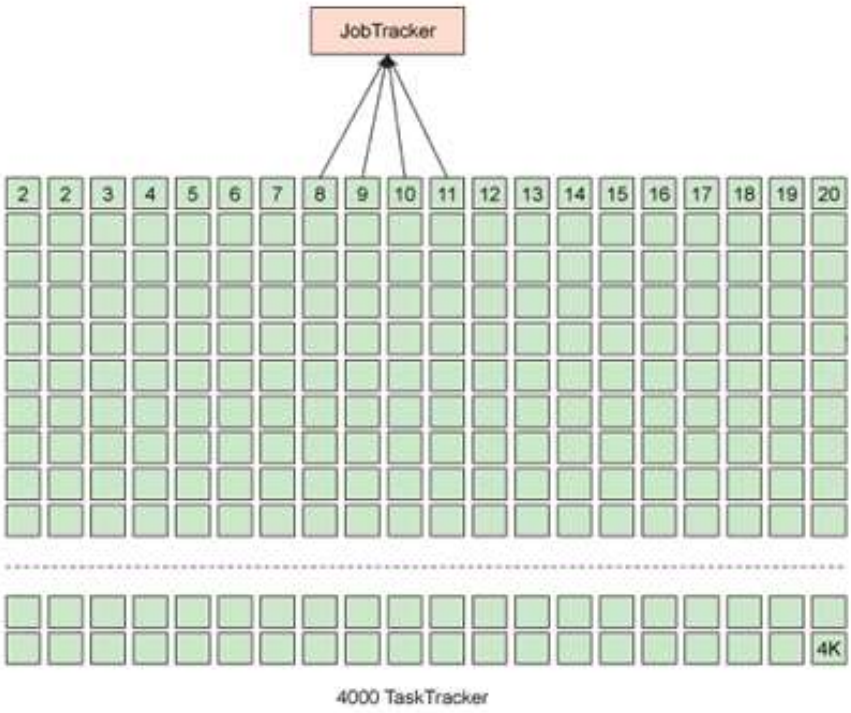

Рис. 1.6 – Навантажений JobTracker у великому кластері Apache Hadoop MapReduce v.1.0

Для вирішення проблеми масштабованості пропонується проста, але ефективна ідея: зняти частину обов'язків з одного процесу JobTracker і делегувати деякі з них процесам TaskTracker, оскільки багато хто з них знаходяться в кластері. Ця концепція знайшла своє відображення в новому дизайні, де обов'язки JobTracker (управління ресурсами кластера і координація завдань) розділені на два види процесів.

Замість одного процесу JobTracker в новому підході вводиться менеджер кластера, що повністю відповідає за відстеження в кластері працездатних вузлів і доступних ресурсів, а також за призначення їх завданням. Для кожного завдання в кластері запускається окремий короткоживучий процес JobTracker для управління виконанням завдань тільки в цьому завданні. Цікаво, що короткоживучі процеси JobTracker запускаються процесами TaskTracker, що виконуються в ведених вузлах. Таким чином координація життєвого циклу завдання розподіляється між усіма доступними машинами кластера. Завдяки такій поведінці більше завдань виконується паралельно, а масшабіруемость різко зростає.

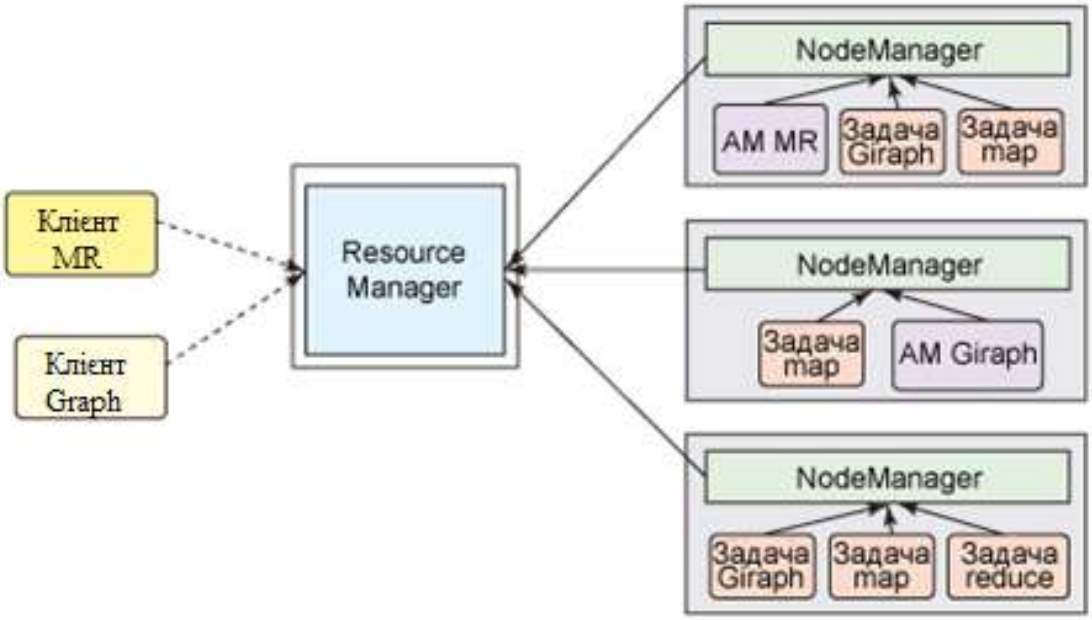

Рис. 1.7 – Архітектура MapReduce v.2.0 YARN

В архітектурі YARN глобальний ResourceManager працює як master-демон (зазвичай на виділеній машині), що розподіляє ресурси кластера між конкуруючими додатками. ResourceManager відстежує працездатні вузли та доступні ресурси в кластері і розподіляє їх між додатками, що запускаються користувачами. Будучи єдиним процесом, що має цю інформацію, ResourceManager приймає рішення про розподіл з урахуванням спільного використання, безпеки та множинної оренди (наприклад, відповідно до пріоритету додатки, місткості черзі, списками управління доступом, розташування даних і т.д.) [http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html].

Коли користувач відправляє додаток на виконання, запускається екземпляр невибагливого до ресурсів процесу ApplicationMaster для координації виконання всіх завдань програми. Сюди входить моніторинг завдань, повторний запуск завдань після збоїв, випереджаюче виконання повільних завдань і обчислення сумарних значень лічильників додатки.

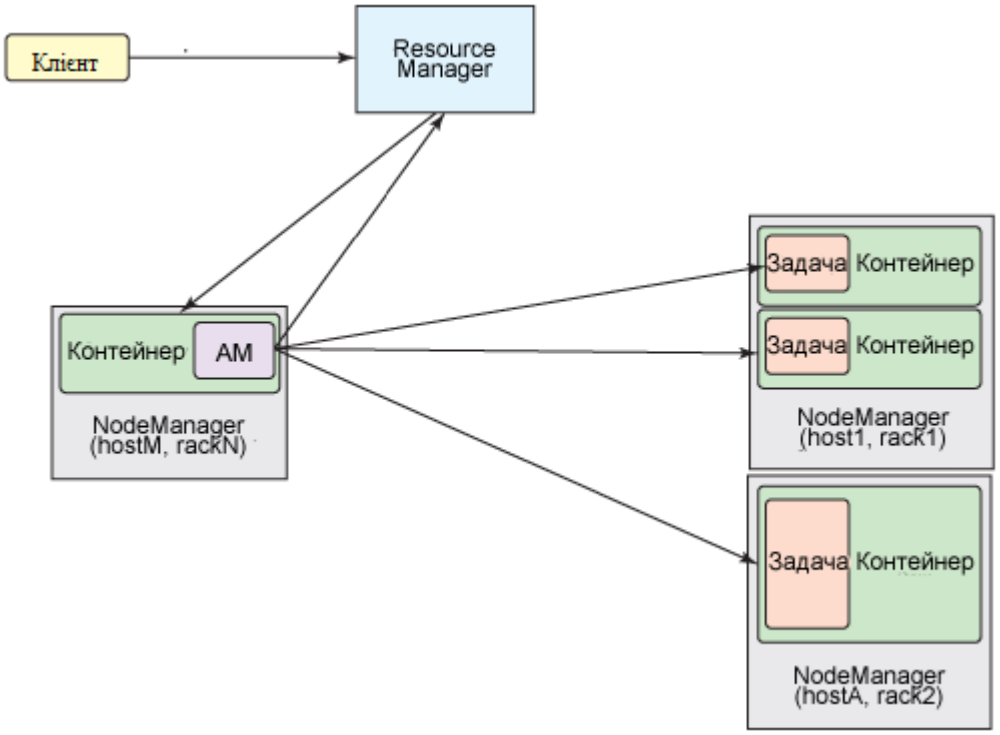

Рис. 1.8 – Запуск додатку у Hadoop MapReduce v.2.0 YARN

Припустимо, що користувачі запускають додатки в ResourceManager за допомогою команди hadoop jar по аналогії з MRv1. ResourceManager обробляє список виконуються в кластері додатків і список доступних ресурсів в кожному працездатному NodeManager. ResourceManager повинен визначити додаток для отримання наступної порції ресурсів кластера. На це рішення впливає багато обмежень, наприклад місткість черзі, списки управління доступом і равнодоступность ресурсів. ResourceManager використовує підключається планіровщіе (Scheduler). Scheduler займається тільки плануванням, визначаючи черговість отримання ресурсів кластера (у вигляді контейнерів), але не виконує моніторинг завдань в додатку і не намагається перезапустити завдання після збоїв.

Коли ResourceManager підтверджує запуск нової програми, одним з перших рішень Scheduler є вибір контейнера, в якому буде виконуватися ApplicationMaster. Запущений ApplicationMaster відповідає за весь життєвий цикл цього додатка. В першу чергу це буде передача в ResourceManager запиту ресурсів у вигляді контейнерів для виконання завдань програми.

YARN пропонує очевидні переваги в масштабованості, ефективності та гнучкості в порівнянні з класичним механізмом MapReduce в першій версії Hadoop. Виграш від використання YARN отримують як маленькі, так і великі кластери. Для кінцевого користувача (розробника, а не адміністратора) ці зміни майже непомітні, оскільки новий механізм дозволяє виконувати немодифіковані завдання MapReduce, використовуючи ті ж самє програмні інтерфейси MapReduce і інтерфейси CLI [Brown M., 2013, IBM Developer Works]. Також існують дослідження, які доводять ефективність використання Hadoop MapReduce [http://blog.cloudera.com/blog/2010/12/a-profile-of-hadoop-mapreduce-computing-efficiency-continued/].

1.5 Висновок

Можливість запуску процесу на звичайному, недорогому обладнанні є перевагою при роботі з великими даними, більша частина яких у далекій перспективі можуть зовсім знецінитись. Тобто, розділення задачі на невеликі фрагменти дозволяє виконати її дешевше та швидше.

MapReduce – це є не база даних, і не її замінник, але ця модель дає можливість більш ефективно коритуватись базами даних, особливо на підприємствах. Після того, як за допомогою цієї моделі будуть отримані важливі фрагменти з потоку великих даних, ці фрагменти можуть бути розміщені в традиційну базу даних, що дозволить отримати широкий доступ до запитів та звітів.

Реляційні бази даних, хмарні технології та модель MapReduce не є конкурентними технологіями, а є частиною аналітичної екосистеми, яка дозволить отримати необхідну інформацію з великих даних. Всі три технології можуть бути використані для досягнення максимальних результатів. Існує багато різних способів об’єднання цих технологій, наприклад, розробка баз даних, які працюють в хмарі, а дані перед обробкою мовою SQL фільтруються за допомогою MapReduce, або навпаки, тощо. Також варто відмітити, що грід-

обчислення також можуть бути частиною цієї екосистеми, якщо спроектувати її правильним чином [Френкс Б., Манн, Иванов и Фербер, 2014, с. 146]. Не варто розділяти ці технології, варто проаналізувати, як саме краще їх використовувати разом.

Для реалізації MapReduce у дипломному досліджені було обрано систему Hadoop MapReduce v.2.0 YARN з мовою програмування Java. Ця система пропонує переваги в масштабованості, ефективності та гнучкості в порівнянні з класичним механізмом MapReduce в першій версії Hadoop, а мова програмування Java є більш ефективною у даній роботі, вона дозволить уникнути помилок із пам’ятю, переповненням буферу та інших, які є поза межами даного дослідження.

Важливим напрямком досліджень на сьогоднішній день є інтелектуальний аналіз даних (Data Mining). Важливе положення Data Mining – не тривіальність розшукуваних шаблонів. Це означає, що знайдені шаблони повинні відображати неочевидні, несподівані регулярності в даних, складові так званих прихованих знань. До суспільства прийшло розуміння, що необроблені дані містять глибинний пласт знань, при грамотній розкопці якого можуть бути знайдений справжні самородки [Петренко А., http://allted.kpi.ua/downloads/DataMining.pdf]. Технологія MapReduce добре адаптується для вирішення задач DataMining. В наступних розділах цієї роботи будуть представлені моделі обробки великих даних за допомогою технології MapReduce, а також описані результати практичної реалізації цих моделей.

2 МОДЕЛІ ДЛЯ ОБРОБКИ ВЕЛИКИХ ДАНИХ НА ОСНОВІ ТЕХНОЛОГІЇ MAPREDUCE

2.1 Задача кластеризації

Для дослідження великої сукупності даних, що дозволяють скласти висновок про рівень корпоративного управління у відповідних структурах вітчизняної економіки, велике значення приділяється застосуванню нових інформаційних технологій в галузі математичної статистики та аналізу. Кластеризація, як перший етап аналізу даних, здійснюється на підставі певних методів, які мають відображення в алгоритмах розбиття груп об'єктів. Виділення груп дозволяє спростити роботу з даними, після кластеризації застосовуються інші методи, для кожної групи будується окрема модель [Leskovec J., 2014., 242 p].

Постановка задачі кластеризації для всіх алгоритмів полягає в наступному:

Дано: X – простір об'єктів; $X_l=\{x_i\}_{i=1}^{l}$ – вибірка елементів; $d: X \times X \to [0; \infty)$ – функція відстані між об’єктами.

Знайти: Y – множину кластерів і відображення $a: X \to Y$ – алгоритм кластеризації такий, що кожен кластер складається з близьких між собою об’єктів, а об’єкти різних кластерів суттєво відрізняються [Волосюк Ю., Інформаційні технології, с. 112].

Кластеризація являє собою процес вивчення колекції «точок» і їх угрупування в «кластери» відповідно до деякої міри відстані. Мета полягає в тому, що точки в одному кластері мають невелику відстань один від одного, в той час як точки в різних кластерах знаходяться на великій відстані один від одного.

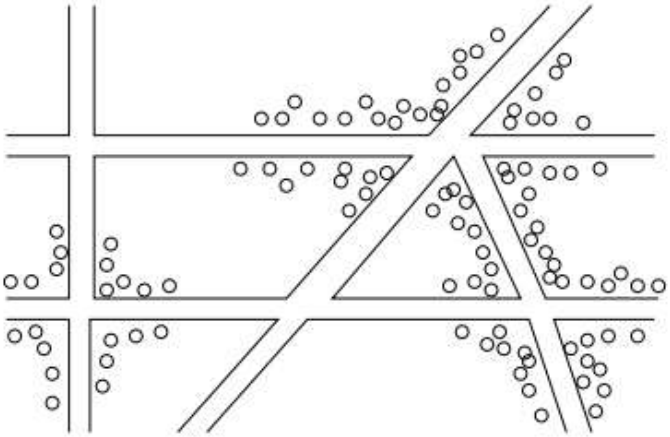

Рис. 2.1 – Координати хворих на карті Лондона

Кластери, наприклад, можуть виглядати як показано на рис. 2.1. На рисунку видно, що є три кластери навколо трьох різних дорожніх перетинів, але два кластери недостатньо розділені і знаходяться поряд один з одним.

Набір даних, які підходять для кластеризації є набором точок, які є об'єктами, що належать до деякого простору. У найзагальнішому випадку, простір це просто універсальний набір точок. Тим не менш, потрібно пам'ятати про загальний випадок евклідового простору, який має ряд важливих властивостей корисних для кластеризації. Зокрема, точка у евклідовому просторі є вектором дійсних чисел. Довжина вектора є число вимірів простору. Компоненти вектора зазвичай називають координатами точок [Leskovec J., 2014].

Всі простори, для яких можна виконати кластеризацію мають міру відстані, що дає відстань між будь-якими двома точками в просторі. Існують різні підходи для визначення відстані. Загальна евклідова відстань (квадратний корінь з суми квадратів різниць між координатами точок в кожному вимірі) служить для всіх евклідових просторів, хоча також існують деякі інші варіанти відстані в евклідових просторів, в тому числі – Манхеттенська відстань (сума модулей різниць в кожному вимірі) і L_∞-відстань (максимальна величина різниці в будь-якому вимірі).

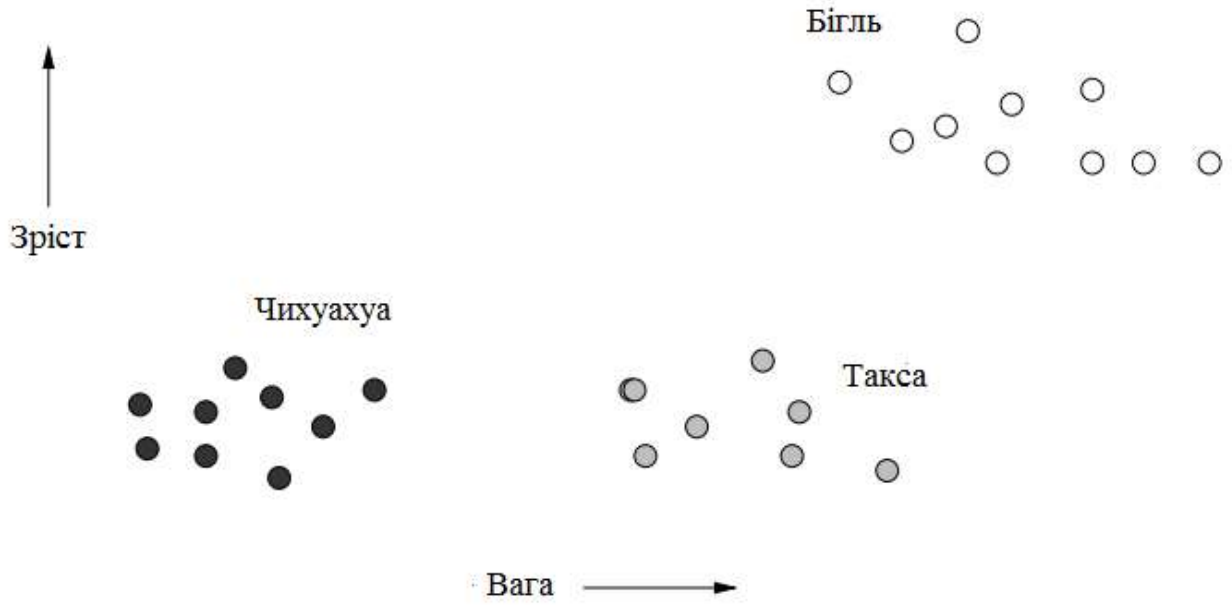

Рис. 2.2 – Зображення вибірки із трьох порід собак за параметрами зросту та ваги

Не знаючи, яка собака має яку породу, можна побачити, просто подивившись на діаграмі, що собаки діляться на три кластери, і ці кластери дозволяють поділити загальну вибірку на три групи. При невеликих обсягах даних, будь-який алгоритм кластеризації встановить правильні кластери, і просто зображення точок допоможе визначити, до якого кластеру належить той, чи інший об'єкт [Leskovec J., 2014].

Однак, сучасні задачі кластеризації не такі прості. Вони можуть включати в себе евклідовий простір дуже високої розмірності або простір, який не є евклідовим взагалі. Наприклад, складною задачею є кластеризація документів по їх темі, на основі виникнення загальних, незвичайних слів в документах. Іншим складним прикладом є кластеризація кіноманів по типу або типами фільмів, які їм подобаються.

Алгоритми кластеризації можна розділити на дві групи, які слідують дві принципово різні стратегії:

- Ієрархічні або агломераційні алгоритми починаються з кожною точкою в своєму власному кластері. Кластери об'єднуються на основі їх близькості за допомогою однієї з багатьох можливих визначень цього поняття.

Алгоритм зупиняється, коли подальше поєднання призводить до кластерів, які є небажаними по з кількох причин. Наприклад, можна зупинитися, коли заздалегідь визначено кількість кластерів, або можна використовувати міру компактності для кластерів, і відмовлятись побудувати кластер шляхом об'єднання двох невеликих кластерів, якщо в результаті кластер матиме точки, які розкидані по занадто великій області.

- Інший клас алгоритмів пов'язаний з точкою призначення. Точки розміщуються і якомусь порядку, і кожна з них призначається кластеру, в який вона найкраще підходить. Цьому процесу, як правило, передує короткий етап, на якому оцінюються початкові кластери. Варіації дозволяють випадкові об'єднання або розщеплення кластерів, або можуть дозволити точки, які не будуть призначеними до якогось кластера, якщо вони є викидами (точки занадто далеко від будь-якого з поточних кластерів).

Класичні ієрархічні алгоритми працюють тільки з категорійними атрибутами, коли будується повне дерево вкладених кластерів. У них проводиться послідовне об'єднання вихідних об'єктів і відповідне зменшення числа кластерів. Неієрархічні алгоритми ґрунтуються на оптимізації деякої цільової функції, що визначає оптимальне, в певному сенсі, розбиття множини об'єктів на кластери. У цій групі популярні алгоритми сімейства k-середніх (k-means), які в якості цільової функції використовують суму квадратів зважених відхилень координат об'єктів від центрів кластерів, які шукають. Кластери шукають сферичної або еліпсоїдної форми [Leskovec J., 2014, p. 262].

2.1.1 Опис алгоритму CURE

У цій частині буде описаний алгоритм, в основі якого лежить призначення точок. Цей алгоритм називається CURE (Clustering Using REpresentatives), кластеризація за допомогою репрезентативної вибірки. У даному підході передбачене використання евклідового простору. Важливо

відмітити, що у даному алгоритмі не грає роль форма кластерів; вони не повинні бути розподілені нормально, і навіть можуть мати дивні вигини, S-форму, або навіть кільце. Замість подання кластерів їх центроїда, він використовує набір репрезентативних точок, що і випливає з назви.

CURE є ієрархічним алгоритмом кластеризації. В основному CURE покладений ієрархічний алгоритм кластеризації, який використовує розбиття даних. Поєднання випадкової вибірки і поділу використовується тут так, що велика база даних можуть бути оброблена. У процесі роботи цього алгоритму випадково взяту з набору даних вибірку спочатку розділяють, а потім кожну отриману частину частково групують. Часткові кластери потім знову групуються в другому проході, з отриманням бажаних кластерів. Підтверджується експериментами, що якість кластерів, отриманих за допомогою алгоритму CURE набагато краща, ніж за допомогою інших алгоритмів ієрархічної кластеризації, адаптованих до обробки великих наборів даних [Thakare A., 2015, pp.605-614].

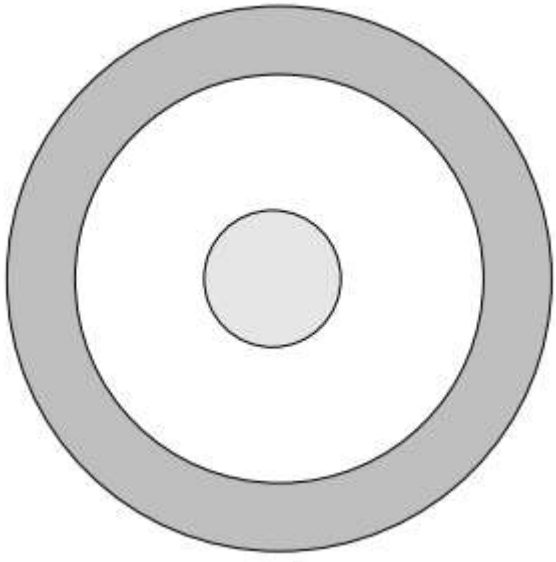

Рис. 2.3 – Два кластери, один навколо іншого

На рисунку 2.3 проілюстровано два кластери. Внутрішній кластер являє собою звичайний коло, другий кластер – кільце, яке знаходиться навколо першого кола. Таке розташування не є повністю патологічним. Істота з іншої галактики могли б подивитися на нашу Сонячну систему і спостерігати, що

об'єкти кластера утворюють собою внутрішнє коло (планети) і зовнішні кільця (пояс Койпера), з невеликим проміжком між ними [Leskovec J., 2014].

Алгоритм CURE складається з наступних кроків на початковому етапі:

- Необхідно взяти невелику вибірку даних і кластеризувати її в основний пам'яті. В цілому, будь-який метод кластеризації може бути використаний, але, як CURE призначений для обробки кластерів незвичайної форми, часто доцільно використовувати ієрархічний метод, в якому кластери поєднуються, коли вони мають близьку пару точок. Далі у прикладі буде розглянуто цей крок.
- Вибираємо невеликий набір точок з кожного кластера, щоб ці точки стали репрезентативними для кластера. Ці точки повинні бути обрані так далеко один від одного, наскільки це можливо.
- Переміщення кожної з репрезентативних точок на фіксованої частку відстані між її розташування і центроїдом кластеру. Зазвичай, обирають значення 20% для цього переміщення. Важливо відмітити, що цей крок вимагає евклідового простору, так як у противному випадку, не може бути поняття лінії між двома точками.

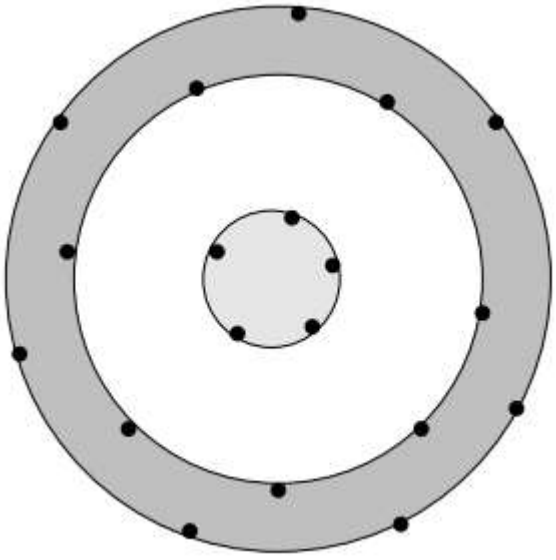

Рис. 2.4 – Вибір репрезентативних точок у кожному з кластерів

На другому етапі необхідно обрати репрезентативні точки. Якщо зразок, з якого побудовані кластери досить великий, можна розраховувати на вибіркових

точках в кластері по найбільшій відстані одного від іншого, що лежить на кордоні кластера. На рисунку 2.4 зображено, що початковий вибір точок вибірки може виглядати наступним чином.

На останньому етапі, необхідно змістити точки на фіксовану частку відстані від їх справжнього місця розташування до центроїду кластера. Слід зазначити, що на рис. 2.4 обидва кластера мають свій центр ваги в одному і тому ж місці: центр внутрішнього кола. Таким чином, зображені точки будуть рухатися всередину кластера, як і було зазначено раніше.

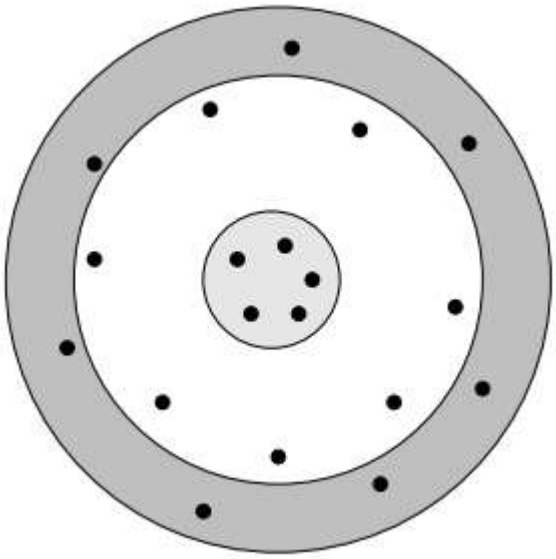

Рис. 2.5 – Переміщення репрезентативних точок на 20% у напрямку центроїда

Точки на зовнішній границі кільця також переміщаються в напрямку центроїда їх кластеру і будуть розміщені в кластері. В той же час – точки на внутрішній границі кільця після переміщення будуть за межами кластеру. Остаточні місця представницьких точок з рис. 2.4 запропоновані на рис. 2.5.

Наступний етап CURE полягає в об'єднанні двох кластерів, якщо вони мають пару репрезентативних точок, по одному від кожного кластера, які досить близькі. Користувач може вибрати відстань, яка буде визначати близькість кластерів. Цей крок злиття можна повторювати, поки не буде більше досить близьких кластерів [Leskovec J., 2014].

Положення на рис. 2.5 служить корисної ілюстрацією. Існує певний аргумент, що кільце і коло дійсно повинні бути об'єднані, тому що їх центр ваги

однакові. Наприклад, якщо відстань між кільцем і кругом була б набагато менше, це могло б також бути передумовою того, щоб об'єднання точок кільця і кола в один кластер мала місце. Наприклад, кільце Сатурна має вузькі проміжки між ними, але розумно візуалізувати кільця як єдиний об'єкт, а не кілька концентричних об'єктів. В цьому випадку вибір складається з наступних елементів: необхідно обрати частину відстані до центроїду, на яку перемістити репрезентативні точки та вибір того, як далеко один від одного репрезентативні точки двох кластерів повинні бути, щоб уникнути злиття цих кластерів. Тобто, користувач має задати ці константи перед початком роботи, або ці константи можуть бути розраховані автоматично за наявності необхідних для цього алгоритмів. Останній крок CURE алгоритму – це призначення точок. Кожна точка *p*, яка надходить з вторинного місця зберігання порівнюється з репрезентативними точками. Додамо точку *p* тому кластеру, для репрезентативних точок якого дана точка є найближчою.

Алгоритм CURE знаходить кластери з великої бази даних, яка є більш стійкою до викидів, і ідентифікує кластери, які мають несферичні форми і широкі відхилення в розмірах. CURE використовує комбінацію збору даних, обробки даних за допомогою випадкової вибірки і розбиття. При наявності великих масивів даних в таких прикладних областях, як біоінформатика, медична інформатика, аналіз наукового даних, фінансовий аналіз, телекомунікації, роздрібна торгівля і маркетинг, подібні алгоритми стають все більш необхідними для досягнення цілей інтелектуального аналізу, а саме – паралельно і швидко з використанням розподілених систем [Thakare A., 2015, pp.605-614].

2.1.2 Розподілена реалізація алгоритму CURE з використанням MapReduce

Для більшої ефективності алгоритму CURE необхідно внести деякі зміни, які будуть працювати в розподіленої системі, і в цьому випадку може бути використана технологія MapReduce.

Загальна модель представлена на малюнку 2.6. Вона складається з *N* вузлів, які керують програмою Map і *1* вузла, який запускає програму Reduce. Частини загального набору даних, вводяться в програму Map. Програма Map завершує свою роботу після того, як повний набір даних на цьому вузлі був оброблений.

Вихід програми Map являє собою набір пар ключ-значення, де ключ дорівнює *1*, а значення являє собою набір представницьких точок з кластера *(I, J)*. В цьому випадку *I* – це номер вузла і *J* – це номер кластера з цього вузла. Наприклад, якщо на вузлі *2* є *3* кластери даних, результат повинен бути наступний: ключ – *1*, значення – кластер *(2,1)*; ключ – *1*, значення – кластер *(2,2)*; ключ – *1*, значення – кластер *(2,3)* [Yaremenko V., SAIT 2017, p. 204].

Як показано на рисунку 2.6, програма Reduce отримує вихід програми Map. Основне завдання програми Reduce є злиття кластерів, отриманих з усіх наборів даних. Після того, як виконається процес злиття, програма Reduce видає результуючий набір даних, який містить опис кожного кластера: центроїд і репрезентативні точки.

Перед початком кластеризації необхідно обрати три константи, а саме – критичну відстань, ближче якої кластери об'єднуються, відсоток відстані до центроїда кластеру, на який необхідно буде здвинути репрезентативні точки, а також – кількість репрезентативних точок. Програма Map працює за наступним алгоритмом:

1) Завантажити частину даних в оперативну пам’ять;
2) Провести ієрархічну кластеризацію даних у пам’яті;
3) Порахувати репрезентативні точки та центроїди отриманих кластерів;
4) Переміщення репрезентативних точок в напрямку центроїда на таку відстань, яка задана константами;
5) У випадку, якщо ще є дані на цьому обчислювальному вузлі, повернутись до пункту 1, у протилежному випадку – перейти до пункту 6;

6) Для кожного отриманого кластеру на цьому вузлі необхідно створити пару ключ-значення, де ключ – це 1, а значення – опис кластеру (набір репрезентативних точок, центроїд);
7) Відправити отриману пару ключ-значення до програми Reduce.

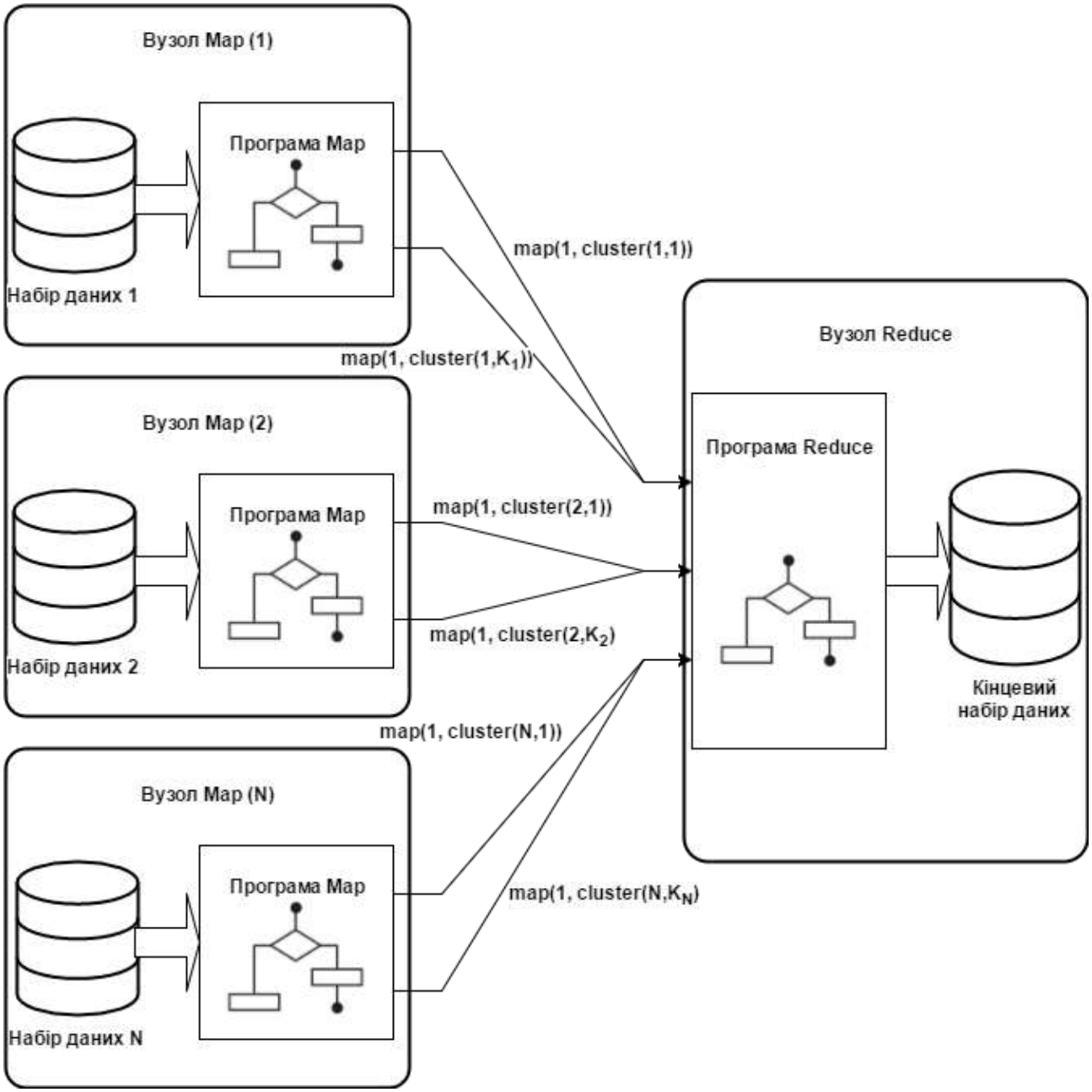

Рис. 2.6 Модель кластеризації даних за допомогою технології MapReduce

У той же час – програма Reduce працює за іншим алгоритмом:

1) Для кожного отриманого кластеру перевірити, чи він може бути об'єднаний із одним з існуючих, якщо так – крок 2, якщо ні – крок 5;
2) Об'єднати два кластери;

3) Порахувати нове значення центроїду;
4) Обрати репрезентативні точки та здвинути їх у напрямку центроїда;
5) Зберегти новий кластер;
6) Для кожного отриманого кластеру – додати список репрезентативних точок до загального виходу програми.

Після виконання цих кроків програма MapReduce завершує свою роботу і отримані точки на кроці 6 програми Reduce і будуть рішенням задачі кластеризації.

Перевага описаного підходу в розподілених обчисленнях окремо на кожному вузлі. Крім того, немає необхідності завантажувати повний набір даних в оперативну пам'ять, так як програма Map обробляє частину даних правильним чином [Yaremenko V., SAIT 2017, p. 204].

2.2 Задача пошуку частих предметних наборів

Метою цього аналізу може бути пошук закономірностей в бізнес-процесах, виявлення зв'язків між продажами окремих товарів і їх груп, виявлення типових шаблонів покупок, прогнозування обсягів продажів і так далі. Знання, отримані в процесі такого аналізу, дозволяють приймати рішення, спрямовані на поліпшення роботи компанії за найрізноманітнішими напрямками: оптимізація закупівель, зниження витрат на доставку товарів, виявлення цільових груп клієнтів, визначення цінової політики та маркетингової стратегії, стимулювання попиту і багато іншого.

Однією з головних проблем, пов'язаних з пошуком асоціативних правил, є необхідність обробки великої кількості транзакцій, кожна з яких містить безліч предметів (товарів). Якщо робити це за допомогою прямого перебору, то на практиці доведеться розглянути безліч можливих асоціацій, що робить задачу нерозв'язною. Для вирішення даної проблеми у 1994 році Р. Агравал і Р. Шрікант запропонували алгоритм пошуку асоціативних правил на

транзакційних базах даних, що отримав назву Apriori [Орешков В., https://basegroup.ru/community/articles/sequential-patterns-1].

Для розширення можливостей аналізу транзакційних даних з урахуванням тимчасового аспекту, послідовності появи предметів і орієнтованості на конкретного клієнта існує задача Data Mining під назвою послідовні шаблони (sequential pattern, time-serial sequential pattern).

Теорія послідовних шаблонів багато в чому заснована на теорії асоціативних правил і, по суті, є її розширенням. Зокрема, базовими поняттями в ній також є транзакція, предметний набір, частота набору, підтримка і т.д. Крім цього, для пошуку послідовних шаблонів широко використовується адаптований алгоритм Apriori і його модифікації. Але при розгляді послідовних шаблонів необхідно враховувати ряд особливостей. Головна з них полягає в тому, що якщо в асоціативних правилах розглядається тільки факт спільного появи товарів в одній транзакції, то послідовних шаблонах розглядається послідовність появи товарів. Послідовний шаблон – це завжди послідовність появи предметів і їх груп.

Інтуїтивно зрозуміло, що типовою послідовністю (шаблоном) може бути тільки така послідовність, яка зустрічається в базі даних досить часто. Тому при пошуку послідовних шаблонів виникає та ж проблема, що і при пошуку асоціативних правил. Велике число розглянутих предметів породжує величезну кількість можливих послідовностей, що призводить до серйозних обчислювальних витрат при використанні повного перебора.Но і тут можна застосувати принцип антімонотонності – послідовності, що містять рідкісні події, не можуть бути частими, що дозволяє істотно знизити простір пошуку.

Застосування послідовних шаблонів виходить за рамки аналізу ринкового кошика. Вони дозволяють виявляти типові послідовності подій в найрізноманітніших предметних областях. Наприклад, послідовність може містити такі події, як взяття кількох фільмів клієнтом в відеопрокаті. Якщо фільм являє собою трилогію, наприклад, "Зоряних воєн", то типовою послідовністю, в якій клієнт буде брати ці фільми - 1-ша частина, 2-га частина і

3-я частина. При цьому послідовність необов'язково повинна бути безперервною. У проміжку клієнт може взяти й інший фільм, але типова послідовність при цьому збережеться. Аналіз всіх послідовностей фільмів, взятих напрокат певними клієнтами, може виявити найбільш типові шаблони. Знання таких послідовностей допоможе стимулювати клієнтів, пропонуючи їм фільми в порядку, встановленому шаблоном.

Розглянемо постановку задачі пошуку послідовних шаблонів, використовуючи завдання аналізу ринкової кошика. Саме так вона була спочатку сформульована в основоположною статті Р. Агравал і Р. Шріканта "Mining Sequential Patterns", що дозволить провести аналогії з асоціативними правилами і виявити найбільш істотні відмінності. При цьому будемо дотримуватися термінології першоджерела.

Нехай є база даних *D*, в якій кожен запис являє собою клієнтську транзакцію. Кожна транзакція містить наступні поля: ідентифікатор клієнта, дата / час транзакції, і набір куплених товарів. Введемо обмеження, що жоден клієнт не має двох або більше транзакцій, здійснених в один і той же момент часу [Орешков В., https://basegroup.ru/community/articles/sequential-patterns-1].

Введемо кілька основних понять. Предметний набір – це непорожній набір предметів (товарів), що з'явилися в одній транзакції.Последовательность – це впорядкований список предметних наборів. Послідовність будемо укладати в трикутні дужки, а предметний набір - в круглі. Тоді, якщо позначати предмети цілими числами, то предметні набори будуть записані у вигляді *(2, 4, 5), (1, 3)*, а послідовність, що містить ці набори *<(2, 4, 5), (1, 3)>* . Якщо предмети з'явилися в одному наборі, це означає, що вони були придбані одночасно.

Позначимо предметний набір $I = (i_1, i_2, ..., i_m)$ (від англ. Itemset – предметний набір, набір елементів) де i_j – предмет. Позначимо послідовність $S = I_1, I_2, ..., I_m$, де I_i – предметний набір.

Послідовність S_1 міститься в іншій послідовності S_2, якщо все предметні набори S_1 містяться в предметних наборах S_2. Наприклад, послідовність *<(3);*

(4; 5); (8)> міститься в послідовності *<(7); (3,8); (9); (4,5,6); (8)>*, оскільки *(3) ⊆ (3,8), (4,5) ⊆ (4,5,6) і (8) ⊆ (8)*. Однак, *<(3); (5)> ⊄ <(3,5)>* і навпаки, оскільки в першій послідовності предмети *3* і *5* були куплені один за іншим, а в другій – куплені спільно.

Послідовність *S* називається максимальною, якщо вона не міститься в будь-якій іншій послідовності. Усі транзакції одного клієнта можуть бути показані у вигляді послідовності, в якій вони впорядковані за датою, часу або номеру візиту. Такі послідовності будемо називати клієнтськими. Формально це записується в такий спосіб [Орешков В., https://basegroup.ru/community/articles/sequential-patterns-1].

Нехай клієнт зробив кілька впорядкованих в часі транзакцій $T_1, T_2, ..., T_3$. Тоді кожен предметний набір в транзакції T_i позначимо $I(T_i)$, а кожну клієнтську послідовність для даного клієнта запишемо як $I(T_1), I(T_2), ..., I(T_3)$. Іншими словами, клієнтська послідовність – це послідовність предметних наборів, що містяться у всіх транзакціях, скоєних даним клієнтом.

Послідовність S називається підтримуваної клієнтом, якщо вона міститься в клієнтської послідовності даного клієнта. Тоді підтримка послідовності визначається як число клієнтів, що підтримують дану послідовність. Таким чином, поняття підтримки в теорії послідовних шаблонів дещо відрізняється від аналогічного поняття теорії асоціативних правил.

Для бази даних клієнтських транзакцій завдання пошуку послідовних шаблонів полягає в виявленні максимальних послідовностей серед всіх послідовностей, що мають підтримку вище заданого порогу. Кожна така максимальна послідовність і є послідовний шаблон. Далі будемо називати послідовності, що задовольняють обмеження мінімальної підтримки, частими послідовностями (по аналогії з часто зустрічаються предметними наборами або популярними наборами в теорії асоціативних правил) [Орешков В., https://basegroup.ru/community/articles/sequential-patterns-1].

Додатково введемо кілька термінів. Довжина послідовності – це число предметних наборів, яке в ній міститься. Послідовність довжини *k* будемо

називати *k*-послідовністю. Підтримкою предметного набору *I* є число клієнтів, які придбали предмети в одній транзакції. Таким чином, предметний набір *I* і *1-послідовність* <*I*> мають одну і ту ж підтримку. Нагадаємо, що в асоціативних правилах предметний набір, що задовольняє рівню мінімальної підтримки, називається частим. Звернемо увагу, що будь-який предметний набір в частій послідовності також повинен бути частим, на що вказує властивість антімонотонності. Отже, будь-яка часта послідовність буде складатися тільки з частих предметних наборів.

2.2.1 Пошук частих предметних наборів за допомогою алгоритму SON

Алгоритми для частих предметних наборів, що обговорювалися досі використовують один прохід для кожного набору даних. Якщо основна пам'ять занадто мала для зберігання даних, необхідного для підрахунку частих наборів одного розміру, не можна тоді якимось чином уникнути *K* проходів, щоб знайти правильне рішення. Проте, існує безліч сфер застосування цього типу алгоритмів, де не є важливим виявлення всіх частих наборів. Наприклад, якщо ми шукаємо товари, придбані разом в супермаркеті, ми не збираємося робити якісь висновки, що засновані на кожному частому наборі. Необхідно лише знайти більшість таких наборів.

У цьому розділі буде розглянутий алгоритм, який дозволяє знайти всі або найбільш часто зустрічаємі набори, використовуючи не більше двох проходів. Алгоритм називається SON (який названий на честь містерів Савасере, Оміецински та Навате) використовує два проходи, отримує точну відповідь, і, як буде показано у наступному розіділі, піддається реалізації за допомогою MapReduce або іншого паралельного обчислювального режиму.

Для початку необхідно описати роботу простого, рандомізованого алгоритму. Замість того щоб використовувати весь файл кошиків, можна було б вибрати випадкову підмножина кошиків і робити вигляд, що це весь набір даних. Потрібно регулювати поріг підтримки, щоб відобразити меншу кількість кошиків. Наприклад, якщо поріг підтримки повного набору даних *s*, і ми

вибираємо за зразок *1%* кошиків, то ми повинні досліджувати зразок для наборів, які з'являються щонайменше, *S / 100* кошиків [Leskovec J., 2014, p. 228].

Найбезпечніший спосіб вибрати зразок – це прочитати весь набір даних, і для кожної корзини вибрати кошик з деякою фіксованою ймовірністю *p*. Припустимо, що існує *m* кошиків у всьому файлі. Зрештою, ми будемо мати зразок, розмір якого дуже близький до $p \times m$ кошиків. Однак, якщо у нас є підстави вважати, що кошики з'являються у випадковому порядку в файлі, то не потрібно навіть читати весь файл. Можна вибрати перші $p \times m$ кошиків для нашої вибірки. Або, якщо файл є частиною розподіленої файлової системи, можна вибрати деякі шматки у випадковому порядку, щоб вони служили в якості зразка.

Не можна виключити помилкові негативи повністю, але мможнаи можемо зменшити їх кількість, якщо обсяг оперативної пам'яті достатньо великий. Якщо припустили, що *s* є порогом підтримки, а також зразком усього набору даних є *p*, то ми використовуємо $p \times s$ в якості порогового значення для підтримки зразка. Проте, можна використовувати щось менше, наприклад $0.9 \times p \times s$. Маючи більш низький поріг, більше наборів кожного розміру мають бути підраховані, тому вимога у розмірі основної пам'яті зростає. З іншого боку, якщо є достатньо оперативної пам'яті, то потрібно визначити, яка має бути підтримка $0.9 \times p \times s$ в зразку майже всіх цих наборів елементів, які мають підтримку принаймні *s*. Якщо потім зробити повний прохід, щоб усунути ці набори елементів, які були ідентифіковані як часті в зразку, але вони не є частими в цілому, то не буде помилкових спрацьовувань і, швидше за все, не буде жодного або буде дуже мало помилкових визначень частого предметного набору як нечастого [Leskovec J, 2014, p.229].

На наступному кроці удосконалення алгоритму можна уникнути як хибно-негативних, так і хибнопозитивних результатів за рахунок створення двох повних проходів. Цей алгоритм називається SON. Ідея полягає в тому, щоб розділити вхідний файл на шматки (які можуть бути «шматками» в сенсі

розподіленої файлової системи, або просто шматком файлу). Розглядати кожен шматок в якості зразка, і запустити алгоритм, запропонований раніше для цього шматка. Далі використовувати $p \times s$ в якості порогового значення, якщо кожен фрагмент є частиною p всього файлу, і s є пороговою підтримкою. Далі необхідно зберігати на диску всі часті набори, знайдені для кожної порції.

Після того, як всі шматки будуть оброблені таким чином, необхідно почати об'єднання всіх наборів, які були виявлені частими для одного або декількох шматків. Ці набори будуть наборами-кандидатами. Варто звернути увагу на те, що якщо набору немає в жодній із части, то його підтримка менше $p \times s$ в кожній порції. Так як число фрагментів дорівнює $1 / p$, приходимо до висновку, що загальна підтримка менша, ніж $(1/ p) \times p \times s = s$. Отже, кожен частий предметний набір є частим прийнамні у одній частині загального набору, і можна бути впевненим, що всі дійсно часті предметні набори знаходяться саме серед наборів-кандидатів.

До цього етапу був зроблений один прохід через дані, за який були оброблені всі частни загального набору. У другому проході необхідно пройти через всі набори-кандидати і вибрати ті з них, які мають підтримку не меншу за s. Отримані предметні набори і будуть частими.

2.2.2 Розподілена реалізація алгоритму SON з використанням технології MapReduce

Алгоритм SON добре піддається до розпаралелювання у відповідному обчислювальному середовищу. Кожен з фрагментів може бути оброблений паралельно, і часто зустрічаються набори з кожного блоку, які можуть бути об'єднані, щоб сформувати кандидата. Можна розподілити кандидатів на багато процесорів, які підрахують підтримку кожного кандидата в підмножині кошиків, і, нарешті, підсумують отримані дані, щоб знайти підтримку для кожного кандидата у цілому наборі даних. Цей процес не повинен бути реалізований в MapReduce, але існує природний спосіб вираження кожного з двох проходів в якості операції MapReduce. На рис. 2.7 зображена загальна

модель вирішення цієї задачі за допомогою двох проходів MapReduce, а нижче в деталях описані кожен із двох етапів цього процесу.

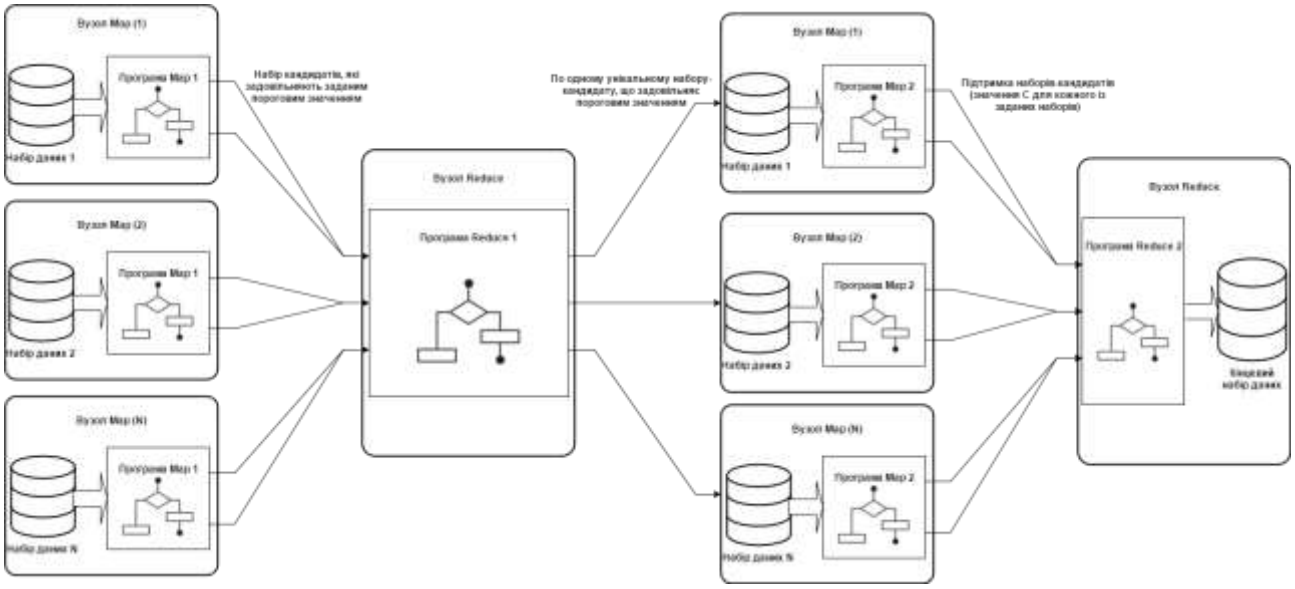

Рис. 2.7 – Модель вирішення задачі пошуку частих предметних наборів за допомогою MapReduce

Перша функція Map. Беремо призначений підмножина кошиків і знаходмо набори часто зустрічаємих елементів з використанням алгоритму, описаного раніше. Як було зазначено, нижній поріг підтримки від s до $p \times s$, якщо кожна задача Map отримує частинку p від загального розміру вхідних даних. Виходом цієї функції є набір пар ключ-значення *(F, 1)*, де F є частим предметним набором, а значення 1 не є важливим.

Перша функція Reduce. Кожному Reduce-завданню призначаються ключі з попереднього етапу, які в той же час є частими наборами. Значення ключа ігнорується, і ця програма просто передає далі набори, які зустрічаються не менше одного разу. Таким чином, виходом функції Reduce є набори-кандидати.

Друга функція Map. Завдання приймає всі вихідні дані з першої Reduce функції (набори-кандидати) і частини вхідного файлу даних. Кожне завдання Карти підраховує кількість входжень кожного з кандидатів в частині набору даних, яка була призначена на даному вузлі. Виходом є набір пар ключ-значення *(C, V)*, де *C* є одним з наборів-кандидатів і *V* є підтримка цього набору серед кошиків, які були подані на вхід цього Map-завдання.

Друга функція Reduce. На даному етапі набори елементів з завдання Map надаються в якості ключів і необхідно підсумувати відповідні для кожного ключа значення. В результаті отримуємо загальну підтримку для кожного з наборів-кандидатів, що були оброблені. Ті набори, підтримка яких не менша за *s* і подаються на вихід цієї функції. Набори елементів, які не мають достатню підтримку не передаються на вихід завдання Reduce.

Після виконання вищезазначених кроків алгоритм завершує свою роботу і результатом її є набір предметних наборів, які зустрічаються разом найчастіше серед заданих кошиків.

2.3 Висновок

У даному розділі були розглянуті дві задачі – кластеризації та пошуку частих предметних наборів. Кластеризація, як перший етап аналізу даних, здійснюється на підставі певних методів, які мають відображення в алгоритмах розбиття груп об'єктів. Виділення груп дозволяє спростити роботу з даними, після кластеризації застосовуються інші методи, для кожної групи будується окрема модель. Інша задача полягає у виявленні максимальних послідовностей серед всіх послідовностей, що мають підтримку вище заданого порогу. Кожна така максимальна послідовність і є послідовним шаблон. Послідовності, що задовольняють обмеження мінімальної підтримки є частими послідовностями.

Для кожної із задач у випадку великого набору вхідних даних існує проблема, що не всі дані можливо одночасно розмістити в оперативній пам'яті. Тому розробка алгоритмів, що дозволяють вирішувати ці задачі, завантажуючи в один момент часу лише частину даних, є важливим напрямком досліджень.

У даному розділі були запропоновані дві моделі обробки великих масивів даних. В основі першої моделі (для задачі кластеризації) лежить алгоритм CURE, який використовує репрезентативну вибірку кластеру і не зберігає усі його точки. В основі другої моделі (для задачі пошуку частих предметних

наборів) лежить алгоритм SON, який за два проходи по даним знаходить часті набори із заданою пороговою підтримкою.

В наступному розділі будуть описані результати апробації вищевказаних алгоритмів у розподіленій системі з використанням Hadoop MapReduce.

3 АПРОБАЦІЯ МОДЕЛЕЙ ОБРОБКИ ВЕЛИКИХ МАСИВІВ ДАНИХ З ВИКОРИСТАННЯМ HADOOP MAPREDUCE

3.1 Налаштування Hadoop MapReduce

Налаштування інфраструктури для проведення тестування відбувалось на початковому етапі для кластеру з одним вузлом, а в подальшому – для кластеру з трьома вузлами. Деталі налаштування кластерів будуть описані далі.

3.1.1 Кластер з одним вузлом

У цьому розділі описується, як встановити та налаштувати Hadoop для кластеру, який складається з одного вузла, і приклад швидкого виконання простих операцій з використанням Hadoop MapReduce і розподіленої файлової системи Hadoop (HDFS). GNU / Linux підтримується як розвиток і виробничої платформи. Hadoop була продемонстрована на кластерах GNU / Linux з 2000 вузлами. Win32 підтримується в якості платформи розробки. Розподілена операція не була добре перевірена на Win32, так що вона не підтримується в якості виробничої платформи. Тому в даному дослідженні використовуємо дистрибутив Linux – Ubuntu 14.04 [http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html].

Необхідне програмне забезпечення для Linux і Windows, включає в себе: JavaTM 1.6.x, переважно від Sun, а також – SSH повинен бути встановлений і SSHD повинен бути запущений, щоб використовувати скрипти Hadoop, які керують віддаленими демонами Hadoop. Після перевірки наявності необхідних програм – завантажуємо дистрибутив Hadoop.

Розпаковуємо завантажений дистрибутив Hadoop. У дистрибутиві, відредагуємо файл Conf / hadoop-env.sh, щоб визначити, принаймні що JAVA_HOME вказує на встановлену Java.

За замовчуванням, Hadoop налаштований для роботи в режимі нерозподіленого, як єдиний процес Java. Це корисно для налагодження. У

наступному прикладі каталог conf необхідно використовувати в якості вхідних даних, а потім знаходити і відображає кожне співпадіння даного регулярного виразу. Висновок (результат) записується до заданої вихідної директорії.

```
$ mkdir input
$ cp conf/*.xml input
$ bin/hadoop jar hadoop-examples-*.jar grep input output 'dfs[a-z.]+'
$ cat output/*
```

Рис. 3.1 – Базові налаштування та перший запуск Hadoop

Hadoop, також може бути запущений на одному вузлі – в режимі псевдо-розподілених обчислень де кожен Hadoop демон запускається в окремому процесі Java. Для цього необхідно встановити наступні налаштування [https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html].

conf/core-site.xml:

```
<configuration>
     <property>
         <name>fs.default.name</name>
         <value>hdfs://localhost:9000</value>
     </property>
</configuration>
```

conf/hdfs-site.xml:

```
<configuration>
     <property>
         <name>dfs.replication</name>
         <value>1</value>
     </property>
</configuration>
```

conf/mapred-site.xml:

```
<configuration>
     <property>
         <name>mapred.job.tracker</name>
         <value>localhost:9001</value>
     </property>
</configuration>
```

Рис. 3.2 – Конфігурації для псевдо-розподілених обчислень

Далі необхідно відформатувати отриману розподілену файлову систему та запустити демони Hadoop. Для цього використовуємо описані нижче

команди. Вихід журналу Hadoop демона записується в каталог $ {HADOOP_LOG_DIR} (за замовчуванням $ {HADOOP_HOME} / logs).

Копіюємо вхідні дані у розподілену файлову систему, та запускаємо деякі приклади. Після запуску необхідно скопіювати отримані результати на локальну файлову систему чи переглянути їх прямо на розподіленій. Для цього використовуються команди, що наведені на рисунку 3.4. Результат запуску наведений на рисунку 3.5 [https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html].

```
$ bin/hadoop namenode -format
$ bin/start-all.sh
```

Рис. 3.3 – Форматування HDFS та запуск демонів Hadoop

Перегляд веб-інтерфейс для NameNode і JobTracker; за замовчуванням вони доступні за адресами:

- NameNode - http://localhost:50070/
- JobTracker - http://localhost:50030/

```
$ bin/hadoop fs -put conf input
$ bin/hadoop jar hadoop-examples-*.jar grep input output 'dfs[a-z.]+'
$ bin/hadoop fs -get output output
$ cat output/*
$ bin/hadoop fs -cat output/*
$ bin/stop-all.sh
```

Рис. 3.4 – Команди для запуску тестових прикладів

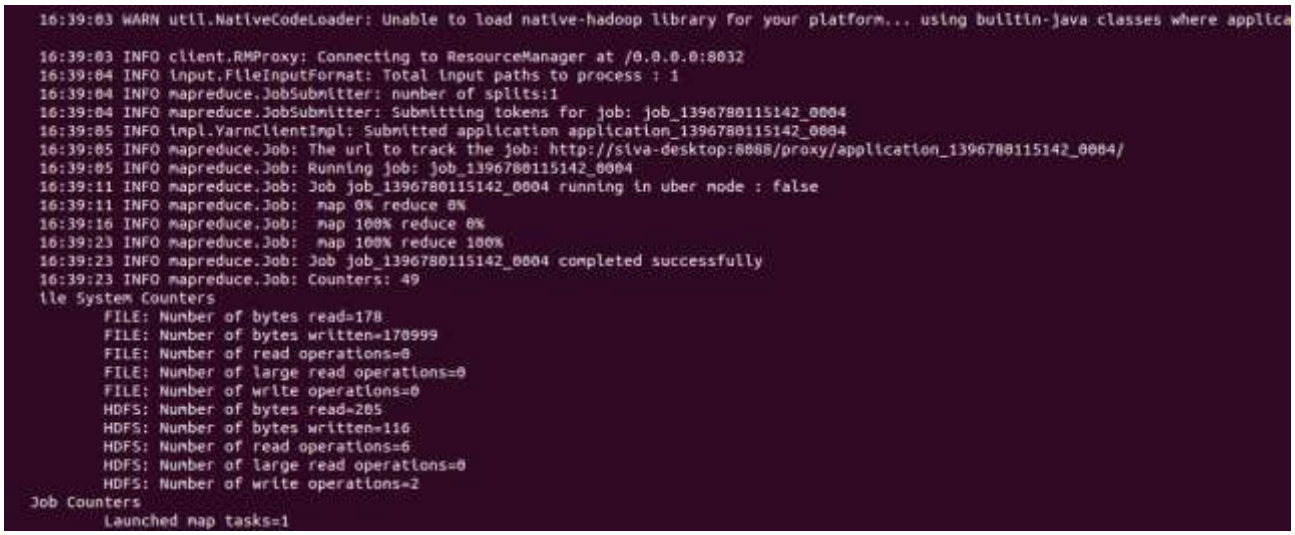

Рис. 3.5 – Результат запуску тестового прикладу

3.1.2 Кластер з декількома вузлами

Налаштування кластера Hadoop, як правило, включає в себе розпакування програмного забезпечення на всіх машинах в кластері. Як правило, один комп'ютер в кластері позначається як NameNode і іншу машину в якості JobTracker, виключно. Це майстер. Решта машин в кластері діють як обидва DataNode і TaskTracker. Це slave. Корінь розподілу називається HADOOP_HOME. Всі машини в кластері зазвичай мають один і той же шлях HADOOP_HOME [http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/ClusterSetup.html].

Конфігурація Hadoop управляється двома типами важливих файлів конфігурації:

- Тільки для читання конфігурації за замовчуванням – src/core/core-default.xml, src/hdfs/hdfs-default.xml and src/mapred/mapred-default.xml
- Сайт-специфічна конфігурація – conf/core-site.xml, conf/hdfs-site.xml and conf/mapred-site.xml.

Крім того, ви можете контролювати скрипти Hadoop, знайдені в bin / directory розподілу, шляхом установки значень по конкретних ділянках через conf/hadoop-env.sh.

Щоб налаштувати кластер Hadoop потрібно буде налаштувати оточення, в якому демони Hadoop виконуються, а також – параметри конфігурації для демонів. Демони Hadoop: NameNode / DataNode і JobTracker / TaskTracker.

Потрібно використовувати conf / hadoop-env.sh скрипт, щоб зробити сайт-специфічної настройки середовища процесу для демонів Hadoop. Потрібно вказати JAVA_HOME так, щоб він був коректно визначений на кожному віддаленому вузлі.

У більшості випадків потрібно також вказати HADOOP_PID_DIR каталог, який може бути записаний тільки користувачами, які збираються запустити Hadoop демон. В іншому випадку є потенціал для SYMLINK атаки.

Можна налаштовувати окремі демони за допомогою опцій конфігурації Hadoop _ * _ КЛЮЧ. Різні варіанти наведені в таблиці нижче.

Таблиця 4.1 – Опції для налаштування демонів

Демони	**Опції**
NameNode	HADOOP_NAMENODE_OPTS
DataNode	HADOOP_DATANODE_OPTS
SecondaryNamenode	HADOOP_SECONDARYNAMENODE_OPTS
JobTracker	HADOOP_JOBTRACKER_OPTS
TaskTracker	HADOOP_TASKTRACKER_OPTS

Нижче в таблицях наведена необхідна інформація для налаштування демонів Hadoop [http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/ClusterSetup.html].

Таблиця 4.2 – Налаштування файлу conf/core-site.xml

Параметр	**Значення**	**Примітка**
fs.default.name	URI of NameNode.	*hdfs://hostname/*

Таблиця 4.3 – Налаштування файлу conf/hdfs-site.xml

Параметр	**Значення**	**Примітка**
dfs.name.dir	Шлях в локальній файловій системі, де NameNode зберігає простір імен і транзакції журнали постійно.	Якщо це розділений комами список каталогів, то таблиця імені реплікується у всіх каталогах.
dfs.data.dir	Розділених комами список шляхів на локальної файлової системи, в DataNode, де він повинен зберігати свої блоки.	Якщо це розділений комами список каталогів, то дані будуть зберігатися у всіх каталогах, перелічених, як правило, на різних пристроях.

Таблиця 4.4 – Налаштування файлу mapred-site.xml

Параметр	**Значення**	**Примітка**
mapred.job.tracker	Хост або IP і порт JobTracker.	Пара *host:port*
mapred.system.dir	Шлях на HDFS, де, коли система рамкові зберігає MapReduce файли, наприклад, / hadoop / mapred / system /.	Це у файлової системі за замовчуванням (HDFS) і повинен бути доступний з сервера і клієнтів.
mapred.local.dir	Розділених комами список шляхів на локальній файловій системі, де написано дані тимчасового MapReduce.	Кілька шляхів поширення допомоги дискового введення / виводу.
mapred.tasktracker.{map|reduce}.tasks.maximum	Максимальна кількість MapReduce завдань, які виконуються одночасно на даній TaskTracker, індивідуально.	За замовчуванням 2 (2 map і 2 reduce).
dfs.hosts/dfs.hosts.exclude	Список дозволених / виключених DataNodes.	Контроль дозволеності.
mapred.hosts/mapred.hosts.exclude	Список дозволених / виключених TaskTrackers.	Контроль дозвленості.
mapred.queue.names	Розділених комами список черг, в яких робочі місця можуть бути представлені.	Система MapReduce завжди підтримує принаймні одну чергу з ім'ям за умовчанням.
mapred.acls.enabled	Булеве значення, вказавши для перевірки списків ACL черг і списків управління доступом на роботу, повинні бути зроблено для авторизації користувачів для виконання операцій і операцій черги завдань.	Якщо це TRUE, списки контроль доступ черзі перевіряються при подачі і адміністрування робочих місць.

Таблиця 4.5 – Налаштування файлу conf/hdfs-site.xml

Параметр	Значення	Примітка
mapred.queue.*queue-name*.acl-submit-job	Список користувачів і груп, які можуть відправляти завдання на ім'я-черзі, зазначеної.	Список користувачів і груп, як розділений комами список імен. Два списку розділені пропуском. Приклад: user1, user2 группа1, group2. Якщо ви хочете визначити тільки список груп, забезпечити заготівлю на початку значення.
mapred.queue.*queue-name*.acl-administer-jobs	Список користувачів і груп, які можуть переглядати відомості про завдання, змінити пріоритет або вбивають робочі місця, які були представлені на ім'я-черзі, зазначеної.	Список користувачів і груп, як розділений комами список імен. Два списку розділені пропуском. Приклад: user1, user2 группа1, group2. Якщо ви хочете визначити тільки список груп, забезпечити заготівлю на початку значення.

Після цього демони є налаштовані і необхідно робити налаштування інших компонентів Hadoop. Контролери завдань в рамках Hadoop MapReduce, які визначаються, як map і reduce завдання запускаються і контролюються. Вони можуть бути використані в кластерах, які вимагають деякі настройки в процесі запуску або управління користувальницьких завдань. Наприклад, в деяких групах, може бути вимога для виконання завдань, як користувача, що

відправив завдання, замість того, як користувач завдання трекера, який, як завдання запускаються за замовчуванням.

Існує два види контролеру задач. Контролер за замовчуванням завдання, яка Hadoop використовує для управління виконанням завдання. Ці завдання виконуються як користувачі завдання трекера [http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/ClusterSetup.html].

У той же час другий контролер завдання, яка підтримується тільки на Linux, виконує завдання, як користувача, що відправив завдання. Вона вимагає, щоб ці облікові записи користувачів, які будуть створені на вузлах кластера, де запускаються завдання. Він використовує Setuid виконуваний файл, який входить в дистрибутив Hadoop. Трекер завдання використовує цей виконуваний файл для запуску і вбити завдання. Для забезпечення максимальної безпеки, цей контролер завдань встановлює обмежені права доступу і власник / групи локальних файлів і каталоги, що використовуються завдання, такі як робота файли .jar, проміжні файли, файли журнал завдання і розподілені файли кеш. Особливо варто відзначити, що через це, крім власника завдання і TaskTracker, ніякий інший користувач не може отримати доступ до будь-якого з локальних файлів / каталогів, в тому числі локалізовані в рамках розподіленої кеш-пам'яті.

TaskTracker (TT) може бути налаштований для моніторингу використання пам'яті завдань Нереститься, так що погано поводилися робочі місця не збити машину з-за надмірного споживання пам'яті. При включенні моніторингу, кожна задача поставлена задача граничного терміну віртуальної пам'яті (Vmem). Крім того, кожен вузол призначається вузловий граничний термін для використання VMEM. TT гарантує, що завдання буде вбита, якщо він і його нащадки, використовувати VMEM над кожним завданням лімітом завдання. Це також гарантує, що один або кілька завдання вбиті, якщо загальна сума використання VMEM всі завдання, і їх нащадкам, перетнути вузол-ліміт.

NameNode і JobTracker отримує ідентифікатор стійки ведених в кластері за допомогою виклику API рішучість під обліковим записом адміністратора

налаштований модуль. API дозволяє ім'я DNS веденого пристрою (також IP-адреса) в стійку ід. Який модуль використовувати можна налаштувати за допомогою елемента конфігурації topology.node.switch.mapping.impl. Реалізація за замовчуванням такий же запускає скрипт / команду, сконфігурованих за допомогою topology.script.file.name. Якщо topology.script.file.name не встановлено, стелаж ідентифікатор / за замовчуванням стійка повертається для будь-якого пройденого IP-адреси. Додаткова конфігурація в Map / Reduce частини є mapred.cache.task.levels, який визначає кількість рівнів (в топології мережі) кешем [http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/ClusterSetup.html].

Перезапуск трекера завдання може відновити виконання завдань, якщо mapred.jobtracker.restart.recover встановлений вірно і JobHistory ведення журналу включено. Також mapred.jobtracker.job.history.block.size значення має бути встановлено до оптимального значення скинути історію завдань на диск якомога швидше, типове значення 3145728 (3MB).

Для запуску та завершення завдання необхідно запустити команди, які вказані на рисунку 3.6.

```
$ bin/hadoop namenode -format
$ bin/start-dfs.sh
$ bin/start-mapred.sh
$ bin/stop-dfs.sh
$ bin/stop-mapred.sh
```

Рис. 3.6 – Команди для запуску та завершення завдань MapReduce

3.2 Архітектура розроблених програм

Для опису програм буде використовуватись мова UML, за допомогою якої можна описати структурні елементи, а також принцип роботи програми.

3.2.1 Програма для вирішення задачі кластеризації

Для початку необхідно визначити, які саме можливості матиме користувач, який використовує розроблену програму. Для цього на рисунку 3.7 зображена діаграма прецендентів (або use-case – діаграма). Як можна побачити на діаграмі, у користувача (інженера) будуть наступні можливості: завантаження вхідних даних, встановлення обмежень для розрахунків (наприклад, мінімальна відстань між кластерами, тощо), запуск процесу кластеризації та завантаження отриманого результату.

Рис. 3.7 – Діаграма прецендентів для програми реалізації алгоритму CURE

Діаграма прецендентів певним чином зображує вимоги, які ставляться до розробника програмного забезпечення. Відповідно, після аналізу цієї діаграми варто виділити незалежні компоненти програми та описати їх та їхню взаємодію на діаграмі класів (рис. 3.8).

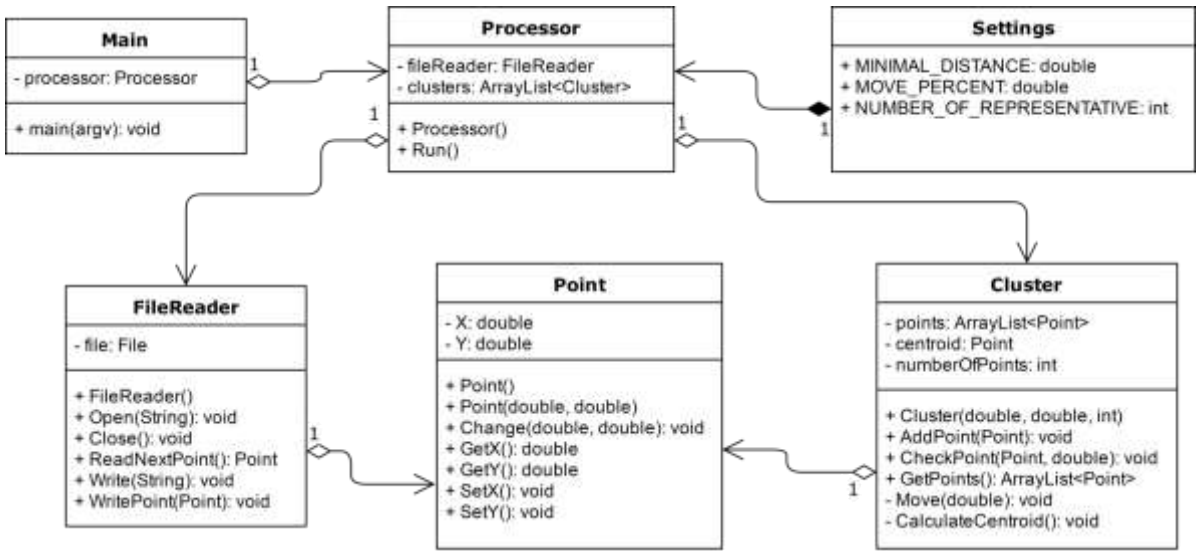

Рис. 3.8 – Діаграма класів для програми реалізації алгоритму CURE

Для даної програми виділено 6 незалежних компонентів, а саме – головний клас, що оброблює запуск програми. Клас для загальної роботи програми, Processor. Клас із статичними полями – налаштування, що містить у собі налаштування мінімальної дистанції, для якої об’єднувати класи, відсоток відстані, на яку здвигати репрезентативні точки, кількість репрезентативних точок. Для роботи з файлами виділений клас FileReader. І, відповідно, для абстракцій над точками та кластерами були створені класи Cluster та Point, які включають в себе всю необхідно для обробки інформацію.

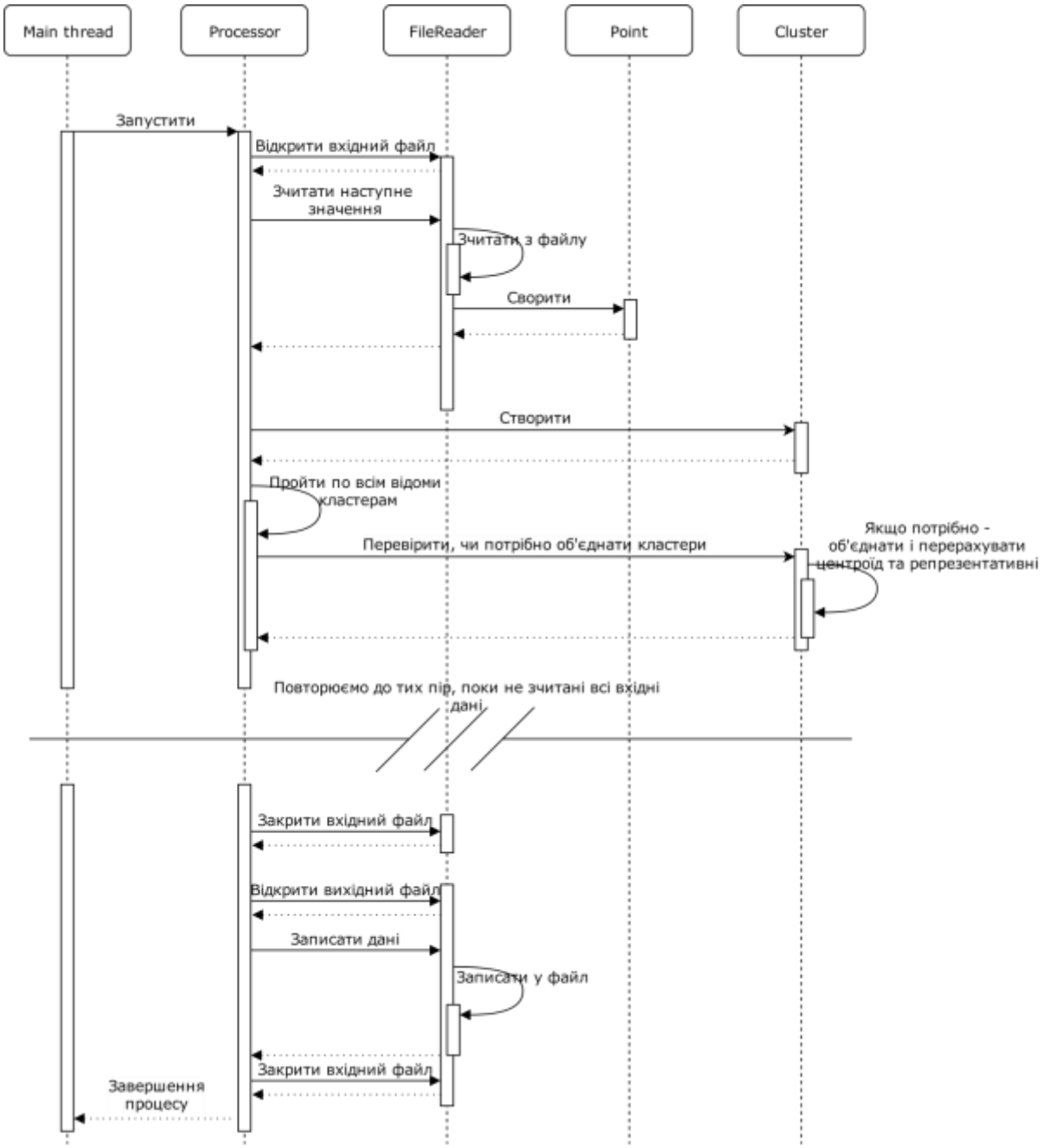

Рис. 3.9 – Діаграма послідовностей для програми реалізації алгоритму CURE

Більш детальна взаємодія класів зображена на діаграмі послідовностей, що на рисунку 3.9. Починається робота з головного потоку, який створює об’єкт класу Processor, в якому буде реалізована основна частина алгоритму. Як зазначалось раніше, для роботи алгоритму необхідні допоміжні класи – Point та Cluster.

Після того, як Processor почав роботу, в цьому класі сворюється об'єкт класу FileReader, який буде проводити зчитування даних з вхідних файлів. Ці файли містять у собі набір точок, і саме тому у класі FileReader є можливість зчитати наступні точки, готові об’єкти.

Як тільки клас Processor отримує нову точку, він створює для неї власний кластер (об'єкт класу Cluster). Далі цей новостворений кластер може об’єднатись з одним із інших кластерів, після чого необхідно буде перевірити, чи не потрібно об’єднати якісь з існуючих кластерів.

Цей процес необхідно повторювати до тих пір, поки не будуть зчитані усі точки з вхідного файлу. Після цього, клас Processor створює новий об’єкт класу FileReader вже для запису вихідних даних.

Варто також описати роботу програми, основуючись саме на технології MapReduce. На рисунку 3.10 зображена діаграма діяльностей, на якій можна побачити взаємодію саме частин Map та Reduce з точки зору будови програми.

Починається програма у головному потоці, який запускає процеси Map. Далі після ініціалізації цього процесу у кожному окремому процесу Map відкривається відповідний файл вхідних даних і починається локальна кластеризація на окремому вузлі. Як тільки процес Map на всіх вузлах завершив виконання, головний процес створює процеси Reduce. У цьому процесі проходить об'єднання кластерів, отриманих з різних обчислювальних вузлів. Які тільки усі дані оброблені – головний потік завершує роботу програми.

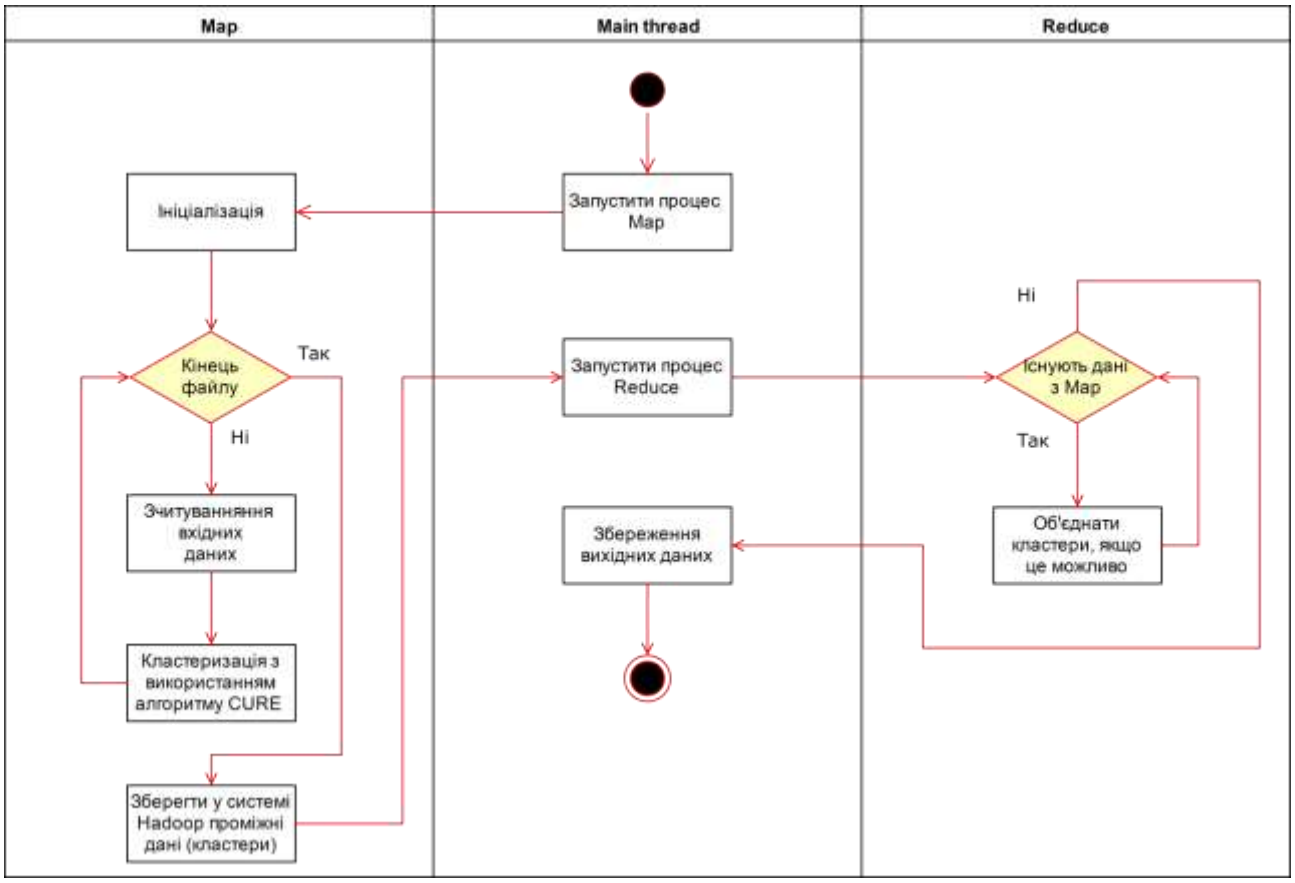

Рис. 3.10 – Діаграма діяльностей для програми реалізації алгоритму CURE

3.2.2 Програма для вирішення задачі пошуку частих предметних наборів

Діаграма прецендентів схожа на аналогічну діаграму з попереднього пункту, різниця лише у тому, що в даному випадку головним є процес пошуку частих предметних наборів. На рис. 3.11 зображена діаграма для даної задачі. Після аналізу вимог до програми було розроблено діаграму класів.

Діаграма класів зображена на рис. 3.12. Для даної програми відрізняються формати вхідних даних, тому клас FileReader має іншу реалізацію. Також, у класі Settings вказані константи – кількість частих наборів та кількість елементів у кожному з наборів.

Для полегшення обробки даних у класі Processor був створений клас Dataset, який є абстракцією над окремим частим предметним набором. Діаграма послідовностей зображена на рис. 3.13. Ця діаграма схожа до попередньої. Так само необхідно обробити всі вхідні дані з файлів, але в цьому випадку необхідно знайти часті предметні набори.

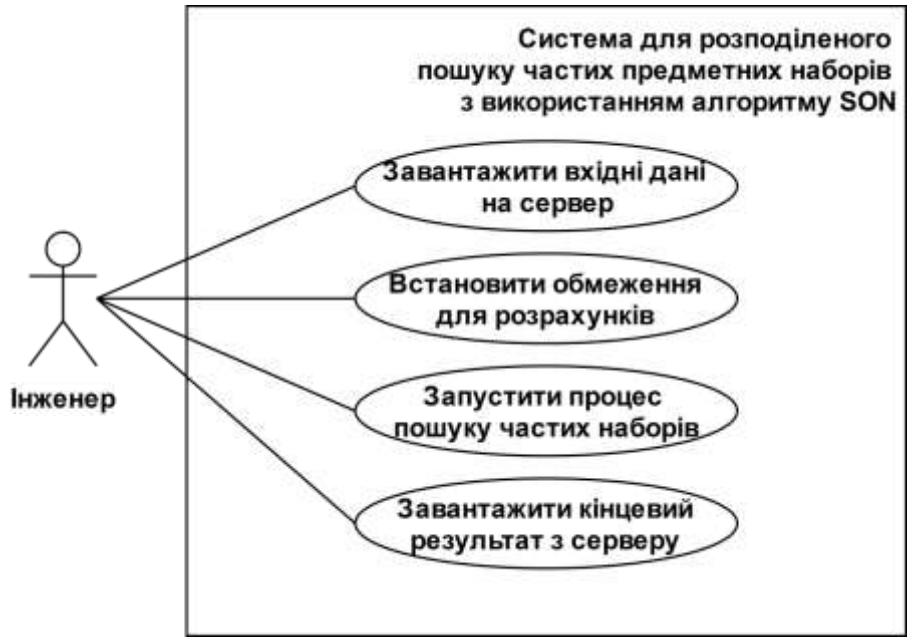

Рис. 3.11 – Діаграма прецендентів для програми реалізації алгоритму SON

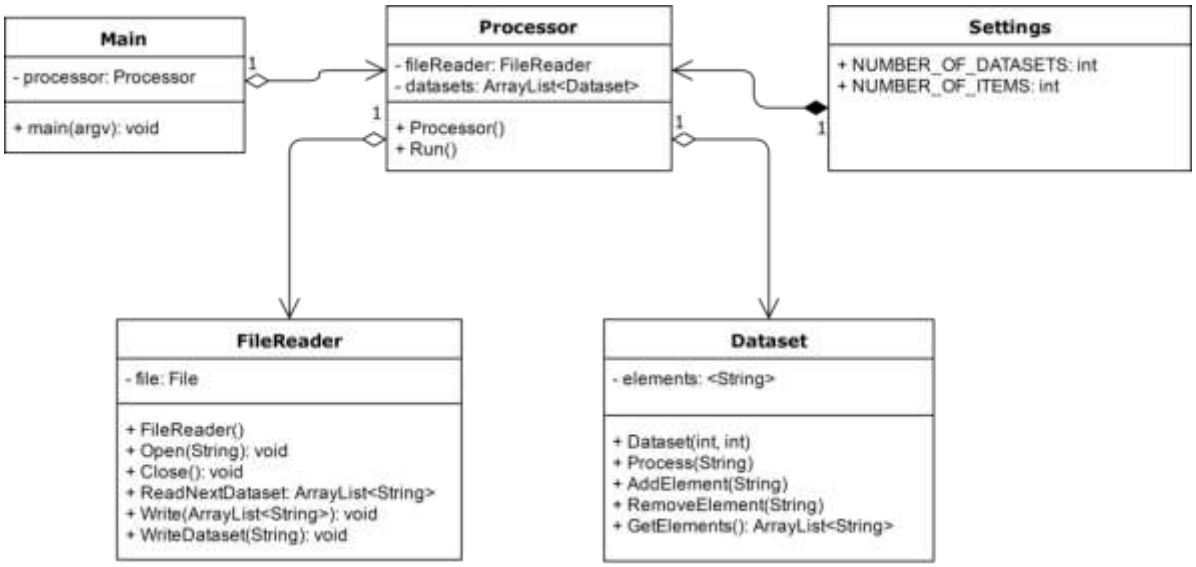

Рис. 3.12 – Діаграма класів для програми реалізації алгоритму SON

Діаграма діяльностей зображена на рис. 3.14. На ній описана робота кожної, з трьох сутностей даної програми. Також варто відзначити, що для рішення даної задачі необхідно робити 2 проходи по всім даним, на першому етапі – знайти часті набори-кандидати, а на другому етапі – порахувати кількість входжень кожного з наборів.

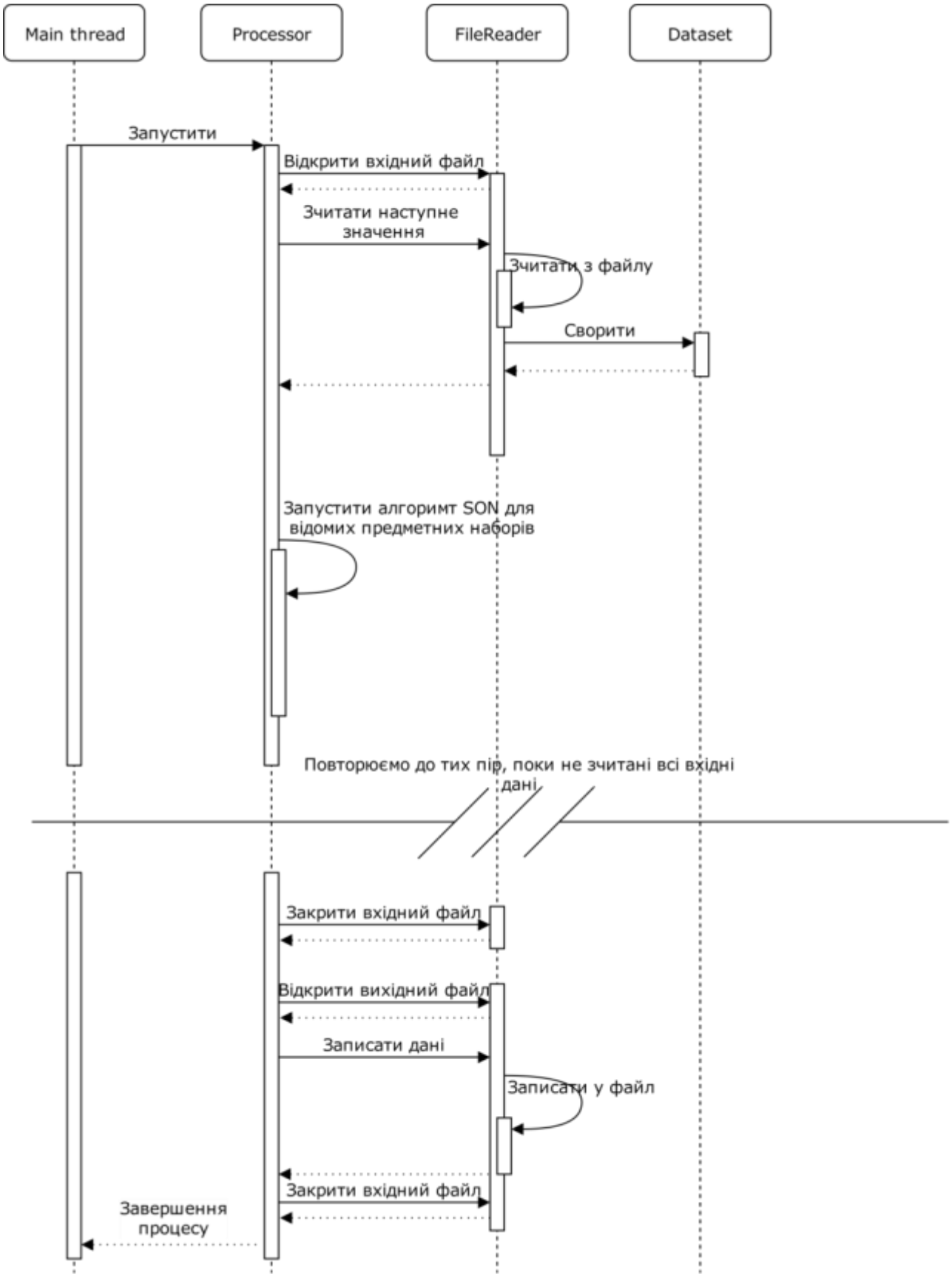

Рис. 3.13 – Діаграма послідовностей для програми реалізації алгоритму SON

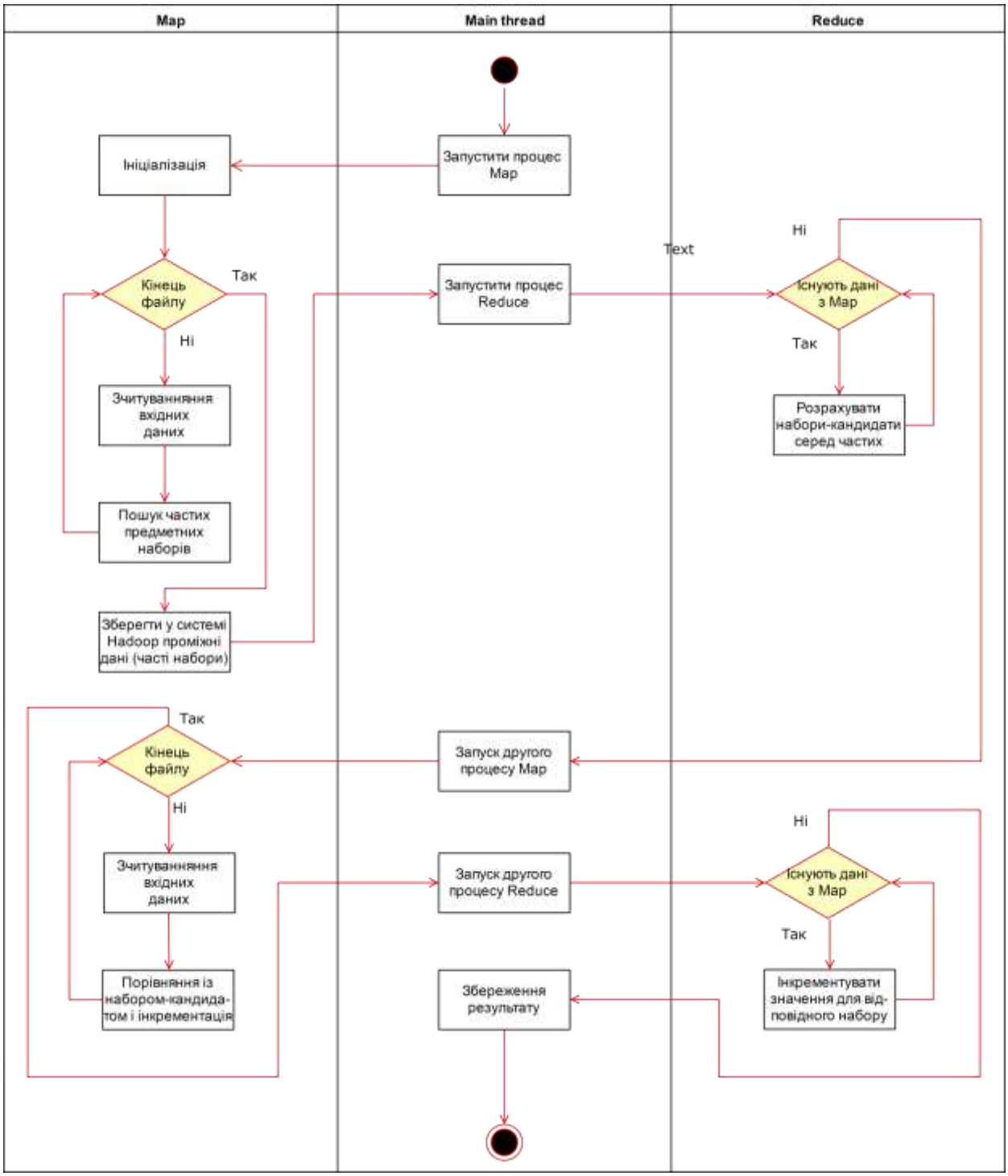

Рис. 3.14 – Діаграма діяльностей для програми реалізації алгоритму SON

3.3 Результати апробації запропонованих моделей

Для апробації моделей були частково були згенеровані тестові виборки даних. Також, деякі з даних були взяті з доступних онлайн-ресурсів, проте, варто відмітити, що існує мала кількість безкоштовних та доступних виборок в

інтернеті. Тому більша частина даних для апробації алгоритмів була саме згенерована власноруч.

3.3.1 Апробація моделі для вирішення задачі кластеризації

На початковому етапі дослідження використовувався локальний комп'ютер для тестування запропонованих підходів. Тестування відбувалось у наступній послідовності: спочатку була створена програма для обробки даних, що не містить у собі розподілені обчислення. А далі запускалась програма на основі MapReduce. Обидві були написані з використанням мови програмування Java. Як тестові виборки для апробації алгоритму CURE були згенеровані набори двовимірних координат, як зображено на рис. 3.15.

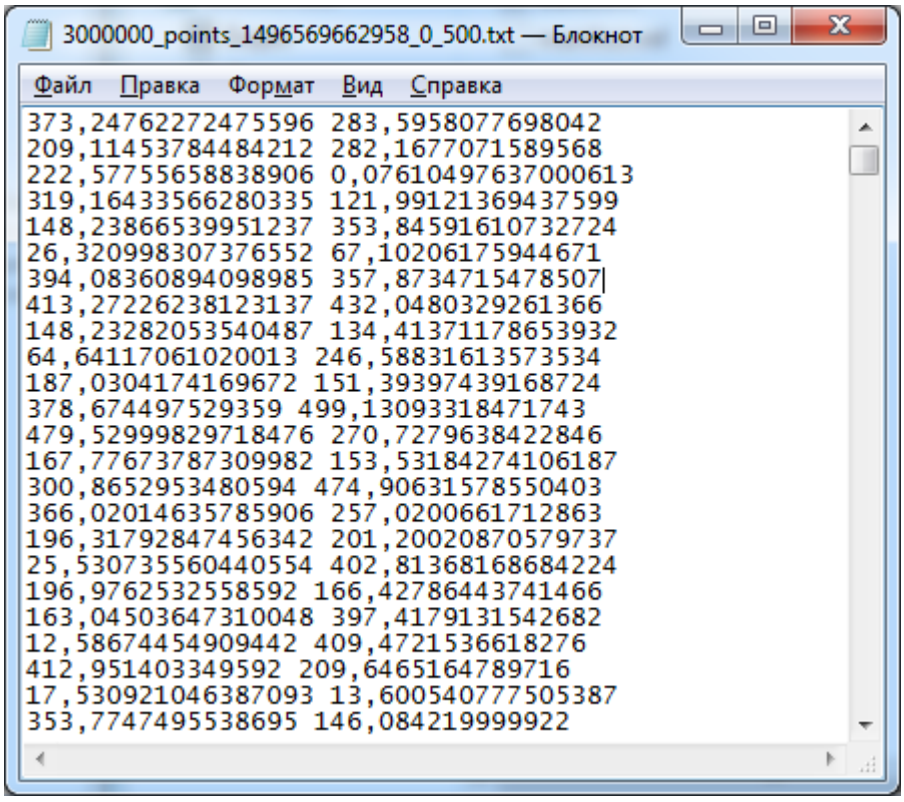

Рис. 3.15 – Приклад файлу вхідних даних для алгоритму CURE

Вихідні дані зображені на рис.3.16. У вихідних даних містяться репрезентативні точки кожного з отриманих кластерів. Аналогічно, першою йде координата по осі X, а другою йде координата по осі Y.

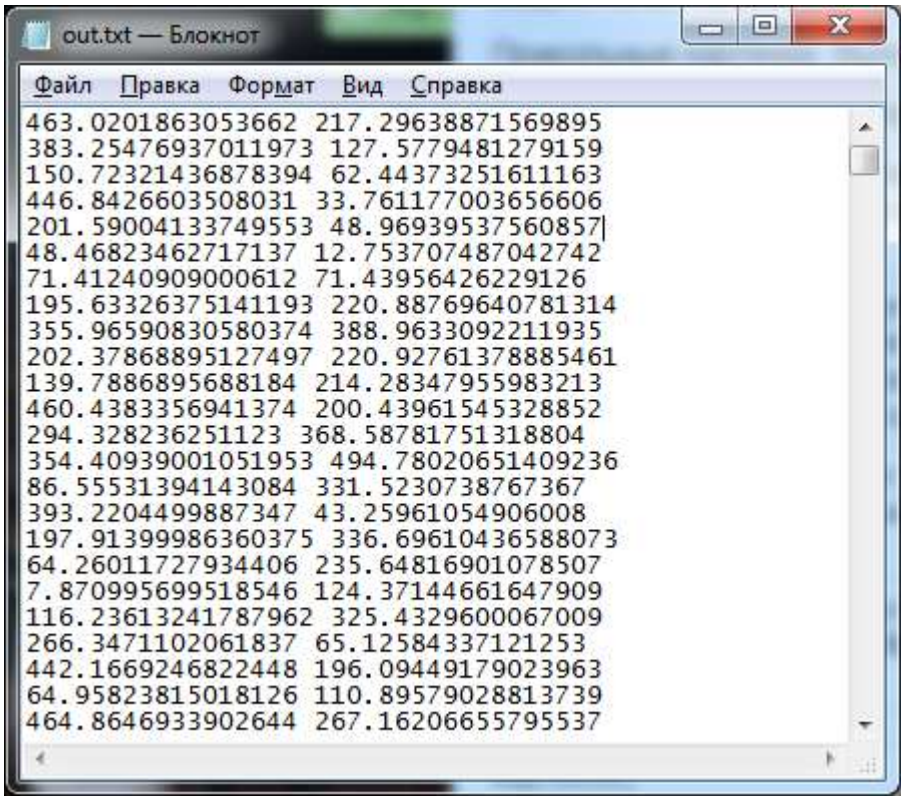

Рис. 3.16 – Приклад файлу вихідних даних з алгоритму CURE

Перед тим, як дослідити швидкість роботи розробленої моделі, були перевірені результати її роботи, що зображені на рисунках 3.17, 3.18 та 3.19. Для першого випадку було згенеровано 5 кластерів з невеликою кількістю точок у кожному. Значення для програми були обрані наступні: відстань рівна 0.5, кількість репрезентативних точок рівна 5.

У другому прикладі (рисунок 3.18) кількість точок та кластерів більше. Серед констант обрані наступні: кількість репрезентативних точок рівна 10, а мінімальна відстань для об’єднання кластерів рівна 5. На цьому рисунку є цікавий приклад – точка з координатами (44;72), яка одночасно і є власним кластером і єдиною репрезентативною точкою і центроїдом свого кластеру.

У третьому прикладі наглядно видно основну властивість алгоритму CURE – можливість кластери, якщо вони знаходяться один в іншому, тощо.

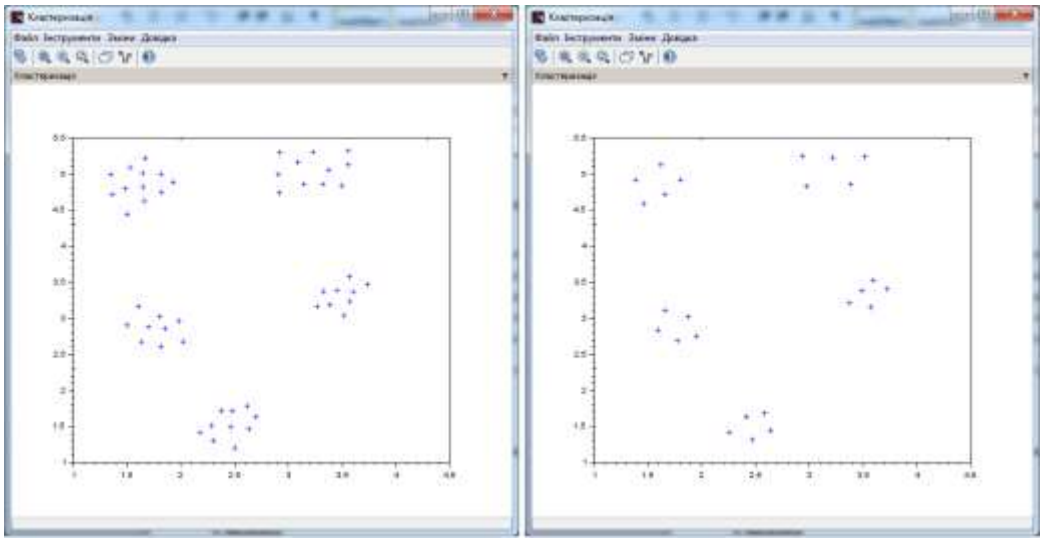

Рис. 3.17 – Приклад роботи алгоритму (до та після кластеризації), 5 кластерів

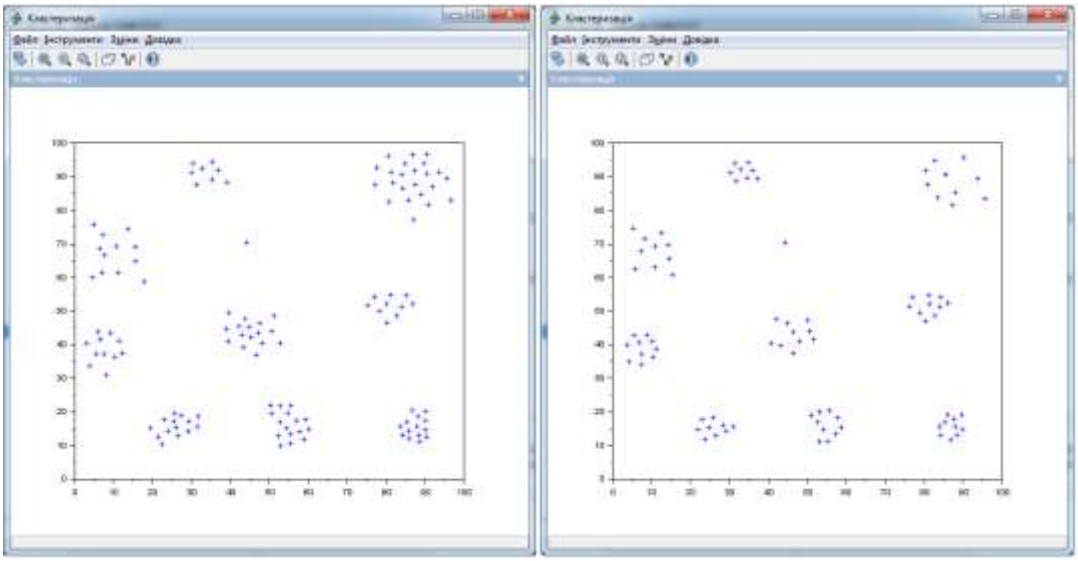

Рис. 3.18 – Приклад роботи алгоритму (до та після кластеризації), 10 кластерів

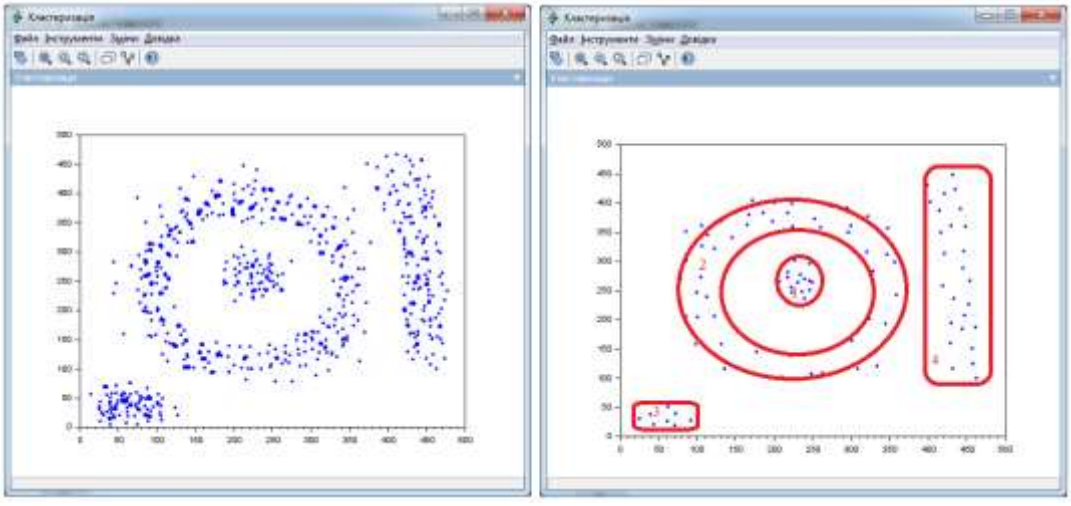

Рис. 3.19 – Приклад роботи алгоритму (до та після кластеризації), кластер у кластері та інші поряд

На першому етапі дослідження інфраструктура Hadoop MapReduce була налаштована на локальному комп'ютері з операційною системою Ubuntu. Для аналізу алгоритмів обробки великих даних даний підхід не є показовим.

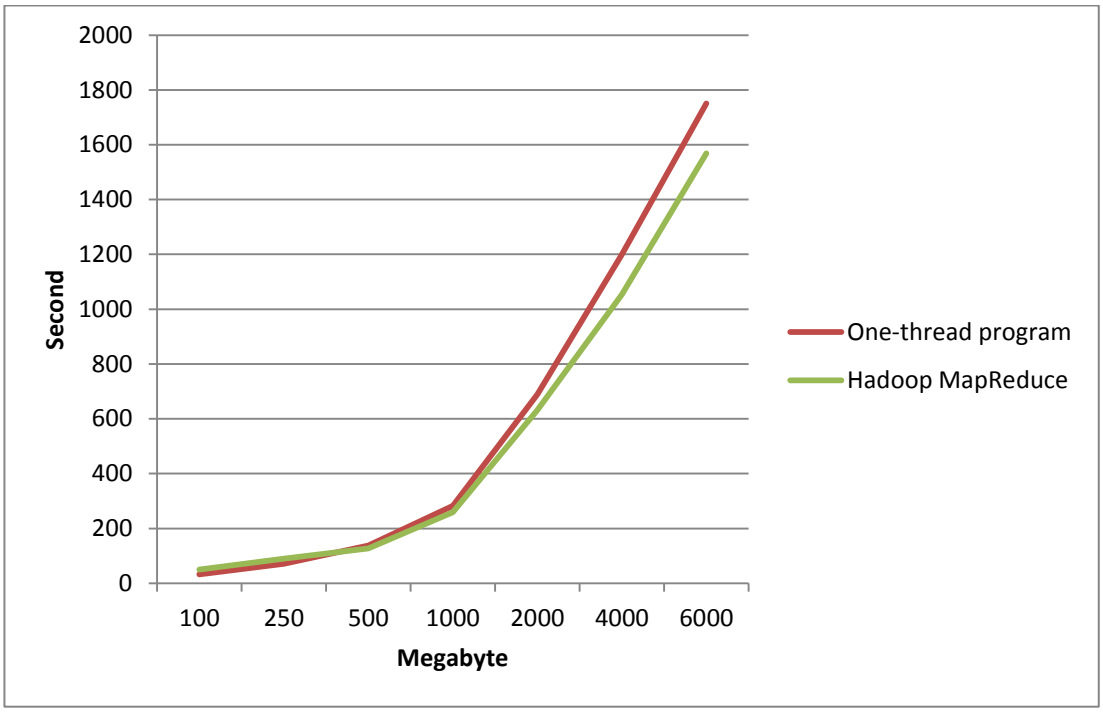

Рис. 3.20 – Час роботи прямого та розподіленого алгоритмів на локальному комп’ютері

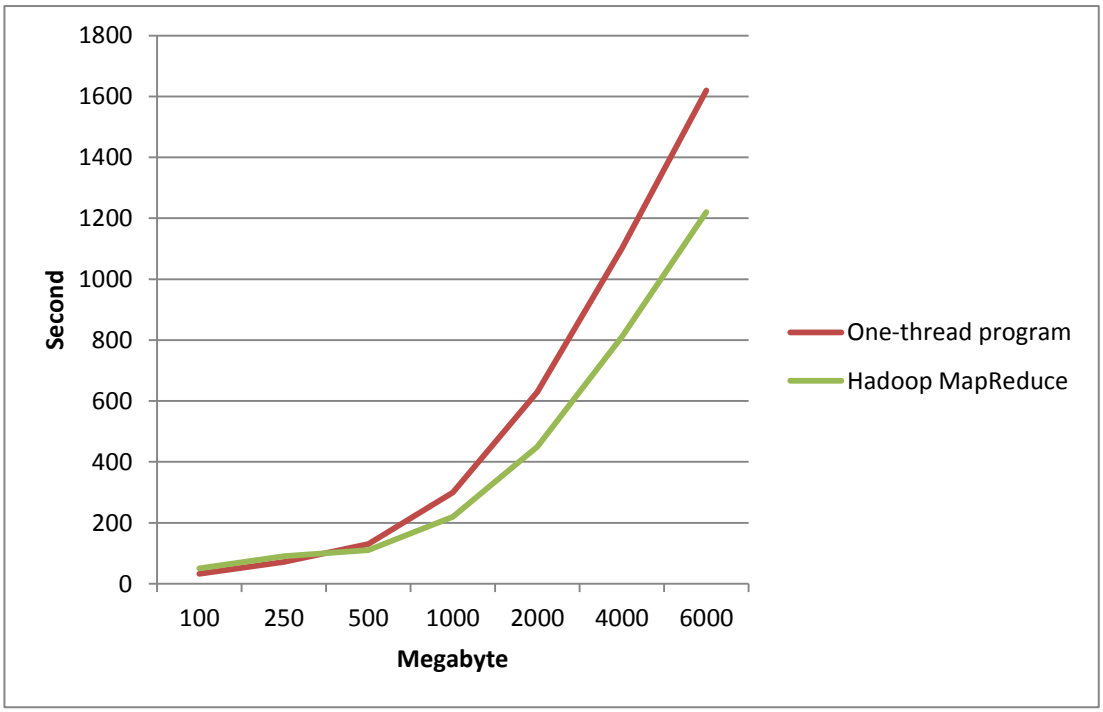

Рис. 3.21 – Час роботи прямого та розподіленого алгоритмів на локальному комп’ютері після закриття усіх фонових програм

Проте, навіть на рисунку 3.20 видно, що використовуючи Hadoop MapReduce, обробка даних відбувається швидше, ніж без його використання. Цікавий є і той факт, що пришвидшення не є великим [Yaremenko V., “Internauka”, 2017, pp. 81-83]. Тому у даному випадку було проведено додаткове дослідження – чому пришвидшення саме таке. І було виявлено, що через запущені додаткові програми, які працювали в один час із проведенням дослідження, були причиною затримок. Тому після закриття усіх фонових програм (особливо важливим було закриття Google Chrome) швидкість роботи збільшилась, як зображено на рис. 3.21.

Після запуску на локальній машині, дослідження було проведено на сервері кафедри Системного проектування з можливістю задіяти 3 обчислювальні вузли. Результати були наступними: для кластеризації даних, розмір яких менше за 250 МБ не варто використовувати Hadoop MapReduce, бо пришвидшення роботи немає. Якщо розмір даних більший за 250 МБ можна розподіляти роботу між обчислювальними вузлами. Важливим моментом є те, що пришвидшення є навіть у випадку використання одного вузла MapReduce.

Таблиця 4.6 – Результат запуску програми у розподіленій системі

Розмір даних	Програма з одним потоком	Hadoop MapReduce (1 вузол)	Hadoop MapReduce (2 вузли)	Hadoop MapReduce (3 вузли)
10	3	15	20	22
50	18	25	29	31
100	30	45	50	48
250	65	80	75	70
500	120	100	90	85
1000	250	200	120	100
2000	560	385	245	188
4000	950	720	510	402
6000	1430	1102	860	709

Зі зростом кількості вузлів, швидкість обробки даних зростає. Результати дослідження представлені у таблиці 4.6, а також графічно на рис. 3.22 та 3.23. Отримати швидкість роботи MapReduce меншу за 14 секунд під час експериментів не вийшло. Це пов'язано з тим, що для налаштування

інфраструктури, у тому числі – для запуску демонів, необхідно витратити певний час.

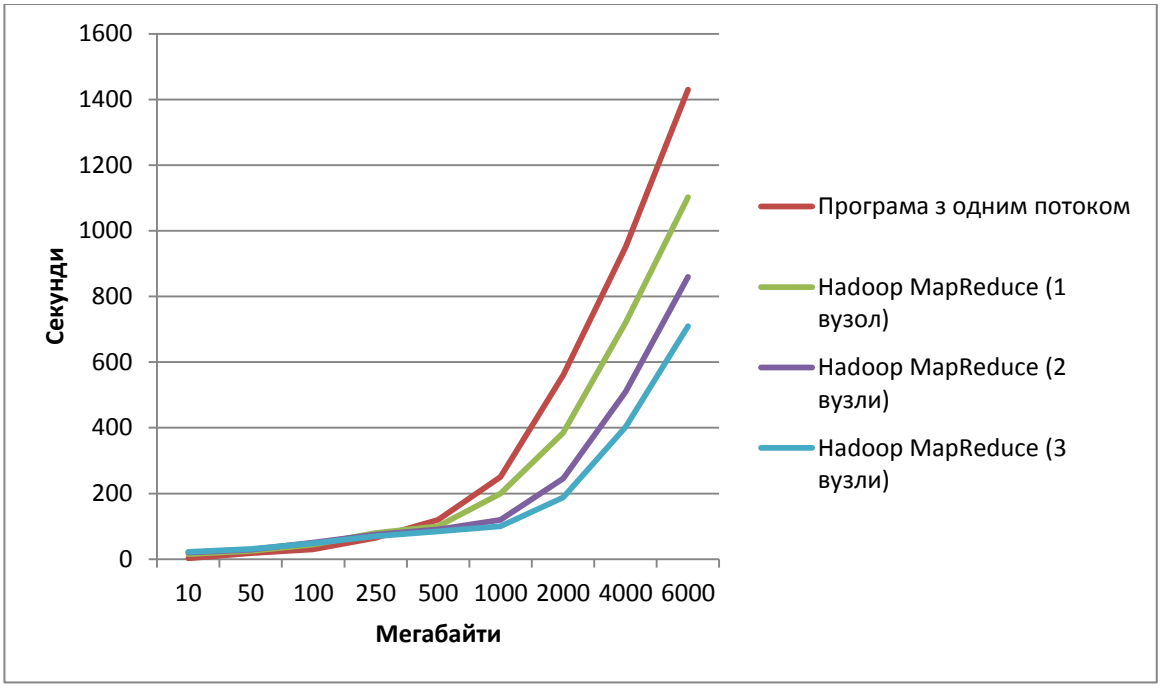

Рис. 3.22 – Час роботи прямого та розподіленого алгоритмів на сервері (приблизно 10000 кластерів) з різною кількістю вузлів (1, 2 та 3)

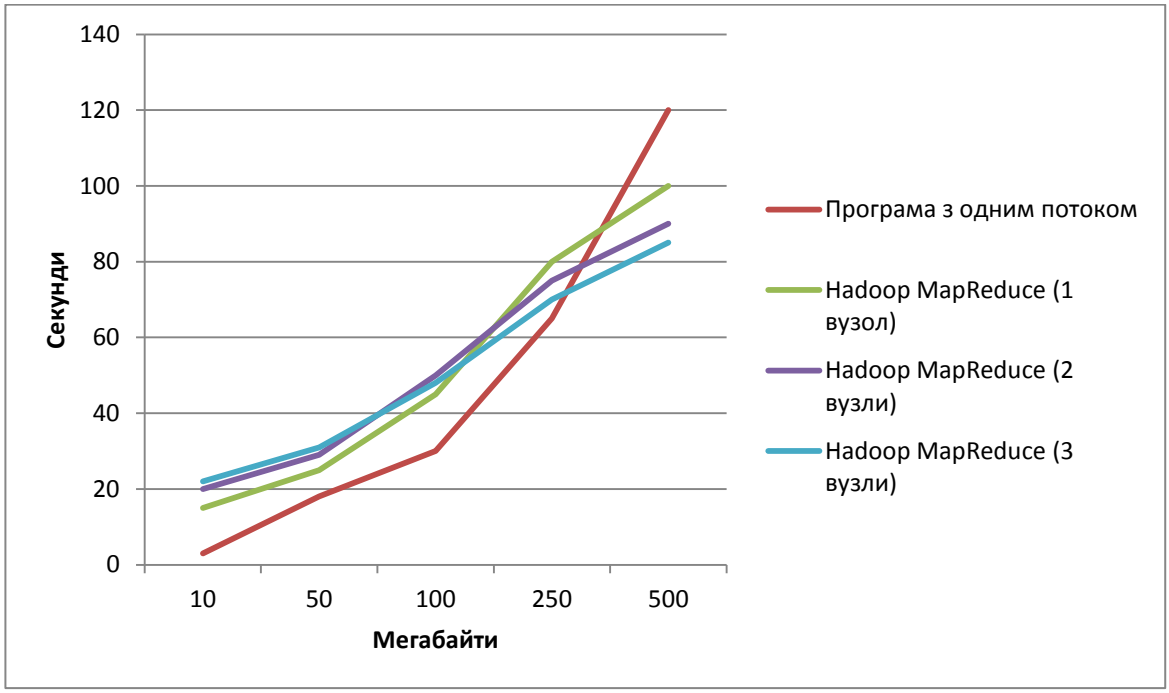

Рис. 3.23 – Час роботи прямого та розподіленого алгоритмів на сервері (приблизно 10000 кластерів) з різною кількістю вузлів (1, 2 та 3) для даних розміром до 500 МБ

3.3.2 Апробація моделі для вирішення задачі пошуку частих предметних наборів

В даному пункті описано результати апробації моделі обробки великих даних з використанням алгоритму SON. Для цього алгоритму користувачем задаються дві константи, а саме – кількість частих предметних наборів та кількість елементів у кожному з наборів.

На рисунку 3.24 зображено приклад файлу вхідних даних для цього алгоритму. Цей файл має містити у собі набір слів / символьних значень, де у кожному рядку описаний один можливий кошик.

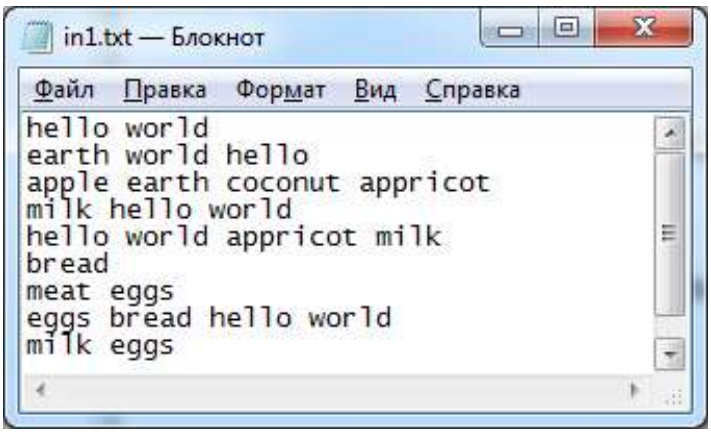

Рис. 3.24 – Вигляд вхідних даних для алгоритму SON

Приклад вихідних даних зображений на рис. 3.25. У кожному рядку вказаний окремий частий набір, який частіше інших зустрічається серед вхідних даних.

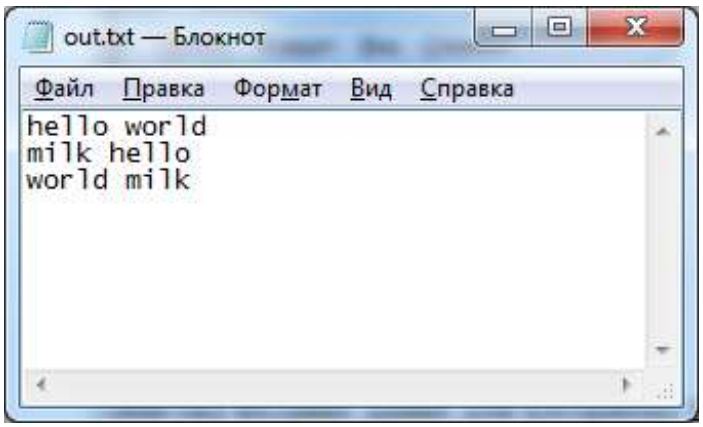

Рис. 3.25 – Вигляд вихідних даних алгоритму SON

На початку випробовування алгоритму були створені тестові набори даних, починаючи з 1 слова в наборі, далі 2 і продовжуючи збільшення

кількості слів. Це необхідно було зробити, щоб переконатись у правильності роботи розробленої програми. На рисунку 3.26. зображений один із вхідних файлів даних та вихідний файл.

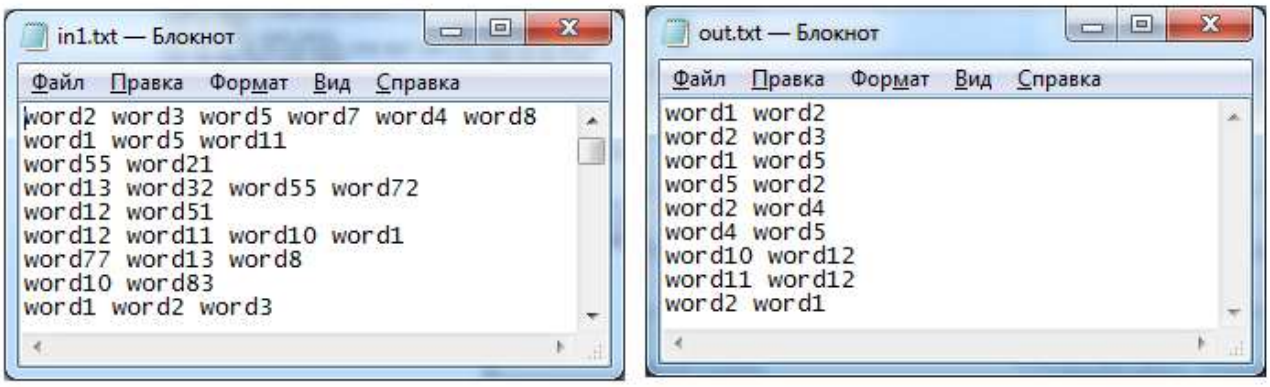

Рис. 3.26 – Приклад обробки згенерованих даних (10 найбільш частих наборів по 2 елементи в кожному)

В ході даного дослідження в інтернеті шукались набори даних, що містять приклади кошиків відвідувачів супермаркету. А саме – хто і що купив на один чек. На жаль, цієї інформації, навіть тестової, не було знайдено в інтернеті, але, на одному з сайтів була інформація про купівельні кошики. Інформація була у наступному вигляді – кожному товару було призначено у відповідність певне число, а набір чисел у кожному з рядків і відповідав одному кошику, що зібрав покупець. Приклад обробки даного тестового набору зображений на рис. 3.27.

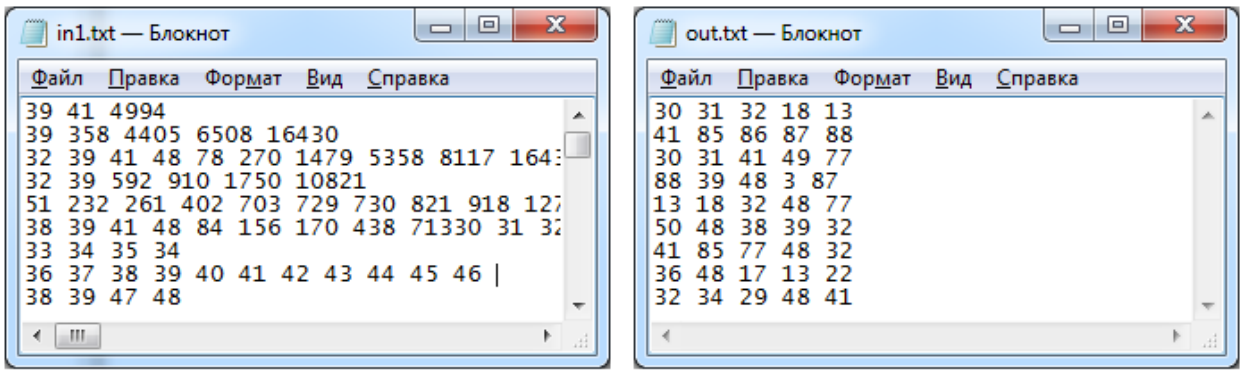

Рис. 3.27 – Приклад обробки даних з інтернету (10 найбільш частих наборів по 5 елементів в кожному)

Загальні графіки швидкості роботи програм, аналогічно до апробації алгоритму CURE, зображено на рисунках 3.28 та 3.29.

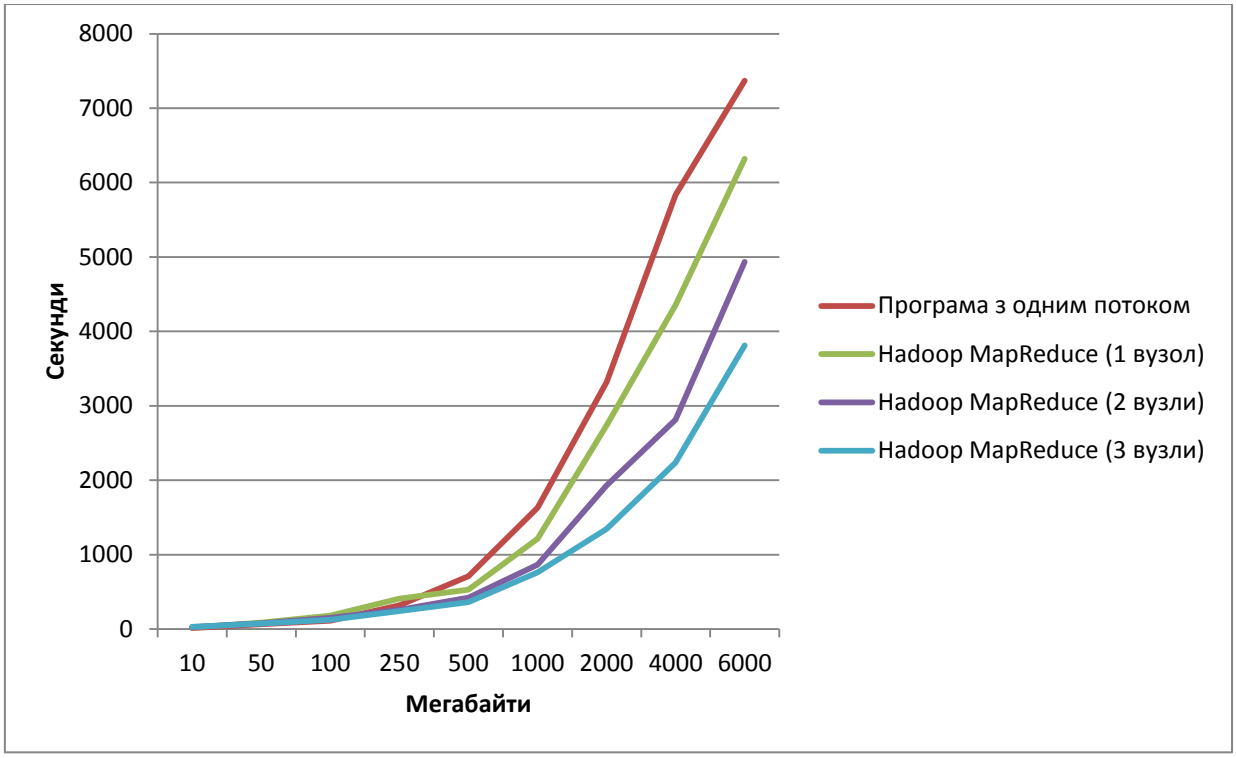

Рис. 3.28 – Час роботи прямого та розподіленого алгоритмів на сервері з різною кількістю вузлів (1, 2 та 3)

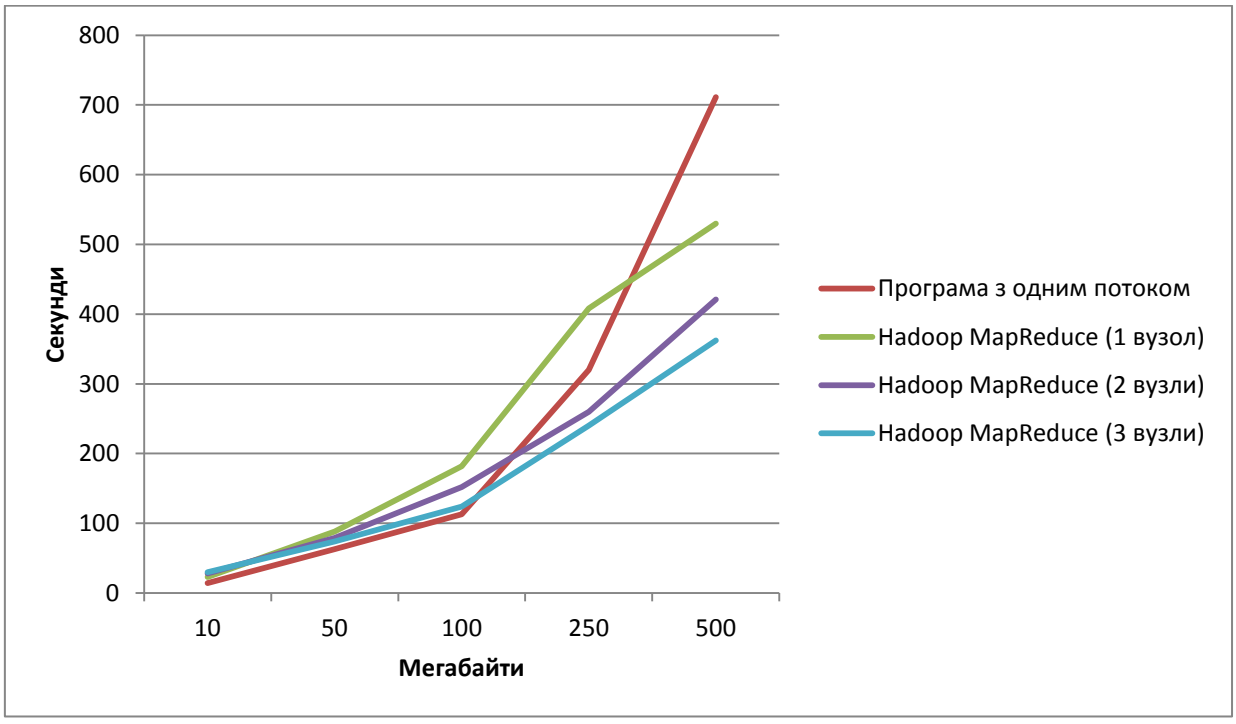

Рис. 3.29 – Час роботи прямого та розподіленого алгоритмів на сервері з різною кількістю вузлів (1, 2 та 3) для даних розміром до 500 МБ.

Варто відмітити, що швидкість роботу даного алгоритму значно повільніша за попередній і максимальний час обробки даних зайняв близько двох годин. Проте, з графіків помітно, що використовувати Hadoop MapReduce доцільно, але доцільніше це робити для даних більших за 500 МБ.

3.4 Висновок

У даному розділі була представлена апробація запропонованих моделей обробки великих масивів даних, а саме – апробація розподілених алгоритмів CURE та SON.

Для випробовування моделей необхідно було налаштувати Hadoop MapReduce на локальному комп'ютері (1 вузол) та на сервері, який складається з декількох ком'ютерів (2 та 3 вузли). Послідовність дій для налаштування цієї інфраструктури описана в першій частині даного розділу.

У другій частині розділу представлені UML-діаграми, що описують роботу та архітектуру розроблених програм. Для опису використовували 4 типи діаграм: діаграма прецедентів, діаграма класів, діаграма послідовностей та діаграма діяльностей. За допомогою уніфікованої мови моделювання значно простіше зрозуміти алгоритм роботи та архітектуру розроблених програм.

Апробація алгоритмів була проведена на локальному комп'ютері та на сервері кафедри Системного проектування, що складався з трьох вузлів. З отриманих результатів можна зробити висновок, що розроблені моделі доцільно використовувати для даних більших за 500 МБ, а також важливо, що зі збільшенням кількості вузлів, швидкість обробки зростає. Недоліком даного дослідження є те, що тестування відбувалось більше на тестових даних, аніж на реальних. Причиною цього є відсутність доступу до баз великих даних в інтернеті, бо частина даних є платною, інша частина – це дані об'ємом до 500-1000 МБ.

4 РЕАЛІЗАЦІЯ СТАРТАП-ПРОЕКТУ

4.1 Опис ідеї та технологічний аудит стартап-проекту

У даному розділі описано економічне обґрунтування реалізації стартап-проекту на тему «Створення системи обробки великих даних».

Таблиця 4.1 – Опис ідеї стартап-проекту

Зміст ідеї	*Напрямки застосування*	*Вигоди для користувача*
Ідея полягає у тому, щоб створити системи, завдяки якій можливо швидко виконувати операції над великими даними у розподіленому середовищі.	1. Вирішення задачі кластеризації	Користувачу необхідно лише буде завантажити свої дані, необхідні для кластеризації, запустити програму і отримати результат.
	2. Вирішення задачі пошуку частих предметних наборів	В користувача буде можливість швидко вирішувати задачу пошуку частих предметних наборів (наприклад, для аналізу кошику супермаркету, перевірки на плагіат, тощо).

Аналіз потенційних техніко-економічних переваг ідеї порівняно із пропозиціями конкурентів передбачає:

- визначення переліку техніко-економічних властивостей та характеристик ідеї
- визначення попереднього кола конкурентів (проектів-конкурентів) або товарів-замінників чи товарів-аналогів, що вже існують на ринку, та проводиться збір інформації щодо значень техніко-економічних показників для ідеї власного проекту та проектів-конкурентів відповідно до визначеного вище переліку;
- проводиться порівняльний аналіз показників: для власної ідеї визначаються показники, що мають а) гірші значення (W, слабкі); б) аналогічні (N, нейтральні) значення; в) кращі значення (S, сильні).

Таблиця 4.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	*Техніко-економічні характеристики ідеї*	*(потенційні) товари/концепції конкурентів*				*W (слабка сторона)*	*N (нейтральна сторона)*	*S (сильна сторона)*
		Мій проект	*Конкурент1*	*Конкурент2*	*Конкурент3*			
1.	Форма виконання	Програма	Програма	Веб-додаток	Програма			+
2.	Собівартість	Низька	Висока	Низька	Висока			+
3.	Наявність адміністратора для налаштування	Треба	Не треба, дистанційно	Треба	Треба		+	
4.	Наявність інтернету	Не треба	Необхідно	Не треба	Не треба			+
5.	Кросплатформеність	Ні	Так	Так	Ні	+		

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності.

В межах даного підрозділу необхідно провести аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення товару).

Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових:

- за якою технологією буде виготовлено товар згідно ідеї проекту?
- чи існують такі технології, чи їх потрібно розробити/доробити?
- чи доступні такі технології авторам проекту?

Таблиця 4.3 – Технологічна здійсненність ідеї проекту

№ п/п	*Ідея проекту*	*Технології її реалізації*	*Наявність технологій*	*Доступність технологій*
1.	Система для	Amazon Web Services	Наявна	Платна, недоступна

	обробки великих даних			
		Hadoop MapReduce	Наявна	Безкоштовна, доступна
Обрана технологія реалізації ідеї проекту: для створення системи обробки великих даних обрана система Hadoop MapReduce, яка є безкоштовною та доступною.				

4.2 Аналіз ринкових можливостей

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Спочатку проводимо аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку.

Таблиця 4.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	*Показники стану ринку (найменування)*	*Характеристика*
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн/ум.од	20000 грн./ум.од
3	Динаміка ринку (якісна оцінка)	Зростає/спадає/стагнує
4	Наявність обмежень для входу (вказати характер обмежень)	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	R = (3000000 * 100) / (1000000 * 12) = 25%

Рентабельність — поняття, що характеризує економічну ефективність виробництва, за якої за рахунок грошової виручки від реалізації продукції (робіт, послуг) повністю відшкодовує витрати на її виробництво й одержується прибуток як головне джерело розширеного відтворення

[http://buklib.net/books/29473/]. Суть одного із найважливіших методів оцінки економічної ефективності інвестицій полягає у розрахунку їх середньої рентабельності за формулою [http://pidruchniki.com/1566072162240/turizm/prognozuvannya_efektivnosti_investitsiynogo_proektu]

$$R = \frac{P}{1 \times n} \times 100$$

де Р - прибуток за час експлуатації проекту; / - повна сума інвестиційних витрат; п - час експлуатації проекту. Інвестувати грошові засоби доцільно тоді, коли від цього можна отримати більший прибуток, ніж від їх зберігання у банку. Порівнюючи середньорічну рентабельність інвестицій зі ставкою банківського відсотка, можна дійти висновку, що вигідніше [http://pidruchniki.com/1566072162240/turizm/prognozuvannya_efektivnosti_investitsiynogo_proektu].

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	*Потреба, що формує ринок*	*Цільова аудиторія (цільові сегменти ринку)*	*Відмінності у поведінці різних потенційних цільових груп клієнтів*	*Вимоги споживачів до товару*
1.	Необхідно програмне забезпечення для обробки великих наборів даних	Потенційними цільовими групами є дослідницькі центри, університети та компанії, специфіка роботи яких пов’язана із великими даними	Цільова група займається дослідженнями або має обробляти даних великих розмірів	Рішення має бути швидким, що дозволить концентруватись на інших задачах, і використовувати даний програмний продукт як інструментарій

Після визначення потенційних груп клієнтів проводиться аналіз ринкового середовища: складаються таблиці факторів, що сприяють ринковому

впровадженню проекту, та факторів, що йому перешкоджають (табл. №№ 6-7). Фактори в таблиці подавати в порядку зменшення значущості.

Ринкові можливості - це сприятливі обставини, які підприємство може використовувати для отримання переваг. Слід зазначити, що можливостями з погляду SWOT-аналізу є не всі можливості, які існують на ринку, а тільки ті, які можна використовувати [http://pidruchniki.com/1974070454048/menedzhment/strategichniy_analiz_pidpriyemstva].

Ринкові загрози - події, настання яких може несприятливо вплинути на підприємство.
[http://pidruchniki.com/1974070454048/menedzhment/strategichniy_analiz_pidpriyemstva].

Таблиця 4.6 – Фактори загроз

№ п/п	*Фактор*	*Зміст загрози*	*Можлива реакція компанії*
1.	Конкуренція	Вихід на ринок великої компанії	1) Вихід з ринку 2) Запропонувати великій компанії поглинути себе 3) Передбачити додаткові переваги власного ПЗ для того, щоб повідомити про них саме після виходу міжнародної компанії на ринок
2.	Зміна потреб користувачів	Користувачам необхідне програмне забезпечення з іншим функціоналом	1) Передбачити можливість додавання нового функціоналу до створюваного ПЗ

Таблиця 4.7 – Фактори можливостей

№ п/п	*Фактор*	*Зміст можливості*	*Можлива реакція компанії*
1	Зростання можливостей	Зростанням держфінансування	Запропонувати свої послуги державним підприємствам

	потенційних покупців	досліджень у галузі обробки великих наборів даних	та дослідницьким центрам
2	Зниження довіри до конкурента 1	У ПЗ конкурента №1 нещодавно була знайдена помилка, завдяки якій дані досліджень усіх клієнтів стали доступні в інтернеті для всіх користувачів	При виході на ринок звертати увагу покупців на безпеку нашого ПЗ

Надалі проводиться аналіз пропозиції: визначаються загальні риси конкуренції на ринку (табл. 8).

Таблиця 4.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	*В чому проявляється дана характеристика*	*Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)*
1. Вказати тип конкуренції - досконала	Існує 3 фірми-конкурентки на ринку	Врахувати ціни конкурентних компаній на початкових етапах створення бізнесу, реклама (вказати на конкретні переваги перед конкурентами)
2. За рівнем конкурентної боротьби - міжнародний	Одна з компаній – з ішної країни, дві – з України	Додати можливість вибору мови ПЗ, щоб легше було у майбутньому вийти на міжнародний ринок
3. За галузевою ознакою - внутрішньогалузева	Конкуренти мають ПЗ, яке використовується лише всередині даної галузі	Створити основу ПЗ таким чином, щоб можна було легко переробити дане ПЗ для використання у інших галузях
4. Конкуренція за видами товарів: - товарно-видова	Види товарів є однаковими, а саме – програмне забезпечення	Створити ПЗ, враховуючи недоліки конкурентів
5. За характером конкурентних переваг	Вдосконалення технології створення ПЗ,	Використання менш дорогих технологій для

- нецінова	щоб собівартість була нижчою	розробки, ніж використовують конкуренти
6. За інтенсивністю - не марочна	Бренди відсутні	-

Після аналізу конкуренції проводиться більш детальний аналіз умов конкуренції в галузі. *М. Портер* вирізняє п'ять основних факторів, що впливають на привабливість вибору ринку з огляду на характер конкуренції.

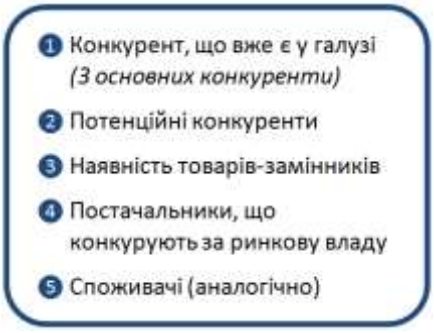

Рис. 4.1 – Складові моделі 5 сил М. Портера

Сильні позиції компанії за кожним з факторів означають її можливості забезпечити необхідні темпи обороту капіталу та її здатність впливати на інших агентів ринку, диктуючі їм власні умови співпраці.

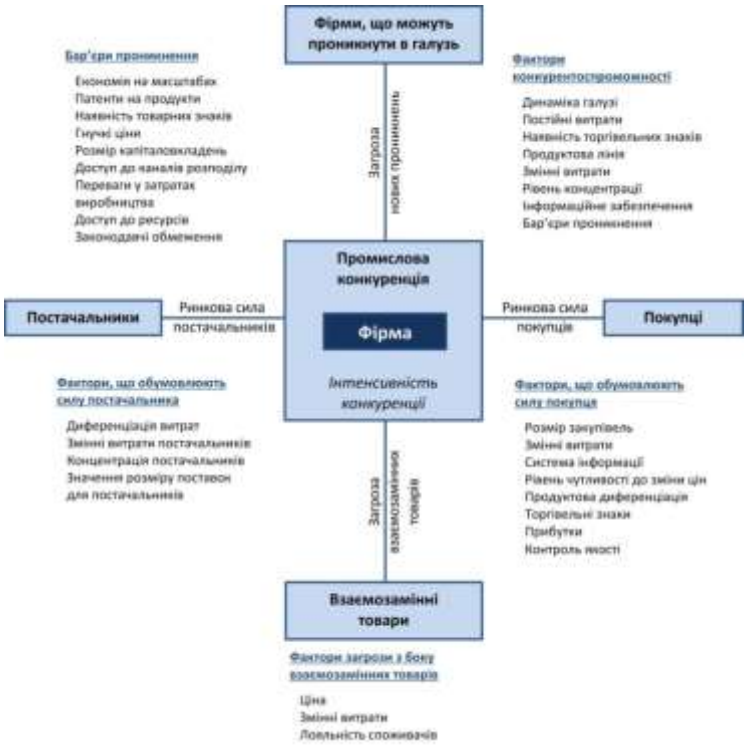

Рис. 4.2 – Модель 5 сил М. Портера для аналізу конкуренції в галузі

Характеристики факторів моделі відрізняються для різних галузей та змінюються із часом. Сила кожного фактору є функцією від структури галузі та її техніко-економічних характеристик.

На основі аналізу складових моделі 5 сил М. Портера розробляється перелік факторів конкурентоспроможності для певного ринку.

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	*Прямі конкуренти в галузі*	*Потенційні конкуренти*	*Постачальники*	*Клієнти*	*Товари-замінники*
	Навести перелік прямих конкурентів	*Визначити бар'єри входження в ринок*	*Визначити фактори сили постачальників*	*Визначити фактори сили споживачів*	*Фактори загроз з боку замінників*
Висновки:	Існує 3 конкуренти на ринку. Найбільш схожим за виконанням є конкурент 1, так як його рішення також представлене у вигляді програми.	Так, можливості для входу на ринок є, бо наше рішення спрощує та пришвидшує роботу спеціаліста.	Постача-льники відсутні.	Важливим для користу-вача є швидкість роботи ПЗ.	Товари-замінники можуть використати більш дешеву техно-логію створення ПЗ та зменшити собі-вартість товару.

За результатами аналізу таблиці робиться висновок щодо принципової можливості роботи на ринку з огляду на конкурентну ситуацію. Також робиться висновок щодо характеристик (сильних сторін), які повинен мати проект, щоб бути конкурентоспроможним на ринку. Другий висновок враховується при формулюванні переліку факторів конкурентоспроможності у п. 3.6.

На основі аналізу конкуренції, проведеного в п. 3.5 (табл. 9), а також із урахуванням характеристик ідеї проекту (табл. 2), вимог споживачів до товару

(табл. 5) та факторів маркетингового середовища (табл. №№ 6-7) визначається та обґрунтовується перелік факторів конкурентоспроможності. Аналіз оформлюється за табл. 10.

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	*Фактор конкурентоспроможності*	*Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)*
1.	Виконання ПЗ у вигляді програми у розподіленій системі	Це рішення дозволяє швидко обробляти дані розміром значно більше за 4 ГБ.
2.	Простота інтерфейсу користувача	Користувач має лише завантажити дані і запустити програму на виконання.

За визначеними факторами конкурентоспроможності (табл. 10) проводиться аналіз сильних та слабких сторін стартап-проекту (табл. 11). Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін.

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	*Фактор конкурентоспроможності*	*Бали 1-20*	*Рейтинг товарів-конкурентів у порівнянні з нашим підприємством*						
			–3	*–2*	*–1*	*0*	*+1*	*+2*	*+3*
1	Виконання ПЗ у вигляді програми у розподіленій системі	15			+				
2	Простота інтерфейсу користувача	20	+						

Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками (прогнозованими результатами) впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення. Наприклад: зниження доходів потенційних споживачів – фактор загрози, на основі якого можна зробити

прогноз щодо посилення значущості цінового фактору при виборі товару та відповідно, – цінової конкуренції (а це вже – ринкова загроза).

Таблиця 4.12 – SWOT-аналіз стартап-проекту

Сильні сторони: простий інтерфейс користувача, виконання ПЗ у вигляді програми	Слабкі сторони: необхідно мати власний сервер (або орендувати його)
Можливості: у конкурента 1 виявлена проблема із безпекою ПЗ, додаткове держфінансування для досліджень у підприємствах, які є потенційними покупцями	Загрози: конкуренція, зміна потреб користувачів

На основі SWOT-аналізу розробляються альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок. Визначені альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів.

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	*Альтернатива (орієнтовний комплекс заходів) ринкової поведінки*	*Ймовірність отримання ресурсів*	*Строки реалізації*
1.	Створення програми на основі Hadoop MapReduce для обробки великих даних	80%	6 місяців
2.	Створення програми на основі без використання будь-яких фреймворків для обробки даних	30%	12 місяців

З означених альтернатив обирається та, для якої: а) отримання ресурсів є більш простим та ймовірним; б) строки реалізації – більш стислими. Тому обираємо альтернативу 1.

4.3 Розробка ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів.

Таблиця 4.14 – Вибір цільових груп потенційних споживачів

№ п/п	*Опис профілю цільової групи потенційних клієнтів*	*Готовність споживачів сприйняти продукт*	*Орієнтовний попит в межах цільової групи (сегменту)*	*Інтенсивність конкуренції в сегменті*	*Простота входу у сегмент*
1.	Університе-тські дослідже-ння	Час виконання не є критичним у даному випадку, а однією із найголовні-ших переваг є саме це	Дослідження в області великих даних проводяться постійно	Існує 3 конкуренти, які надають схожі, але менш швидкі рішення.	У сегмент увійти непросто, бо бюрократія всередині університеті в занадто складна, також відсутня дуже впливова перевага ПЗ
2.	Дослід-ницькі центри	Пришвидшення часу виконання додає їм можливості виконати більшу кількість замовлень	Дослідження в області великих даних проводяться постійно, до того ж – за замовленням приватних компаній		Маючи перевагу у зручності інтерфесу користувача, що пришвидшує роботу та отримання результату, вийти на ринок

					нескладно
3.	Підприє-мства	Пришвидшення часу виконання додає їм можливості виконати більшу кількість замовлень, до того ж в цьому році з держбюджету виділені додаткові кошти на цей напрям досліджень			Маючи перевагу у тому, що рішення є зручним для користування і швидким вийти на ринок не є складно
Які цільові групи обрано: обираємо дослідницькі центри та підприємства, які підтримуються держзамовленням					

За результатами аналізу потенційних груп споживачів (сегментів) автори ідеї обирають цільові групи, для яких вони пропонуватимуть свій товар, та визначають стратегію охоплення ринку.

Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку. За М. Портером, існують три базові стратегії розвитку, що відрізняються за ступенем охоплення цільового ринку та типом конкурентної переваги, що має бути реалізована на ринку (за витратами або визначними якостями товару).

Стратегія лідерства по витратах передбачає, що компанія за рахунок чинників внутрішнього і/або зовнішнього середовища може забезпечити більшу, ніж у конкурентів маржу між собівартістю товару і середньоринковою ціною (або ж ціною головного конкурента).

Стратегія диференціації передбачає надання товару важливих з точки зору споживача відмітних властивостей, які роблять товар відмінним від товарів конкурентів. Така відмінність може базуватися на об’єктивних або

суб'єктивних, відчутних і невідчутних властивостях товару(у ширшому розумінні – комплексі маркетингу), бути реальною або уявною. Інструментом реалізації стратегії диференціації є ринкове позиціонування.

Стратегія спеціалізації передбачає концентрацію на потребах одного цільового сегменту, без прагнення охопити увесь ринок. Мета тут полягає в задоволенні потреб вибраного цільового сегменту краще, ніж конкуренти. Така стратегія може спиратися як на диференціацію, так і на лідерство по витратах, або і на те, і на інше, але тільки у рамках цільового сегменту. Проте низька ринкова доля у разі невдалої реалізації стратегії може істотно підірвати конкурентоспроможність компанії.

Таблиця 4.15 – Визначення базової стратегії розвитку

№ п/п	*Обрана альтернатива розвитку проекту*	*Стратегія охоплення ринку*	*Ключові конкурентоспроможні позиції відповідно до обраної альтернативи*	*Базова стратегія розвитку**
1.	Створення програми з використанням Hadoop MapReduce	Ринкове позіціювання	Простота інтерфейсу, пришвидшення роботи	Диференціації

Наступним кроком є вибір стратегії конкурентної поведінки.

Стратегія лідера. Залежно від міри сформованості товарного(галузевого) ринку, характеру конкурентної боротьби компанії-лідери обирають одну з трьох стратегій: розширення первинного попиту, оборонну або наступальну стратегію або ж застосувати демаркетинг або диверсифікацію.

Стратегія розширення первинного попиту доцільна у разі, якщо фірмі-лідерові недоцільно розмінюватися на боротьбу з невеликими конкурентами, вона може отримати велику економічну віддачу від розширення первинного рівня попиту. В цьому випадку компанія займається реалізацією заходів по формуванню попиту(навчанню споживачів користуванню товаром, формування

регулярного попиту, збільшення разового споживання), також пропаганду нових напрямів застосувань існуючих товарів, виявлень нових груш споживачів.

У міру зростання ринку, його становлення позиції компанії-новатора починають атакувати конкуренти-імітатори. В цьому випадку, компанія може вибрати оборонну стратегію, метою якої є захист власної ринкової долі.

Наступальна стратегія припускає збільшення своєї частки ринку. При цьому переслідувана мета полягає в подальшому підвищенні прибутковості роботи компанії на ринку за рахунок максимального використання ефекту масштабу.

Якщо фірма потрапляє під дію антимонопольного законодавства, вона може удатися до стратегії демаркетинга, що припускає скорочення своєї частки ринку, зниження рівня попиту на деяких сегментах за рахунок підвищення ціни. При цьому ставиться завдання недопущення на ці сегменти конкурентів, а компенсація втрат прибутку через зменшення обсягів виробництва компенсується встановленням надвисоких цін.

Стратегія виклику лідера. Стратегію виклику лідерові найчастіше вибирають компанії, які є другими, третіми на ринку, але бажають стати лідером ринку. Теоретично, ці компанії можуть прийняти два стратегічні рішення: атакувати лідера у боротьбі за частку ринку або ж йти за лідером.

Рішення атакувати лідера є досить ризикованим. Для його реалізації потрібні значні фінансові витрати, know – how, краще співвідношення «ціна-якість», переваги в системі розподілу і просування і т. д. У разі не реалізації цієї стратегії, компанія може бути відкинута на аутсайдерські позиції на досить довгий час. Залежно від цього компанія може вибрати одну з альтернативних стратегій: фронтальної або флангової атаки.

Стратегія наслідування лідеру. Компанії, що приймають слідування за лідером – це підприємства з невеликою часткою ринку, які вибирають адаптивну лінію поведінки на ринку, усвідомлюють своє місце на нім і йдуть у

фарватері фірм-лідерів. Головна перевага такої стратегії – економія фінансових ресурсів, пов'язаних з необхідністю розширення товарного(галузевого) ринку, постійними інноваціями, витратами на утримання домінуючого положення.

Стратегія заняття конкурентної ніші. При прийнятті стратегії зайняття конкурентної ніші (інші назви – стратегія фахівця або нішера) компанія в якості цільового ринку вибирає один або декілька ринкових сегментів. Головна особливість – малий розмір сегментів/сегменту. Ця конкурентна стратегія являється похідною від такої базової стратегії компанії, як концентрація. Головне завдання для компаній, що вибирають стратегію нішера або фахівця, – це постійна турбота про підтримку і розвиток своєї конкурентної переваги, формування лояльності і прихильності споживачів, підтримка вхідних бар'єрів.

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	*Чи є проект «першопрохідцем» на ринку?*	*Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?*	*Чи буде компанія копіювати основні характеристики товару конкурента, і які?*	*Стратегія конкурентної поведінки**
1.	Ні	Так	Буде, а саме: основною задачею ПЗ є обробка великих даних (конкуренти 1, 2, 3), простий інтерфейс користувача (конкурент 2)	Зайняття конкурентної ніші

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту, а також в залежності від обраної базової стратегії розвитку та стратегії конкурентної поведінки розробляється стратегія

позиціонування, що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 4.17 – Визначення стратегії позиціонування

№ п/п	*Вимоги до товару цільової аудиторії*	*Базова стратегія розвитку*	*Ключові конкурентоспроможні позиції власного стартап-проекту*	*Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)*
1.	Простота інтерфейсу, швидкість	Дифе-ренціації	Простота користувацького інтерфейсу, що дозволяє пришвидшити та спростити роботу працівників, швидкість роботи, що дозволяє підвищити швидкість експериментів	Швидкість, простота, безпека

4.4 Розробка маркетингової програми

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у табл. 18 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару.

Таблиця 4.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	*Потреба*	*Вигода, яку пропонує товар*	*Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити*
1.	Швидкість обробки даних	ПЗ працює у розподіленій системі і значно пришвидшує обробку даних	Переваги у швидкості
2.	Спрощення інтерфейсу користувача	Простота роботи з ПЗ	Користувачам не потрібно замислюватись над тим, як саме обробити велики дані. Лише необхідно надати дані, та запустити програму.

Надалі розробляється трирівнева маркетингова модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (табл. 19).

1-й рівень. При формуванні задуму товару вирішується питання щодо того, засобом вирішення якої потреби і / або проблеми буде даний товар, яка його основна вигода. Дане питання безпосередньо пов'язаний з формуванням технічного завдання в процесі розробки конструкторської документації на виріб.

2-й рівень. Цей рівень являє рішення того, як буде реалізований товар в реальному/ включає в себе якість, властивості, дизайн, упаковку, ціну.

3-й рівень. Товар з підкріпленням (супроводом) - додаткові послуги та переваги для споживача, що створюються на основі товару за задумом і товару в реальному виконанні (гарантії якості , доставка, умови оплати та ін). [http://pidruchniki.com/19670511/marketing/razrabotka_produktovoy_strategii_kompanii#891] [http://marketing-helping.com/konspekti-lekcz/21-konspekt-lekczj-qosnovi-marketinguq/412-marketingove-ponyattya-ta-klasifkaczya-vidv-tovaru.html]

Таблиця 4.19 – Опис трьох рівнів моделі товару

Рівні товару	*Сутність та складові*		
I. Товар за задумом	Об’єкт допомогає фахівцям у області великих даних вирішувати задачу кластеризації та пошуку частих предметних наборів дуже швидко. Користувач має лише завантажити необхідні дані у систему та запустити обробку.		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Простота інтерфейсу користувача 2. Швидкість роботи 3. Безпека згідно до світових стаднартів	-	-
	Якість: згідно до стандарту ISO 4444 буде проведено тестування		
	Маркування відсутнє.		
	Моя компанія. «Біг Дата – В»		
III. Товар із підкріплю-нням	1-місячна пробна безкоштовна версія та безкоштовне встановлення		
	Постійна підтримка для користувачів		
За рахунок чого потенційний товар буде захищено від копіювання: ноу-хау.			

Після формування маркетингової моделі товару слід особливо відмітити – чим саме проект буде захищено від копіювання. Захист може бути організовано за рахунок захисту ідеї товару (захист інтелектуальної власності), або ноу-хау, чи комплексне поєднання властивостей і характеристик, закладене на другому та третьому рівнях товару.

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проекту), яке передбачає аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів (табл. 20). Аналіз проводиться експертним методом.

Таблиця 4.20 – Визначення меж встановлення ціни

№ п/п	*Рівень цін на товари-замінники*	*Рівень цін на товари-аналоги*	*Рівень доходів цільової групи споживачів*	*Верхня та нижня межі встановлення ціни на товар/послугу*
1.	25000	30000	200000	20000

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (табл. 21).

Таблиця 4.21 – Формування системи збуту

№ п/п	*Специфіка закупівельної поведінки цільових клієнтів*	*Функції збуту, які має виконувати постачальник товару*	*Глибина каналу збуту*	*Оптимальна система збуту*
1.	Купують ПЗ та роблять щорічні внески для подовження ліцензії	Продаж	0(напряму), 1(через одного посередника)	Власна та через посередників

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (табл. 22).

Таблиця 4.22 – Концепція маркетингових комунікацій

№ п/п	*Специфіка поведінки цільових клієнтів*	*Канали комунікацій, якими користуються цільові клієнти*	*Ключові позиції, обрані для позиціонування*	*Завдання рекламного повідомлення*	*Концепція рекламного звернення*
1.	Купівля ПЗ через інтернет, робота з ПЗ на комп'ютерах з різними ОС	Інтернет	Швидкодія, простота використання, безпека	Показати переваги ПЗ, у тому числі і перед конкурентами	Демо-ролик із використання

4.5 Елементи фінансової підтримки стартапу та аналіз ризиків

Таблиця 4.23 – Сукупні інвестиційні витрати на реалізацію стартап-проекту

№ з/п	*Стаття витрат*	*Сукупні витрати, тис. грн.*
1.	*Загальні початкові витрати*	405
1.1.	Проведення пошукових та прикладних досліджень	5
1.2.	Розробка проектних матеріалів і ТЕО	10
1.3.	Робоче проектування і прив'язка проекту	10
1.4.	Витрати на придбання обладнання та устаткування та пристроїв	100
1.5.	Витрати на придбання нематеріальних активів	40
1.6.	Витрати на утримання обладнання та приміщень	40

1.7.	Витрати на передвиробничі маркетингові дослідження	50
1.8.	Витрати на створення збутової мережі	50
1.9.	Витрати на просування та рекламу	50
1.10	Оплата юридичних послуг	50
2.	*Витрати на матеріальні ресурси*	0
2.1.	матеріали	0
2.3.	комплектуючі	0
3.3.	сировина	0
3.	*Витрати на оплату праці команди стартапу*	300
	Разом:	705

Таблиця 4.24 – Визначення основних фінансово-економічних показників проекту

№ з/п	*Стаття витрат*	*Сукупні витрати, тис. грн.*
1.	Обсяг виробництва продукції в натуральних показниках	*1*
2.	Собівартість одиниці продукції, тис. грн.	*705000*
3.	Собівартість виробництва продукції, тис. грн. *(3 = 1 · 2)*	*705000*
4.	Обсяг реалізації продукції в натуральних показниках	*100*
5.	Ціна реалізації продукції без ПДВ, тис. грн.	*20000*
6.	Виручка від реалізації продукції без ПДВ, тис. грн. *(6 = 4 · 5)*	*2000000*
7.	Податок на додану вартість (ПДВ), тис. грн.	*200000*
8.	Валовий прибуток *(8 = 6 – 3)*	*1295000*

9.	Податок на прибуток	*259000*
10.	Чистий прибуток *(10 = 8 – 9)*	*1036000*

Рентабельність продажів (або маржа прибутку) показує, скільки прибутку приносить кожна гривня з обсягу реалізації. Маржу прибутку, як правило, визначають окремо за кожним видом діяльності або за кожною групою реалізованої продукції за формулою:

$$Rs = \frac{П}{B}100\%,$$

де П – прибуток;

В – виручка від реалізації продукції.

$$Rs = 51.8\%$$

Період окупності проекту відображає час, який потрібен для того, щоб сума надходжень від реалізації проекту відшкодувала суму витрат на його впровадження. Період окупності звичайно вимірюється в роках або місяцях та може бути розрахований за формулою:

$$PBP = \frac{II}{ACI}$$

де PBP – період окупності інвестицій, роки;

II – сума інвестиційних витрат, тис. грн.;

ACI – щорічні надходження (річний чистий прибуток), тис. грн.

$$PBP = 0.35(роки)$$

Рентабельність довгострокових інвестицій – коефіцієнт повернення інвестицій, показник рентабельності вкладень, що у відсотковому співвідношенні демонструє прибутковість (у разі, коли його значення більше 100%) або збитковість (у разі, коли його значення менше 100%) інвестицій в проект. Якщо розрахований показник менший 100%, то інвестиційні вкладення не окупуються. Показник може бути розрахований за формулою:

$$ROI = \frac{Д - С}{ІІ} \cdot 100\%$$

де Д – дохід (виручка від реалізації продукції), тис. грн.;

С – повна собівартість, тис. грн.;

ІІ – сума інвестиційних витрат, тис. грн.

$$ROI = 46.9\%$$

Таблиця 4.25 – Програма запобігання та реагування на ризики проекту

Ризики проекту	*Заходи запобігання ризиків*	*Заходи реагування при виникненні ризиків*
Вихід з ладу системи контролю версій	Збереження копій вихідних кодів проекту, проектної документації та інших даних на інших серверах	Отримання копії даних з інших серверів
Звільнення члена команди	Детальна декомпозиція завдань, щоб зробити кожне з них максимально простим та незалежним. Використання систем контролю версій.	Продовження роботи без цієї людини
Збільшення собівартості через зміну ліцензії Hadoop MapReduce	Немає	Необхідне додаткове фінансування для продовження ліцензії
Зміна системного Hadoop MapReduce	Приймати участь в обговореннях переваг та недоліків цієї технології на офіційному сайті Hadoop, абстрагувати деякі рівні ПЗ для незалежності від системного API	Змінити код у тих місцях, де використовується системне API

4.6 Висновок

Згідно до проведених досліджень існує можливість ринкової комерціалізації проекту. Також, варто відмітити, що існують перспективи впровадження з огляду на потенційні групи клієнтів, бар’єри входження не є високими, а проект має дві значні переваги перед конкурентами.

Для успішного виконання проекту необхідно реалізувати програму із використанням засобів Hadoop MapReduce. В рамках даного дослідження були розраховані основні фінансово-економічні показники проекту, а також проведений менеджмент потенційних ризиків. Проаналізувавши отримані результати, можна зробити висновок, що подальша імплементація є доцільною.

ВИСНОВОК

1. Була досліджена технологія MapReduce та її альтернативи. Після аналізу наукових публікацій останніх років можна зробити висновок, що реляційні бази даних, ‘хмарні’ технології та MapReduce не є конкурентними технологіями, а є частиною аналітичної екосистеми, яка дозволить отримати необхідну інформацію з великих даних. Ці технології можуть бути використані разом для досягнення кращих результатів. Існують різні способів об’єднання цих технологій, наприклад, розробка баз даних, які працюють в ‘хмарі’, а дані перед обробкою мовою SQL фільтруються за допомогою MapReduce. Варто відмітити, що грід-обчислення є альтернативою до MapReduce. З одного боку, з використанням грід-технологій можна вирішувати більш широке коло задач, а з іншого – MapReduce дозволяє не писати програмний код для роботи з мережевими протоколами, системними викликами, а працювати на більш високому рівні.

2. У даній роботі були запропоновані дві моделі обробки великих масивів даних, в основі яких – технологія MapReduce. Перша модель дозволяє вирішувати задачу кластеризації, а друга модель – задачу пошуку частих предметних наборів.

3. В основі моделі для вирішення задачі кластеризації, знаходиться модифікований алгоритм CURE, ця модифікація дозволяє обробляти дані паралельно. Також перевагою даної моделі є те, що використовується репрезентативна вибірка кластеру і не зберігаються усі його точки, що дозволяє зменшити завантаженість пам’яті комп’ютера. Іншою перевагою є те, що не потрібно завантажувати всю вхідну вибірку даних у оперативну пам’ять.

4. В основі другої моделі, для вирішення задачі пошуку частих предметних наборів, знаходиться модифікований алгоритм SON. Аналогічно до попереднього пункту, алгоритм модифіковано для того, щоб була можливість

обробляти дані паралельно. У цьому випадку за два проходи по даним знаходяться часті набори, що задовільняють заданим користувачем параметрам.

5. Для апробації запропонованих моделей було написано 4 програми на мові Java. Дві програми – реалізація алгоритмів CURE та SON, інші дві – реалізація їх модифікацій з використанням технології MapReduce.

6. Спочатку для апробації було налаштовано фреймвор Hadoop MapReduce на локальному комп'ютері. Були згенеровані тестові вибірки даних обсягом від 10 МБ до 6 ГБ. Відповідно, було проаналізовано швидкість роботи двох програм без використання MapReduce та двох програм з використанням даної технології. На основі даних результатів можна зробити висновок, що запропоновані моделі дають правильний результат, а також – що використання даних моделей дозволяє пришвидшити обробку даних. Особливо це помітно для даних об'ємом більше за 500 МБ.

7. Після завершення перевірки роботи програм локально, фреймворк Hadoop MapReduce був налаштований на сервері кафедри Системного програмування, та було створено три обчислювальні вузли на ньому. Відповідно, було проведено аналогічне дослідження, як описано у попередньому пункті, проте з однією відмінністю – був проведений аналіз швидкодії запропонованих моделей при використанні різної кількості обчислювальних вузлів. З отриманих результатів можна зробити висновок, що із збільшенням кількості вузлів – швидкість обробки даних збільшується. Іншим висновком є те, що із збільшенням розміру вхідних даних, використання технології MapReduce стає більш доцільним.

8. В рамках даного дослідження була запропонована реалізація стартап-проекту, розраховані основні фінансово-економічні показники та проведений менеджмент потенційних ризиків. Проаналізувавши отримані результати, можна зробити висновок, що подальша реалізація є доцільною та існує можливість ринкової комерціалізації проекту. Для успішного виконання проекту необхідно реалізувати програми із використанням засобів Hadoop MapReduce.

ПЕРЕЛІК ПОСИЛАНЬ

1. Борис Т. В. Порівняльний аналіз технології паралельного обчислення великих масивів даних MapReduce / Т. В. Борис, М. О. Алєксєєв. // Second International Conference "Cluster Computing" CC 2013 (Ukraine, Lviv, June 3-5, 2013). – 2013. – №2. – С. 54–57.
2. Браун М. Интеллектуальный анализ данных в документо-ориентированном мире [Електронний ресурс] / Браун Мартин. – 2013. – Режим доступу до ресурсу: https://www.ibm.com/developerworks/ru/library/ba-datamining/.
3. Волосюк Ю. В. Аналіз алгоритмів кластеризації для задач інтелектуального аналізу даних / Ю. В. Волосюк. // Інформаційні технології. – 2014. – С. 112–119.
4. Гаврилова А. А. Використання Алгоритмів Кластеризації Для Первинного Аналізу Стану Корпоративного Управління / А. А. Гаврилова. // Інформаційні технології та системи управління. – С. 101.
5. Дрейпер, У. Стартапы : профессиональные игры Кремниевой долины / У. Дрейпер ; предисл. Э. Шмидта ; пер. с англ. В. Егорова. – Москва : Эксмо, 2012. – 378 с.
6. Кава А. Введение в YARN [Електронний ресурс] / Адам Кава // IBM Developers Work. – 2014. – Режим доступу до ресурсу: https://www.ibm.com/developerworks/ru/library/bd-yarn-intro/.
7. Маллинс, Дж. Поиск бизнес-модели : как спасти стартап, вовремя сменив план / Дж. Маллинс, Р. Комисар ; пер. с англ. М. Пуксант и Е. Бакушевой. – Москва : Манн, Иванов и Фербер, 2012. – 329 с.
8. Мансурова М. Е. Применение технологии MapReduce Hadoop для кластеризации больших объемов данных / М. Е. Мансурова, А. С. Шоманов, Б. Н. Тулепбергенов. // Вестник КазНУ, сер. мат., мех., инф.. – 2013. – №3. – С. 70–82.

9. Орешков В. Поиск последовательных шаблонов [Електронний ресурс] / В. Орешков // BaseGroup Lab. Технологии анализа данных – Режим доступу до ресурсу: https://basegroup.ru/community/articles/sequential-patterns-1.
10. Офіційна сторінка Apache Spark [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: http://spark.apache.org/docs.
11. Петренко А. Grid і інтелектуальна обробка даних Data Mining [Електронний ресурс] / Анатолій Петренко – Режим доступу до ресурсу: http://allted.kpi.ua/downloads/DataMining.pdf.
12. Петруненков А. Оценка коммерческой привлекательности проекта [Электронный ресурс] // Технологический бизнес. – 1999. – № 2. Режим доступа: http://www.techbusiness.ru/tb/archiv/number2/page01.htm
13. Порівняльний аналіз технології паралельного обчислення великих масивів даних MapReduce. // Second International Conference "Cluster Computing". – 2013. – С. 54–57.
14. Статистика указала на условия для появления стартапов, успешных как Google и Facebook [Электронный ресурс]. – Режим доступа: https://naked-science.ru/article/sci/statistika-ukazala-na-usloviya.
15. Тиль, П. От нуля к единице : как создать стартап, который изменит будущее / П. Тиль, Б. Мастерс; перевод с англ. – Москва : Альпина паблишер, 2015. – 188 с.
16. Френкс Б. Укрощение больших данных / Билл Френкс. – Москва: Манн, Иванов и Фербер, 2014. – 341 с.
17. Харниш, В. Правила прибыльных стартапов : как расти и зарабатывать деньги / В. Харниш ; пер. с англ. В. Хозинского. – Москва : Манн, Иванов и Фербер, 2012. – 279 с.
18. Цибульов П. М. Управління інтелектуальною власністю : монографія/ Цибульов П. М., Чеботарьов В. П., Зінов В. Г. , Суїні Ю., за ред. П. М. Цибульова. – К. : «К. І. С.», 2005. – 448 с.

19. An Enhanced Partial Order Curve Comparison Algorithm And Its Application To Analyzing Protein Folding Trajectories / H.Sun, H. Ferhatosmanoglu, M. Ota, Y. Wang. // Ohio-State University. – 2016. – С. 1–15.
20. Apache Hadoop YARN [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html.
21. Challenges for MapReduce in Big Data / [K. Grolinger, M. Hayes, W. Higashino та ін.]. // . Electrical and Computer Engineering Publications. – 2014. – №44.
22. Dean J. MapReduce: Simplified Data Processing on Large Clusters / J. Dean, S. Ghemawat. // Google, Inc.. – 2004. – P. 1–13.
23. El Gourii N. Introduction to the MapReduce Life Cycle [Електронний ресурс] / Nezha El Gourii // Supinfo International University. – 2016. – Режим доступу до ресурсу: https://www.supinfo.com/articles/single/2807-introduction-to-the-mapreduce-life-cycle.
24. Exploring Hadoop Framework: Hadoop Distributed File System (HDFS) [Електронний ресурс] // GENTLAB. – 2016. – Режим доступу до ресурсу: https://www.gentlab.com/articles/exploring-hadoop-framework-hadoop-distributed-file-system-hdfs.
25. Fritz J. Run Spark and Spark SQL on Amazon Elastic MapReduce [Електронний ресурс] / J. Fritz // Amazon Web Services. – 2013. – Режим доступу до ресурсу: https://aws.amazon.com/articles/4926593393724923.
26. Guha S. CURE: An Efficient Clustering Algorithm for Large Databases [Електронний ресурс] / S. Guha, R. Rastogi, S. Kyuseok – Режим доступу до ресурсу: http://www.cs.upc.edu/~bejar/amlt/material_art/DM%20clustering%20guha98cure.pdf.
27. Hadoop Cluster Setup [Електронний ресурс] // Apache Hadoop. – 2017. – Режим доступу до ресурсу: http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/ClusterSetup.html.

28. Hadoop: Setting up a Single Node Cluster [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html.
29. Haines S. Big Data Analysis with MapReduce and Hadoop [Електронний ресурс] / Steven Haines // InformIT. – 2013. – Режим доступу до ресурсу: http://www.informit.com/articles/article.aspx?p=2008905.
30. HDFS Users Guide [Електронний ресурс] // Apache Hadoop. – 2017. – Режим доступу до ресурсу: http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html.
31. Hornung T. Giant Data: MapReduce and Hadoop [Електронний ресурс] / T. Hornung, M. Przyjaciel-Zablocki, A. Schätzle // Admin-Magazine. – 2010. – Режим доступу до ресурсу: http://www.admin-magazine.com/HPC/Articles/MapReduce-and-Hadoop.
32. Introducing Big Data [Електронний ресурс] // GENTLAB. – 2016. – Режим доступу до ресурсу: https://www.gentlab.com/articles/introducing-big-data.
33. Jon Z. A profile of Apache Hadoop MapReduce computing efficiency [Електронний ресурс] / Zuanich Jon. – 2010. – Режим доступу до ресурсу: http://blog.cloudera.com/blog/2010/12/a-profile-of-hadoop-mapreduce-computing-efficiency-continued/.
34. Karloff H. A Model of Computation for MapReduce [Електронний ресурс] / H. Karloff, S. Suri, S. Vassilvitskii. – 2015. – Режим доступу до ресурсу: https://pdfs.semanticscholar.org/159e/fe23527149666be1b2b1c08853d74d413c1b.pdf.
35. Layton J. Why Use MapReduce? [Електронний ресурс] / Jeffrey Layton. – 2012. – Режим доступу до ресурсу: http://www.enterprisestorageforum.com/storage-management/why-use-mapreduce.html.
36. Leskovec J. Mining of Massive Datasets / J. Leskovec, A. Rajaraman, J. D. Ullman. – Cambridge: Cambridge University Press, 2014. – 495 p.

37. MapReduce: A programming paradigm that allows for massive scalability across hundreds or thousands of servers in a Hadoop cluster [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://www-01.ibm.com/software/data/infosphere/hadoop/mapreduce/.
38. Map-reduce as a Programming Model for Custom Computing Machines / [H. Jackson, C. Tsang, K. Tsoil and others.]. // Proc. IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM). – 2002. – P. 13–21.
39. MapReduce Tutorial [Електронний ресурс] // Apache Hadoop. – 2017. – Режим доступу до ресурсу: http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html.
40. Noll M. G. Writing an Hadoop MapReduce Program in Python [Електронний ресурс] / G. Noll. – 2012. – Режим доступу до ресурсу: http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/.
41. Pal K. Advantages of Hadoop MapReduce Programming [Електронний ресурс] / Pal // TutorialsPoint. – 2015. – Режим доступу до ресурсу: http://www.tutorialspoint.com/articles/advantages-of-hadoop-mapreduce-programming.
42. PFP: Parallel FP-Growth for Query Recommendation [Електронний ресурс] / [H. Li, Y. Wang, D. Zhang та ін.] – Режим доступу до ресурсу: https://static.googleusercontent.com/media/research.google.com/uk//pubs/archive/34668.pdf.
43. Qian Y. CURE-NS: a hierarchical clustering algorithm with new shrinking scheme / Y. Qian, Q. Shi, Q. Wang. // Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on. – 2002.
44. Rathbone M. Hadoop MapReduce Advanced Python Join Tutorial with Example Code [Електронний ресурс] / Rathbone. – 2016. – Режим доступу до ресурсу: https://blog.matthewrathbone.com/2016/02/09/python-tutorial.html.
45. Taylor R. An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics [Електронний ресурс] / R. Taylor // US National Library of Medicine National Institutes of Health Search databaseSearch

term Search. – 2010. – Режим доступу до ресурсу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3040523/.

46. Thakare A. Preprocessing, Mining & CURE Hierarchical Clustering forWebLog Mining / A. Thakare, M. Chaudhari. // - International Journal of Innovative Science, Engineering & Technology. – 2015. – С. 605–614.
47. Tom W. Hadoop: The Definitive Guide / White Tom. – Boston,USA: O'Reilly Media, 2009.
48. What Apache Spark Does [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://hortonworks.com/apache/spark/.
49. What is the difference between Apache Spark and Apache Hadoop (Map-Reduce) [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://www.quora.com/What-is-the-difference-between-Apache-Spark-and-Apache-Hadoop-Map-Reduce.
50. Yaremenko V. S. An approach for data clustering CURE algorithm implementation using the MapReduce technology / V. S. Yaremenko. // System Analysis and Information Technologies. 19-th International Conference SAIT 2017. – 2017. – №19. – P. 204.
51. Yaremenko V. S. Distributed data clustering CURE algorithm approbation using Hadoop MapReduce / V. S. Yaremenko. // International Scientific Journal "Internauka". – 2017. – №6. – P. 81–83.
52. Zhao J. MapReduce: The programming model and practice / J. Zhao, J. Pjesivac-Grbovic., 2009. – 79 с. – (SIGMETRICS (2009)).
53. Zhao W. Parallel K-Means Clustering Based on MapReduce / W. Zhao, H. Ma, Q. He. // CloudCom 2009, LNCS 5931. – 2009. – С. 674–679.