НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
ННК "Інститут прикладного системного аналізу"
(повна назва інституту/факультету)

Системного проектування
(повна назва кафедри)

«На правах рукопису»
УДК _____

«До захисту допущено»
Завідувач кафедри
_____
_____
(підпис) (ініціали, прізвище)
"___"_____20__
р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 8.05010103 системне проектування
(код і назва)

на тему: Розподілене обчислення генетичних алгоритмів за допомогою системи Hadoop.

Виконав: студент _VI_ курсу, групи __ДА-52м__
(шифр групи)

Качко Микита Андрійович _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник к. т. н., доцент Харченко Костянтин Васильович
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, 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»

Інститут (факультет) ННК “Інститут прикладного системного аналізу”
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – другий (магістерський)

Спеціальність 8.05010103 системне проектування
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ _____
(підпис) (ініціали, прізвище)
«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Качко Микиті Андрійовичу
(прізвище, ім’я, по батькові)

1. Тема дисертації Розподілене обчислення генетичних алгоритмів за допомогою системи Hadoop,
науковий керівник дисертації Харченко К.В к.т.н., доц.,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом дисертації _____

3. Об’єкт дослідження: Hadoop

4. Предмет дослідження: прогнозування шляхом екстраполяції та генетичними алгоритмами за допомогою системи розподілених обчислень

5. Перелік завдань, які потрібно розробити

- Розглянути існуючі системи розподілених обчислень.
- Дослідити математичні основи роботи.
- Розробити опис задач для тестування.
- Протестувати програмний додаток та порівняти дві концепції вирішення поставленої задачі.
- Зробити висновки щодо перспективності використання фреймворку HADOOP в прикладних програмних додатках.

6. Орієнтовний перелік ілюстративного матеріалу: презентація на тему Розподілене обчислення генетичних алгоритмів за допомогою системи Hadoop

7. Орієнтовний перелік публікацій

Kachko N. A. Genetic algorithms over distributed systems with MapReduce model / Kachko. // Institute for Applied System Analysis at the Igor Sikorsky Kyiv Polytechnic Institute. – 2017. – С. 199–200.
Kachko N. Prime table generation / Kachko N.: міжнародна науково-технічна конференція «САІТ-2015», 23 червня 2015, Київ, Україна : матеріали. – К. : НТУУ «КПІ», 2015.
Yaremenko V., Kachko N. Primality test problem / Yaremenko V., Kachko N. : Матеріали XIV всеукраїнської науково – практичної студентської конференції, 07 квітня 2015, Київ, Україна : матеріали. – К. : НТУУ «КПІ», 2015. – С. 136.

8. Консультанти розділів дисертації*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	30.09.2016	
2	Збір інформації	21.01.2017	
3	Розробка алгоритму та структури програми	10.03.2017	
4	Розробка плану оцінювання	05.04.2017	
5	Розробка програмної моделі	10.05.2017	
6	Тестування додатку та отримання даних	30.05.2017	
7	Отримання допуску до захисту та подача роботи в ДЕК	12.06.2017	

Студент _____ М. А. Качко
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ К. В. Харченко
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено наукового керівника магістерської дисертації.

РЕФЕРАТ

на магістерську дисертацію

виконану на тему: Розподілене обчислення генетичних алгоритмів за допомогою системи Hadoop

студентом: Качко Микитою Андрійовичем

Робота виконана на 101 сторінці, містить 32 ілюстрації, 41 таблицю. При підготовці використовувалася література з 52 різних джерел.

Актуальність

Технологія розподілення обчислень з кожним днем стає все більш популярною і все частіше використовується в різних областях. Ця технологія має великий потенціал і тому вона активно розвивається.

Актуальність роботи полягає в тому що розподіл обчислень на кілька комп’ютерів зменшить час і зусилля, необхідні для виконання завдання.

Мета

Метою цієї дипломної роботи є дослідження і розробка нового методу прогнозу цін на біржі нафтопродуктів. Для виконання прогнозу необхідно використовувати генетичні алгоритми як спосіб отримання моделі поведінки прогнозованої величини. Також необхідно запропонувати спосіб задання навантажень у розподіленому середовищі для пришвидшення обчислень.

Завдання

Для досягнення поставленої мети необхідне рішення наступних завдань:

- огляд існуючих систем розподілених обчислень.
- дослідження математичних основ роботи.
- розроблення опису задач для тестування.
- тестування програмного додатку та порівняння двох концепцій вирішення поставленої задачі.

Об’єкт досліджень

У відповідності до поставленої мети об'єктом досліджень обрано систему розподілених обчислень Hadoop.

Предмет досліджень

Прогнозування цін на нафту шляхом екстраполяції та генетичними алгоритмами за допомогою системи розподілених обчислень.

Наукова новизна

Наукова новизна роботи полягає в інноваційному алгоритмі розподілення та корегування даних для роботи з генетичними алгоритмами.

Практична цінність

В якості результату розроблено програмне забезпечення з підтримкою платформи Hadoop. Для порівняння було розроблено програмне забезпечення але з іншою, прямою структурою. Результат даної роботи можна використовувати в подальшому для прогнозу інших подій із фрактальною поведінкою.

Ключові слова

Розподіл обчислень, алгоритм, Hadoop, генетичний алгоритм, екстраполяція, поліном, фрактали.

Реферат

на магистерскую диссертацию

выполненную на тему: Распределенный вычисления генетических алгоритмов с помощью системы Hadoop

студентом: Качко Никитой Андреевичем

Работа выполнена на 101 странице, содержит 32 иллюстрации, 41 таблицу. При подготовке использовалась литература из 52 различных источников.

Актуальность

Технология распределения вычислений с каждым днем становится все более популярной и все чаще используется в различных областях. Эта технология имеет большой потенциал и поэтому она активно развивается.

Актуальность работы заключается в том, что распределение вычислений на несколько компьютеров уменьшит время и усилия, необходимые для выполнения задания.

Цель

Целью настоящей дипломной работы является исследование и разработка нового метода прогноза цен на бирже нефтепродуктов. Для выполнения прогноза необходимо использовать генетические алгоритмы как способ получения модели поведения прогнозируемой величины. Также необходимо предложить способ задания нагрузок в распределенной среде для ускорения вычислений.

Задача

Для достижения поставленной цели необходимо решение следующих задач:

- обзор существующих систем распределенных вычислений.
- исследования математических основ работы.
- разработка описания задач для тестирования.

- тестирование программного приложения и сравнение двух концепций решения поставленной задачи.

Объект исследований

В соответствии с поставленной целью объектом исследований выбрана система распределенных вычислений Hadoop.

Предмет исследований

Прогнозирование цен на нефть путем экстраполяции и генетическими алгоритмами с помощью системы распределенных вычислений.

Научная новизна

Научная новизна работы заключается в инновационном алгоритме распределения и корректировки данных для работы с генетическими алгоритмами.

Практическая ценность

В качестве результата разработано программное обеспечение с поддержкой платформы Hadoop. Для сравнения было разработано программное обеспечение но с другой, прямой структурой. Результат данной работы можно использовать в дальнейшем для прогноза других событий с фрактальной поведением.

Ключевые слова

Распределение вычислений, алгоритм, Hadoop, генетический алгоритм, экстраполяция, поленом, фракталы.

ABSTRACT

on the master's thesis

on topic: Distributed computing of genetic algorithms with HADOOP

student: Kachko Nikita

Work carried out on 101 pages, containing 32 figures, 41 tables. The paper was written with references to 52 different sources.

Topicality

Distributed computing technology with each passing day becomes more and more popular and are increasingly used in various fields. This technology has great potential and is therefore actively developing.

The significance of the work is that the distribution of computation on several computers will reduce the time and effort required to complete the task.

Purpose

The purpose of this work is to research and develop a new method of forecasting the price of oil exchange. To perform forecasting there is need to use genetic algorithms as a way of getting a behavior of predicted value. There is also need to offer a way of setting loads in a distributed environment to improve computation performance.

Solution

To achieve this goal, the following tasks are required:

- review of existing distributed computing systems.
- mathematical foundations of research.
- describe the development problems for testing.
- application software testing and comparison of the two concepts solve this problem.

The object of research

According to the research goal the object of study has been selected Hadoop distributed computing system.

Subject of research

Forecasting oil prices by extrapolation and genetic algorithms using distributed computing system.

Scientific novelty

Scientific novelty lies in the innovative algorithm of distribution and correction of data used for genetic algorithms.

The practical value of research

As a result the software was developed with the support of the platform Hadoop. For comparison, we developed other software but on the other, the straightforward structure. The result of this work can be used in the future for other events prediction of fractal behavior.

Keywords

Distributed computing, algorithm, Hadoop, genetic algorithm, extrapolation, polynom, fractals.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних

ІТ – інформаційні технології

ОС – операційна система

ПП – програмний продукт

СКІТ – суперкомп’ютерний комплекс інституту кібернетики

СЛАР – система алгебраїчних лінійних рівнянь

ACID – atomicity, consistency, isolation, durability

BOINC – berkeley open infrastructure for network computing

GPU – graphics processing unit

HDFS – hadoop distributed file system

VDS – virtual disk service

SMP – shared-memory multiprocessing

ВСТУП

Розподі́лені обчи́слення (розподілена обробка даних) — спосіб розв'язання трудомістких обчислювальних завдань з використанням двох і більше комп'ютерів, об'єднаних в мережу.

Розподілені обчислення є окремим випадком паралельних обчислень, тобто одночасного розв'язання різних частин одного обчислювального завдання декількома процесорами одного або кількох комп'ютерів. Тому необхідно, щоб завдання, що розв'язується було сегментоване — розділене на підзадачі, що можуть обчислюватися паралельно. При цьому для розподілених обчислень доводиться також враховувати можливу відмінність в обчислювальних ресурсах, які будуть доступні для розрахунку різних підзадач. Проте, не кожне завдання можна «розпаралелити» і прискорити його розв'язання за допомогою розподілених обчислень.

Робота складається з п’яти розділів:

У першому розділі наведено аналіз існуючих систем для вирішення заданої проблеми, описано про особливості та проблематики задачі розподілення обчислень, сформульована постановка і актуальність даної задачі.

Другий розділ присвячено дослідженню існуючих методів для розв’язання проблеми, побудовано математичну модель, описано за якими критеріями визначалась якість результату роботи системи, наведено алгоритм роботи програми для тестуваня даної системи.

У третьому розділі було обґрунтовано вибір платформи та мови реалізації програмного продукту, було зроблено аналіз архітектури системи. Крім того, були розроблені експерименти за допомогою яких можна було проводити аналіз результатів, отриманих в роботі.

У четвертому розділі проводиться аналіз отриманих результатів, а саме порівняння двох концепцій програмних додатків – з та без підтримання фреймворку HADOOP.

П’ятий розділ присвячено охороні праці. У ньому проведено аналіз умов праці, аналіз шкідливих та небезпечних чинників, з якими стикається робітник під час роботи. В результаті було описано, що потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і було комфортним для робітника.

1 ДОСЛІДЖЕННЯ ЗАДАЧІ РОЗПОДІЛЕННЯ ОБЧИСЛЕНЬ

1.1 Актуальність задачі

Проекти розподілених обчислень (Distributed Computing) мають вагому історію та серйозний обчислювальний потенціал. Сьогодні власникам персональних комп'ютерів пропонують взяти участь у рішенні найрізноманітніших науково-дослідних задач. Найчастіше до технології розподілених обчислень дослідники звертаються тоді, коли наукові проекти вимагають величезної кількості обчислень, які недоцільно та надто дорого виконувати за допомогою суперкомп'ютерів або кластерних обчислювальних мереж. Особливисть таких проектів полягає в тому що вони вирішують проблему одночасного моделювання громіздких систем.

Приклад: моделювання фолдінга (згортання в тривимірну структуру) важливих білків людського організму. Це завдання становить найбільш складну проблему молекулярної динаміки білків, тому що вирішальна стадія фолдінга білка відбувається практично миттєво при взаємному впливі безлічі факторів. Вирішення подібного завдання, у вигляді відпрацьовування методів моделювання на експериментально підтверджених моделях згортання білків, відкриває перспективи дослідження першопричин порушень цього процесу, що призводять до багатьох важких захворювань. Ці дослідження зрештою повинні призвести до розробки нових ліків і способів лікування багатьох форм рака, хвороб Альцгеймера, Паркінсона та ін. [Сайт української команди розподілених обчислень 2015]

Приклад 2: при аналізі сучасних складних математичних моделей, що описуються системами диференційних рівнянь з частинними похідними виникає потреба в значних обчислювальних ресурсах для роботи з розрідженими системами лінійних алгебраїчних рівнянь (СЛАР). Зокрема, вирішення задач теплового проектування електронних пристроїв складної

конструкції методами скінченних елементів чи скінченних різниць зводиться до розв’язування СЛАР з розрідженими матрицями надвеликих розмірностей.

Розмірність таких систем в окремих випадках є настільки великою, що їх розв’язання ускладнюється або й стає неможливим без використання спеціальних обчислювальних ресурсів та засобів, таких як обчислювальні кластери або розподілені системи. Саме системи розподілення обчислень стали останнім часом особливо використовуваними. Така популярність обумовлена, в першу чергу:

- зростанням потреб у вирішенні обчислювальних завдань високої складності;
- стрімким підвищення швидкодії систем розподілення обчислень;
- на порядок нижчою собівартістю таких систем в порівнянні з суперкомп’ютерами.

Розробка методів та засобів розподілення обчислень, що виконуються при розв’язуванні СЛАР великої розмірності дасть змогу, завдяки ефективному використанню наявної обчислювальної інфраструктури: по-перше, зменшити час вирішення задач теплового проектування електронних пристроїв та, по-друге, скоротити обсяг необхідних фінансових ресурсів.

Таким чином, розробка методів та засобів, які дали б можливість розподіленого вирішення задач теплового проектування електронних пристроїв, що потребують розв’язування СЛАР великої розмірності є актуальною науковою задачею. [Семчишин 2010, с. 24]

Ще одна проблема: практично всім науковим організаціям недостатньо фінансування для придбання спеціальних ресурсів - суперкомп'ютерів або кластерних обчислювальних мереж - для проведення ґрунтовних наукових досліджень. Розподілені обчислення, що проводяться за допомогою групи добровільних користувачів Internet, співставні і навіть перевершують потужності суперкомп'ютерів і тому можуть стати альтернативою на окремих

етапах актуальних наукових досліджень. [Розподілені обчислення в Україні 2015]

Розглянемо застосування схем на нашій Батьківщині

Завдяки використанню суперкомп'ютерів науково-дослідні установи НАН України отримали важливі фундаментальні і прикладні результати з біофізики, біохімії, фізичної хімії, теоретичної фізики, матеріалознавства, медицини, геології/геофізики, нанотехнологій тощо. В рамках виконання тематичних планів досліджень, програм держзамовлень, участі у вітчизняних та міжнародних наукових проектах українські науковці постійно застосовують суперкомп'ютерний комплекс СКІТ Інституту кібернетики (ІК) ім.В.М.Глушкова НАН України для розв'язання задач моделювання молекулярної динаміки, квантово-хімічних розрахунків, моделювання теплофізичних, гідродинамічних і геофізичних процесів. На суперкомп'ютерах проводяться розрахунки властивостей хімічних сполук, сплавів, біологічно активних речовин і нанооб'єктів, обробка супутникових зображень і результатів сейсмогеологічних досліджень, медичні популяційні дослідження і дослідження геному, моделювання фізичних і соціально-економічних процесів.

Суперкомп'ютерні обчислення сполучуються грід-обчисленнями, хоча не заміняють одне іншого. Грід – це інфраструктура розподілених обчислень, коли задача поділяється на кілька окремих підзадач, підзадачі розв'язуються незалежно, навіть не одночасно, їх результати записуються в файли, а вже потім об'єднуються. Суперкомп'ютер спрямований на паралельні обчислення, тобто підзадачі виконуються одночасно і взаємодіють через передачу повідомлень з даними. Взагалі Грід може використовувати і використовує суперкомп'ютери як вузли. Але для багатьох задач великої розмірності відсутні алгоритми поділу на незалежні підзадачі, а умови стійкості обмежують розміри елементарних одиниць, отже, і обсяги необхідної пам'яті. Такі задачі взагалі неможливо розв'язати без суперкомп'ютера.

Можливості сучасних суперкомп'ютерів дозволяють створити інформаційні технології, які з позиції системного (комплексного) аналізу та оптимізації розкривають сутність природно взаємопов'язаних досліджуваних явищ. Внаслідок розвитку суперкомп'ютерних технологій вартість і час чисельного моделювання стали нижчими, ніж в разі експериментального випробування реальних об'єктів. Сучасні ІТ та суперкомп'ютери, як інструменти розв'язання класів надскладних задач стали одним з визначальних чинників росту конкурентоспроможності економіки і безпосередньо впливають на спроможність країни забезпечувати потреби населення та суб'єктів господарювання у інформаційному суспільстві.

Створення суперкомп'ютерних ІТ потребує розробки математичних моделей, ефективних методів їх аналізу, алгоритмів та надійних програм. Складність комплексних взаємозв'язків предметної області та зростаючі вимоги до ефективності паралельної реалізації сприяють успіху інтелектуальних ІТ, здатних самостійно налаштовуватись на властивості задачі та обчислювального середовища. Фундаментальна наукова складова цієї тематики полягає в дослідженні і розробці ефективних математичних методів та алгоритмів розв'язання класів задач, що за вимогами до об'єму пам'яті та обчислювальної продуктивності перевищують можливості доступних ЕОМ, аж до задач трансобчислювальної складності.

Подальший прогрес застосування обчислень у фундаментальній та прикладній науковій діяльності НАН України потребує розвитку суперкомп'ютерних потужностей. Кластерний комплекс СКІТ, який є найбільшим обчислювальним ресурсом НАН України та основою ресурсного центру Українського національного Гріду (УНГ), з 2005 р. забезпечує на безоплатній основі високопродуктивні обчислення організацій та установ НАН України, закладів МОН і НКА. Три покоління суперкомп'ютерів СКІТ (СКІТ-1, СКІТ-2, СКІТ-3) розвинуто та успішно застосовано у дослідженнях за програмою «Інтелект» протягом 2007-2009 рр., а за 2010-2011 рр. Впроваджено

гібридний сегмент СКІТ-GPU для експериментальних досліджень нових підходів до розпаралелювання задач. На його основі у 2012 р. розроблено та реалізовано у суперкомп'ютері СКІТ-4 новий комплексний архітектурний проект побудови кластерних обчислювальних систем, що дозволило досягти вдвічі більшої за суперкомп'ютер попереднього покоління СКІТ-3 продуктивності - майже 12 Тфлопс. Крім того, це дозволило суттєво підвищити його енергоефективність, електроспоживання якого (15 кВт-годин) вчетверо менше за СКІТ-3. СКІТ-4 за енергоефективністю відповідає 99 позиції у світовому рейтингу найбільш екологічних суперкомп'ютерів (Green500).

Передбачається подальше нарощування його потужностей. Суперкомп'ютер СКІТ-4 зараз є найпотужнішим обчислювальним засобом України. Він підключений як складова частина до комплексу СКІТ, що є основою Ресурсного центру Українського національного гріду, та пройшов сертифікацію Європейської грід-ініціативи (EGI).

Серед користувачів кластерного комплексу СКІТ найбільшу частку ресурсів у 2010–2012 рр. споживали:

1 – Інститут молекулярної біології і генетики (близько 30%);

2 – Інститут кібернетики ім. В.М. Глушкова (15%);

3 – Український національний грід (за 3 роки його частка зросла з 5,5 до 18%);

4 – Інститут фізики конденсованих систем;

5 – Фізико-технічний інститут низьких температур ім. Б.І. Вєркіна;

6 – Інститут математичних машин і систем;

7 – Інститут гідромеханіки;

8 – Ужгородський національний університет;

9 – Інститут органічної хімії;

10 – Запорізький інститут державного та муніципального управління.

Крім того, суперкомп'ютери СКІТ застосовували Інститут геохімії, мінералогії та рудоутворення ім. М.П. Семененка, Інститут металофізики ім.

Г.В.Курдюмова, Донецький фізико-технічний інститут ім. О.О. Галкіна, Харківський національний університет ім. В.Н. Каразіна. Близько 15% обчислювальних ресурсів використано для розв’язання важливих державних задач (зовнішня розвідка, державний бюджет, оптимізація обслуговування державного боргу). Задачі, які вирішують за допомогою суперкомп’ютерів, часто взагалі неможливо розв’язати в інший спосіб із потрібною точністю чи детальністю. Насамперед це стосується задач моделювання, де критичним параметром є розмір сітки. Наприклад, дослідження довготривалого впливу забруднення на якість питної води неефективно проводити на окремій ділянці з огляду на невизначеність умов на краях моделі, якщо вони не відповідають природним межам системи вододілу поверхневих і підземних потоків. Отже, масштаб такого дослідження – тисячі й десятки тисяч квадратних кілометрів, а глибинність – сотні метрів. Потужність водоносного шару – від кількох метрів, ширина більшості річок не перевищує 10–20 м. Тож йдеться про сітки з мільйонів чи десятків мільйонів комірок. Обрахунок таких сіток потребує на кожному часовому кроці розв’язку системи рівнянь відповідного розміру (кілька рівнянь на комірку). Використання суперкомп’ютера дає змогу кардинально підвищити детальність моделі, а отже, і достовірність результатів. [Богданов 2012, с. 7]

1.2 Особливості й проблематика задачі розподілення обчислень

Проаналізувавши ринок хмарних обчислень, можна сказати, що основна його маса зосереджена на продажу віртуальних машин в хмарі. Однак, хмари, це не тільки маркетинг. Існуючі технології тимчасового збільшення ліміту споживання ресурсу не здатні вирішити проблему динамічної зміни ресурсів. Тобто машина недоотримує ресурси тоді, коли вони їй найбільше потрібні. Друга проблема - коли ресурси, фактично, довелося сплатити, але

використовувати не вдалося. Віртуальній машині вночі просто не потрібно стільки ресурсів. Вона простоює.

Виникає проблема: людина змушена замовляти ресурсів більше, ніж потрібно в середньому, для того, щоб переживати без проблем піки. В інший час ресурси простоюють. Провайдер бачить, що сервер не навантажений, починає продавати ресурсів більше, ніж є (це називають «оверселл»). У якийсь момент, наприклад, через пік навантаження на декількох клієнтів, провайдер порушує свої зобов'язання. Він обіцяв 70 людям по 1 ГГц - але в нього є тільки 40 (2.5 * 16 ядер). Недобре. Продавати смугу чесно (без оверселла) невигідно (і ціни неринкові виходять). Оверселлити - знижувати якість сервісу, порушувати умови договору. Ця проблема не пов'язана з VDS або віртуалізацією, це загальне питання: як чесно продавати простоюють ресурси? [Кобець 2012, с. 28]

Під час розподіленної обробки даних велика задача поділяється на кілька менших, які одночасно виконуються на кількох вузлах комп'ютерної системи. Відтак вихідна задача вирішується значно швидше. Розподіл даних системи за багатьма вузлах здійснюється для того, щоб дати можливість багатьом обчислювальним одиницям отримувати одночасний доступ до даних, розміщених на різних носіях. Цим забезпечується паралелізм під час виконання операцій обчислення/введення/виведення. Відомі два варіанти розміщення даних на багатьох елементах:

- розміщення на різних вузлах таблиць;
- розміщення на різних вузлах рядків однієї таблиці, або горизонтальний поділ таблиць.

Разом із цим постає проблема нерівномірного розподілу даних. Бажано, щоб кортежі відношення розподілялися за вузлами рівномірно, оскільки саме в такий спосіб досягатиметься рівномірний розподіл навантаження на елементи під час виконання операцій обчислення/введення/виведення. А це, в свою чергу, сприяє ефективнішому розподіленню. Однак можливі ситуації, коли одні

елементи містять багато даних, а інші — мало. Такий розподіл, звичайно, є небажаним. Нерівномірний розподіл може бути обумовлений такими факторами:

- нерівномірний розподіл значень атрибутів. Значення деяких атрибутів у відношенні можуть розподілятися нерівномірно. Наприклад, атрибут Зарплата взагалі не має значень поза межами певного діапазону, а всередині нього, скоріш за все, значення розподілені нерівномірно. Якщо такий атрибут використовується в хешуванні чи ранжуванні, то кортежі відношення будуть розподілятися за вузлами нерівномірно;
- нерівномірний розподіл значень у групах ранжування. Застосування методу ранжування за неправильного формування груп може призвести до нерівномірного розподілу кортежів. [Паралельні бази даних 2016]

1.3 Приклад існуючих систем для вирішення задачі

На даний момент часу, ринок програмного забезпечення систем розподілених обчислень представлений великою кількістю різних програмних продуктів. Розглянемо деякі з них:

1.3.1 Folding@Home

Folding@Home – це проект розподілених обчислень для проведення комп’ютерного моделювання згортання молекул білка. Проект запущений 1 жовтня 2000 року ученими зі Стенфордського університету.

Мета проекту – за допомогою моделювання процесів згортання/розгортання молекул білка отримати краще розуміння причин виникнення хвороб, що викликаються дефектними білками, таких як хвороби Альцгеймера, Паркінсона, діабету II типу, хвороби Крейтцфельдта - Якоба (коров’ячий сказ), склероз і різних форм онкологічних захворювань. До

теперішнього часу проект Folding@home успішно змоделював процес згортання білкових молекул протягом 5-10 мкс – що в тисячі разів більше попередніх спроб моделювання.

За результатами експерименту опубліковано трохи менше 200 наукових робіт.

1.3.2 Rosetta@Home

Проект добровільних обчислень, спрямований на вирішення однієї з найбільших проблем в молекулярній біології – обчислення третинної структури білків з їх амінокислотних послідовностей. Завдяки недавно завершеному проекту «Геном людини» відомі амінокислотні послідовності всіх білків в людському організмі. Дослідження за даним проектом також допоможуть у проектуванні нових, неіснуючих білків. В разі успішного вирішення даних проблем людство зможе боротися з такими хворобами як рак, малярія, хвороба Альцгеймера, сибірська виразка та іншими генетичними і вірусними захворюваннями.

Результати обчислень Rosetta@Home не доступні безпосередньо. Так само, не можна використовувати результати обчислень власного комп'ютера. Однак вони використовуються для великої кількості наукових публікацій.

По суті Rosetta – це комп’ютерна програма, основними завданнями якої є:

- пошук структури з найменшою енергією для заданої амінокислотної послідовності для передбачення структури білка;
- вирішення оберненої задачі – пошук амінокислотної послідовності з найменшою енергією для заданої білкової структури;
- розрахунок взаємодії комплексу білок-білок.

В даному проекті використовується зворотній зв'язок з прогнозуванням та отриманим результатом, щоб покращувати потенційні функції і алгоритми пошуку. [Розподілені Обчислення 2013]

1.3.3 Einstein@Home

Einstein @ Home — проект добровольчих обчислень на платформі BOINC по перевірці гіпотези Ейнштейна про існування гравітаційних хвиль, які досі нікому з учених так і не вдалося зафіксувати.

Проект стартував в рамках Всесвітнього року фізики 2005 (англ.) і координується університетом Вісконсіна-Мілуокі (англ.) (Мілуокі, США) і Інститутом гравітаційної фізики ім. Макса Планка (англ.) (Ганновер, Німеччина), керівник — Брюс Аллен (англ.).

З метою перевірки гіпотези проводиться складання атласу гравітаційних хвиль, випромінюваних неосесиметричними зірками, що швидко обертаються: нейтронними зірками (пульсарами), хитними зірками (англ. wobbling star), аккреціюючими (англ. accreting star) і пульсуючими зірками (англ. oscillating star).[What is the purpose of Einstein@Home? 2015] Дані для аналізу надходять з Лазерно-інтерферометричної гравітаційно-хвильової обсерваторії (LIGO) і GEO600.

Крім перевірки загальної теорії відносності Ейнштейна та отримання відповідей на питання «Чи поширюються гравітаційні хвилі з швидкістю світла?» і «чим вони відрізняються від електромагнітних хвиль?» [Пользователей интернета пригласили подтвердить теорию Эйнштейна 2005], пряме виявлення гравітаційних хвиль буде також являти собою важливий новий астрономічний інструмент (більшість нейтронних зірок не випромінюють в електромагнітному діапазоні та гравітаційні детектори здатні привести до відкриття цілої серії раніше невідомих нейтронних зірок[Holger J. et all 2010, с. 5]). Наявність же експериментальних доказів відсутності гравітаційних хвиль відомої амплітуди від відомих джерел поставить під сумнів саму загальну теорію відносності та розуміння сутності гравітації.

1.3.4 PrimeGrid

PrimeGrid — проект добровільних розподілених обчислень на платформі BOINC і PRPNet, метою якого є пошук простих чисел різного виду. Проект стартував 12 червня 2005 року.

Основна мета проекту — нести задоволення пересічному користувачу від знаходження простих чисел. Другою метою PrimeGrid є надання відповідних навчальних матеріалів про прості числа.

І нарешті, прості числа грають центральну роль в криптографічних систем, які використовуються для комп'ютерної безпеки. Через вивчення простих чисел можна показати, скільки вимагається обчислень, щоб зламати код шифрування і таким чином визначити, чи є поточні схеми безпеки достатньо безпечними.

Участь у проекті PrimeGrid можлива у два способи: через BOINC і через Project Staging Area (PSA) [Офіційний сайт PrimeGrid 2014]

1.4 Постановка задачі

Метою даної роботи було аналіз продуктивності програмного забезпечення HADOOP. Для реалізації цього завдання потрібно:

1. Дослідити та вивчити основні принципи роботи програм для розподілення обчислень.
2. Розробити прикладну задачу, на прикладі якої можна ефективно дослідити завдання розподілених обчислень і яка є гнучкою для реалізації на різних мовах програмування та різних операційних системах.
3. Зробити критичний огляд відомих програм для побудови паралельних систем.
4. Проаналізувати існуючі платформи та мови для реалізації даного програмного продукту та вибрати ті, що найкраще підходять для реалізації.

5. Розробити вхідний набір даних для оцінки системи.
6. Зробити висновки щодо отриманих результатів.

1.5 Висновки

В даному розділі розглянуто та досліджено задачу побудови системи розподіленх обчисдень. В ході дослідження було описано особливості та основну проблематику при реалізації даної задачі, розглянуто існуючі системи для вирішення проблеми.

Враховуючи предмет дослідження та специфіку даного завдання, було сформульовано постановку задачі, де описано основні проблеми, які треба вирішити для створення програмного продукту за допомогою якого можна дослідити ефективність застосування системи розподілених обчислень HADOOP

2 МАТЕМАТИЧНІ ОСНОВИ РОБОТИ

2.1 Дослідження області генетичних алгоритмів

Генети́чний алгори́тм — це еволюційний алгоритм пошуку, що використовується для вирішення задач оптимізації і моделювання шляхом послідовного підбору, комбінування і варіації шуканих параметрів з використанням механізмів, що нагадують біологічну еволюцію.

Особливістю генетичного алгоритму є акцент на використання оператора «схрещення», який виконує операцію рекомбінацію рішень-кандидатів, роль якої аналогічна ролі схрещення в живій природі. «Батьком-засновником» генетичних алгоритмів вважається Джон Голланд (англ. John Holland), книга якого «Адаптація в природних і штучних системах» (англ. Adaptation in Natural and Artificial Systems) є фундаментальною в цій сфері досліджень.

2.1.1 Опис

Задача кодується таким чином, щоб її вирішення могло бути представлено в вигляді масиву подібного до інформації складу хромосоми. Цей масив часто називають саме так «хромосома». Випадковим чином в масиві створюється деяка кількість початкових елементів «осіб», або початкова популяція. Особи оцінюються з використанням функції пристосування, в результаті якої кожній особі присвоюється певне значення пристосованості, яке визначає можливість виживання особи. Після цього з використанням отриманих значень пристосованості вибираються особи, допущені до схрещення (селекція). До осіб застосовується «генетичні оператори» (в більшості випадків це оператор схрещення (crossover) і оператор мутації (mutation)), створюючи таким чином наступне покоління осіб. Особи

наступного покоління також оцінюються застосуванням генетичних операторів і виконується селекція і мутація. Так моделюється еволюційний процес, що продовжується декілька життєвих циклів (поколінь), поки не буде виконано критерій зупинки алгоритму. Таким критерієм може бути [Poli 2008, с. 89]:

- знаходження глобального, або надоптимального вирішення;
- вичерпання числа поколінь, що відпущені на еволюцію;
- вичерпання часу, відпущеного на еволюцію.

Генетичні алгоритми можуть використати для пошуку рішень в дуже великих і важких просторах пошуку. В свою чергу різновиди та особливості алгоритмів можуть бути змінені в залежності від галузі:

- У комбінаторній хімії — метод дизайну бібліотеки шляхом оцінки відповідності певних бажаних властивостей (пр., рівня активності в біологічних пошуках, або розрахунково визначених властивостей набору речовин), передбачених за допомогою функції, встановленої статистичними методами при аналізі співвідношення структура — властивість. Ще більш оптимальний дизайн пов'язаний з евристичним процесом, який нагадує генетичну селекцію, де застосовується реплікація, мутація, вилучення.
- У хемометриці — механізм оптимізації, заснований на механізмі дарвінівської еволюції, де використовуються випадкові мутації, процедури схрещення та відбору для розробки кращої моделі чи розв'язку порівняно з тим, які було отримано, виходячи зі стартової сукупності чи вибірки.
- У комп'ютерній хімії — комп'ютерний метод генерування та тестування комбінацій можливих вхідних параметрів для знаходження оптимальних вихідних значень. Використовується для оптимізації у випадку систем з великою кількістю змінних параметрів, зокрема при конформаційному аналізі багатоатомних

складних молекул. Включає методи, що базуються на поняттях природної еволюції, такі як генетична комбінація, мутація та природний відбір.

2.1.2 Етапи генетичного алгоритму

На початку треба насамперед створити початкову популяцію. Це робиться для того щоб сгенерувати базу із якою можна працювати у подальшому. Разом із генерацією нових осіб популяції обчислюється функція пристосованності кожного. Далі необхідно провести вибір індивідів із поточної популяції (селекція), схрестити обрані, та мутувати деякі з них. Для всіх новоотриманих осіб слід потім обрахувати функцію пристосовуваності та сформувати нове покоління. Алгоритм потім повторюється (зміна поколінь) до тих пір поки не буде виконаний критерій зупинки алгоритму.

2.1.2.1 Створення початкової популяції

Перед першим кроком необхідно випадковим чином створити деяку початкову популяцію. Навіть якщо популяція виявиться абсолютно неконкурентоздатною, генетичний алгоритм все одно достатньо швидко переведе її в придатну для життя популяцію. Таким чином, на першому кроці можна не старатися зробити надто пристосованих осіб, достатньо, щоб вони відповідали формату осіб популяції, і на них можна було порахувати функцію пристосованості (фітнес функцію). Наслідком першого кроку є популяція H, що налічує N осіб.

2.1.2.2 Відбір

На етапі відбору необхідно із всієї популяції вибрати її певну долю, яка залишиться в «живих» на цьому етапі популяції. Є декілька способів провести відбір. Ймовірність виживання особи h повинна залежати від значення її пристосованості Fitness(h). Сама ж доля відібраних s зазвичай є параметром генетичного алгоритму, і її просто задають заздалегідь. Внаслідок відбору із N

осіб популяції H повинні залишитись sN осіб, які ввійдуть в наступну популяцію H'. Решта осіб «загине».

2.1.2.3 Розмноження

Розмноження в генетичних алгоритмах зазвичай статеве — щоб «народити» нащадка, необхідно декілька батьків, зазвичай потрібна участь двох. Розмноження в різних алгоритмах описується по різному — воно, звісно, залежить від формату осіб. Головна вимога до розмноження — щоб нащадок чи нащадки мали можливість успадкувати риси всіх батьків, «змішавши» їх якимось достатньо розумним чином.

Розмноження або оператор рекомбінації застосовують відразу ж після оператора відбору батьків для отримання нових особин-нащадків. Сенс рекомбінації полягає в тому, що створені нащадки повинні наслідувати генну інформацію від обох батьків. Розрізняють дискретну рекомбінацію і кросинговер.

Приклад операції розмноження: Вибрати (1-s)p/2 пар гіпотез із H і провести з ними розмноження, отримавши по два нащадка від кожної пари (якщо розмноження описано так, щоб давати одного нащадка, необхідно вибрати (1 — s)p пар), і додати цих нащадків в H'. В результаті H' буде складатися з N осіб.

Особи для розмноження зазвичай вибираються із всієї популяції H, а не із тих, що вижили на першому кроці (хоча останній варіант теж має право на існування). Справа в тому, що головна проблема генетичних алгоритмів — недостача різноманітності (diversity) в особах. Достатньо швидко виділяється єдиний генотип, який являє собою локальний максимум і згодом всі елементи популяції програють йому в відборі, і вся популяція «забивається» копіями цієї особи. Існують різні способи боротьби із таким небажаним ефектом; один з них — вибір для розмноження не з самих «пристосованих», а взагалі зі всіх осіб.

2.1.2.4 Мутації

До мутацій відноситься все те ж, що і до розмноження: є деяка доля мутантів m, що є параметром генетичного алгоритму, і на кроці мутацій необхідно вибрати mN осіб, а згодом змінити їх згідно з заздалегідь заданими операціями мутації.

2.1.3 Застосування генетичних алгоритмів

Генетичні алгоритми в різних формах застосовуються до вирішення багатьох наукових і технічних проблем. Генетичні алгоритми використовуються при створенні інших обчислювальних структур, наприклад, автоматів або мереж сортування. У машинному навчанні вони використовуються при проектуванні нейронних мереж або керуванні роботами. Вони також застосовуються при моделюванні розвитку в різних предметних областях, включаючи біологічні (екологія, імунологія і популяційна генетика) та соціальні (такі як економіка і політичні системи) системи.

Генетичні алгоритми застосовуються для вирішення наступних завдань [Lakhmi et all 2012, с. 272]:

- Оптимізація функцій
- Оптимізація запитів в базах даних
- Різноманітні задачі на графах (задача комівояжера, розфарбування, знаходження паросполучення)
- Налагодження та навчання штучної нейронної мережі
- Завдання компонування
- Складання розкладів
- Ігрові стратегії
- Теорія наближень
- Штучне життя
- Біоінформатика (фолдінг білків)

Проте, можливо найбільш популярне застосування генетичних алгоритмів - оптимізація багатопараметричних функцій. Багато реальних задач можуть бути сформульовані як пошук оптимального значення, де значення - складна функція, що залежить від певних вхідних параметрів. У деяких випадках, потрібно знайти ті значення параметрів, при яких досягається найкраще точне значення функції. В інших випадках, точний оптимум не потрібний - рішенням може вважатися будь-яке значення, краще за певну задану величину. У цьому випадку, генетичні алгоритми - часто найбільш прийнятний метод для пошуку "прийнятних" значень. Сила генетичного алгоритму полягає в його здатності маніпулювати одночасно багатьма параметрами, що використовується в сотнях прикладних програм, включаючи проектування літаків, налаштування параметрів алгоритмів і пошуку стійких станів систем нелінійних диференціальних рівнянь [Організація інтелектуальних обчислень 2015].

2.2 Генетичні алгоритми в контексті апроксимації функцій однієї змінної

Здатності генетичних алгоритмів до прогнозування прямо випливають з їх здатності до узагальнення і виділення схованих залежностей між вхідними і вихідними даними. Після виведення найбільш пристосованого екземпляру популяції можна пророчити майбутнє значення якоїсь послідовності на основі декількох попередніх значень і/або якихось існуючих у даний момент факторів. Слід зазначити, що прогнозування можливе тільки тоді, коли попередні зміни дійсно в якомусь степені визначають майбутні. Наприклад, прогнозування котирувань акцій на основі котирувань за минулий тиждень може виявитися успішним (а може і не виявитися), тоді як прогнозування результатів завтрашньої лотереї на основі даних за останні 50 років майже напевно не дасть ніяких результатів. Задачі прогнозування особливо важливі для практики, зокрема, для фінансових додатків, тому пояснимо способи застосування генетичних алгоритмів у цій області більш докладно.

2.2.1 Апроксимація методом екстраполяції

Екстраполяція – наближення, знаходження за рядом даних значень функції інших її значень, що містяться поза цим рядом. Методи екстраполяції часто схожі з методами інтерполяції.

Найбільш поширеним методом екстраполяції є параболічна екстраполяція, при якій в якості значення $f(x)$ в точці x береться значення многочлена $P_n(x)$ ступеня n, що приймає в $n + 1$ точці x_n задані значення $y_i = f(x_i)$. Для параболічної екстраполяції слід користуватися інтерполяційними формулами.

Метод диффернціальної екстраполяції, запропонований Р. Сторном і К. Прайсом [Price 2016, с. 524-527], служить для знаходження глобального оптимуму недиференційовних, нелінійних, мультимодальних функцій багатьох

змінних і належить до прямих методів оптимізації, тобто в ході його роботи потрібно об- числювати лише значення цільової функції (критерію оптимізації), але не її похідних. Еволюційний процес в алгоритмі диффернціальної екстраполяції організовано таким чином. Спочатку з використанням випадкових чисел генерується деяка множина векторів (покоління популяції), які представляють собою можливі розв'язки задачі оптимізації. Далі на кожній ітерації алгоритму (епосі еволюційного процесу) створюється нове покоління. Для кожного вектора старого покоління, який називається базовим вектором, генерується мутантний вектор з використанням трьох інших випадкових векторів і операцій додавання та віднімання їхніх координат. Над мутантним вектором виконується операція схрещування, в ході якої деякі його координати заміщуються координатами базового вектора. Отриманий після схрещування вектор називається пробним. Якщо він виявляється кращим за базовий вектор (зна- чення цільової функції покращилось), то в новому поколінні базовий вектор замінюється на пробний, у протилежному випадку базовий вектор зберігається в новому поколінні. Таке правило гарантує незмінність розміру популяції в процесі роботи алгоритму. На кожній ітерації для контролю швидкості пошуку оптимального розв'язку визначається кращий вектор покоління. Умовами завершення еволюційного процесу (закінчення алгоритму) можуть бути, наприклад, досягнення задовільного значення критерію оптимізації, вичерпан- ня заданого максимального числа поколінь та ін. У цілому алгоритм диффернціальної екстраполяції представляє собою одну з можливих «неперервних» модифікацій генетичного алгоритму. Водночас він має суттєву особливість, яка багато в чому визначає його властивості. Як джерело шуму при мутації в алгоритмі диффернціальної екстраполяції застосовується не зовнішній генератор випадкових чисел, а «внутрішній», реалізований як різниця між випадково вибраними векторами поточної популяції. Завдяки цьому алгоритм може динамічно моделювати особливості рельєфу функції, що оптимізується, підлаштовуючи під них розподіл

«вбудованого» джерела шуму. Саме цим пояснюється здатність алгоритму швидко проходити складні яри, забезпечуючи ефективність навіть у випадку складного рельєфу.

2.2.2 Визначення області застосування прогнозу

Для практичного прикладу прогнозування візьмемо ціни на котирування акцій нафтової продукції. Фрактальний аналіз є перспективним для застосування у дослідженні динаміки фінансових рядів, та показник Херста може слугувати індикатором, який здатний сигналізувати про появу негативних тенденцій на фінансовому ринку. Підтвердженням цього є апробація методу на даних Національного банку України щодо котирування валют. Чому варто використовувати мультифрактальний аналіз, тому що економічні процеси є складними і описати їх одним показником складно, можна лише сказати про загальну картину та з певною ймовірністю сказати чи буде спад, чи зростання у наступних періодах. За допомогою мультифрактального аналізу детрендованих флуктуацій можна більш повно описати поведінку досліджуваного об'єкту. На думку А. Сіроша та Д. Семьонова використовуючи мультифрактальний аналіз можна виявити глобальні та локальні характеристики часових рядів, що дозволя- тиме прогнозувати та передбачати кризові явища на фінансовому ринку [Black et all 2013, с. 637-659].

2.2.3 Шляхи розв’язання задачі

Отже, було з’ясовано що генетичні алгоритми можуть бути застосовані для апроксимації функції однієї змінної що апроксимує поведінку значень будь-якої фінансової величини що змінюється. В контексті цієї роботи було вирішено апроксимувати послідовний ряд значень:

0	1	2	3	4
y_0	y_1	y_2	y_3	y_4

деякою функцією *F(i, X)*:

$y_i = F(x_0, x_1, \ldots, x_n, i)$.

Якщо брати до уваги те, що вирішення поставленої задачі має мати фрактальний характер, найбільш очевидним рішенням буде використовувати наближення ряду Фур'є для косинусів:

$F(x_0, x_1, \ldots, x_n, i) = x_0 + x_1\cos(2\pi i) + \ldots + x_n\cos(2\pi n i)$.

Враховуючи характер поведінки цін на біржі, прогнозування даних буде задачею із лімітованим часом, тому було вирішено дослідити застосованність іншої апроксимації – ряду Тейлора

$F(x_0, x_1, \ldots, x_n, i) = x_0 + x_1 i + \ldots + x_n i^n$,

оскільки операції з багаточленами є менш обчислювально затратною операцією.

Якщо використовувати коефіцієнти (x_0, x_1, …, x_n) як гени, а суму квадратів відхилення значень $F(x_0, x_1, \ldots, x_n, i)$ від оригинальних як функцію застосованості конкретного індивіду, то отримаємо базис на якому за допомогою генетичних алгоритмів можна побудувати прогнозування значень функції однієї змінної що апроксимує цінову ситуацію на біржі акцій, наприклад нафти.

2.3 Реалізація екстраполяції та корегування передбачуваних значень

Для ліпшого пояснення візьмемо приклад зі статті Genetic algorithms over distributed systems with MapReduce model [Kachko 2017, с. 199-200]. Існує необхідність в розробці алгоритму, який прогнозує значення деякої змінної, маючи попередній набір значень протягом певного періоду часу. Більш того, він повинен бути в змозі прийняти до уваги значення змінної у наступні часи. В якості прикладу можуть бути використані ринкові ціни на нафту або акції. Таким чином, треба досягти дві основні мети: в зв'язку з масивною кількістю попередніх даних він повинен бути в змозі обробляти великі масиви значень, а

через характер торгового ринку він повинен також передбачити ціну перш, ніж вона насправді зміниться, інакше в такому методі немає сенсу.

Для виконання першої вимоги можна застосувати розподілені обчислення. Вони використовують кілька обчислюваних вузлів для того, щоб збільшити загальну продуктивність ціною невеликої затримки через канали зв'язку. MapReduce, зокрема, є реалізацією розподіленого алгоритму, який складається з двох частин. Map виконує фактичну роботу, таку як сортування або фільтрація. Reduce виконує операцію зведення, наприклад, об'єднання всіх елементів в один вихідний сигнал [Wickham et all 2011, с. 1-29].

Для виконання другої вимоги можна застосувати генетичні алгоритми. Було емпірично доведено, що вони показують прийнятну точність в певних умовах. Для того, щоб задовольнити цю потребу існує досить багато методів, які можуть бути використані в якості рішення.

Відомо, що оцінка застосованості є найбільш ресурсоємною задачею і може займати дуже багато часу і, отже, стати вузьким місцем в загальній продуктивності [Nedunchelian 2005]. Тому використання декількох ресурсів вирішить цю проблему. Але постає питання: як двоступеневий алгоритм обчислення можна поєднати з генетичним алгоритмом що має можливість постійно розвиватися і брати до уваги нові порції даних?

Для того щоб вирішити цю проблему потрібно створити новий, модифікований генетичний метод. Для того щоб гарантувати що кожен вузол кластера буде обробляти приблизно рівну кількість даних і виконувати таку ж саму кількість обчислювальної роботи, пропонується наступний алгоритм розподілу, заснований на алгоритмі описаному в статті N. Kachko and V. Yaremenko, “Primality Test Problem” [Yaremenko et all 2015, с. 136]:

1. Кожний вузол розподіленої мережі стартує з створення випадкової початкової популяції, а потім генерує наступні покоління через перестановку, мутацію, кросовер і спеціальні перетворення. Іншими словами, він йде шляхом вироблення прогнозування як звичайний

генетичний алгоритм, створюючи покоління кандидатів і сортуючи їх функції пристосованості. На останньому кроці формується відсортований список з кращими особами. Приклад для 3-х вузлів показано на рис. 2.1.

Node 1	
Candidate	Fitness function result
Candidate 10	0.95
Candidate 11	0.93
Candidate 12	0.91
Candidate 13	0.89
Candidate 14	0.87
…	…

Node 2	
Candidate	Fitness function result
Candidate 20	0.96
Candidate 21	0.93
Candidate 22	0.9
Candidate 23	0.87
Candidate 24	0.84
…	…

Node 3	
Candidate	Fitness function result
Candidate 30	0.94
Candidate 31	0.93
Candidate 32	0.92
Candidate 33	0.91
Candidate 34	0.9
…	…

Рисунок 2.1 - Приклад результату обчислення вузлів

2. Кожен вузол вибирає кілька кращих кандидатів і відправляє їх в кореневий вузол, де той, що більше задовільняє функцію застосованості використовується в якості вихідного, як показано на рис. 2.2.

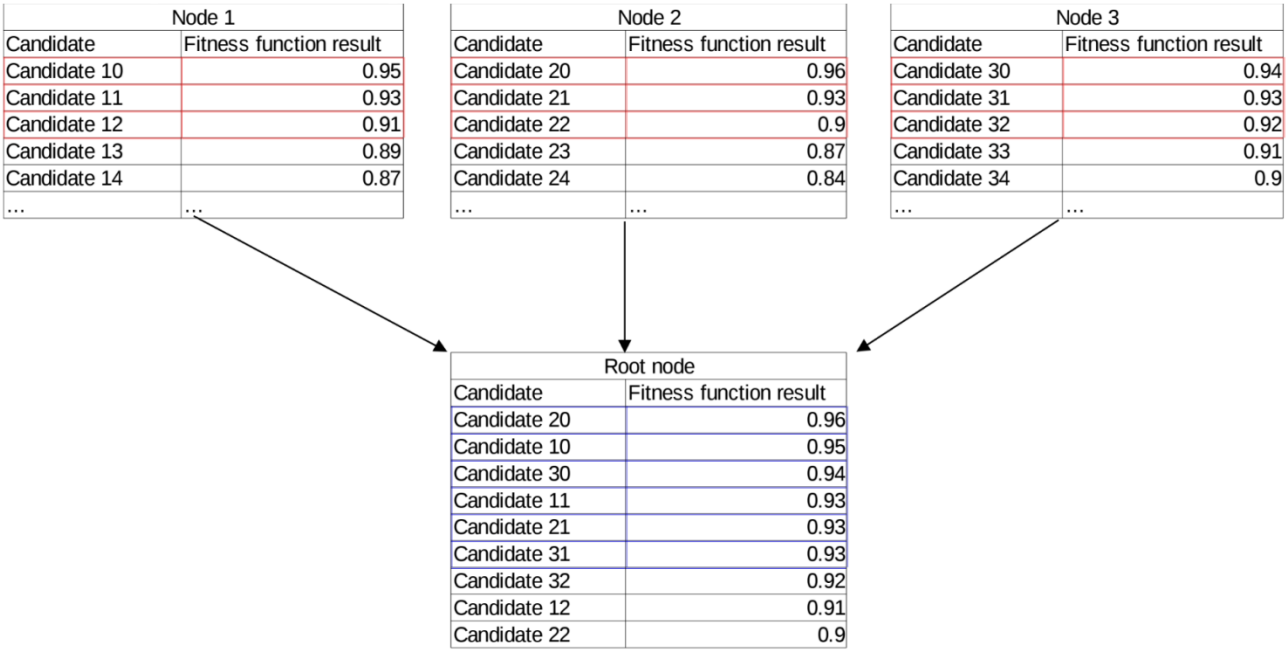

Рисунок 2.2 – Зведення результатів вузлів

3. Оскільки до програми можуть надходити нові фактичні значення змінної, прогнозований шлях вихідного кандидата повинен бути виправлений таким способом, що перший передбачений елемент приймає

точне значення. Приклад з трьома навчальними і трьома передбаченими значенями показаний на рис. 2.3.

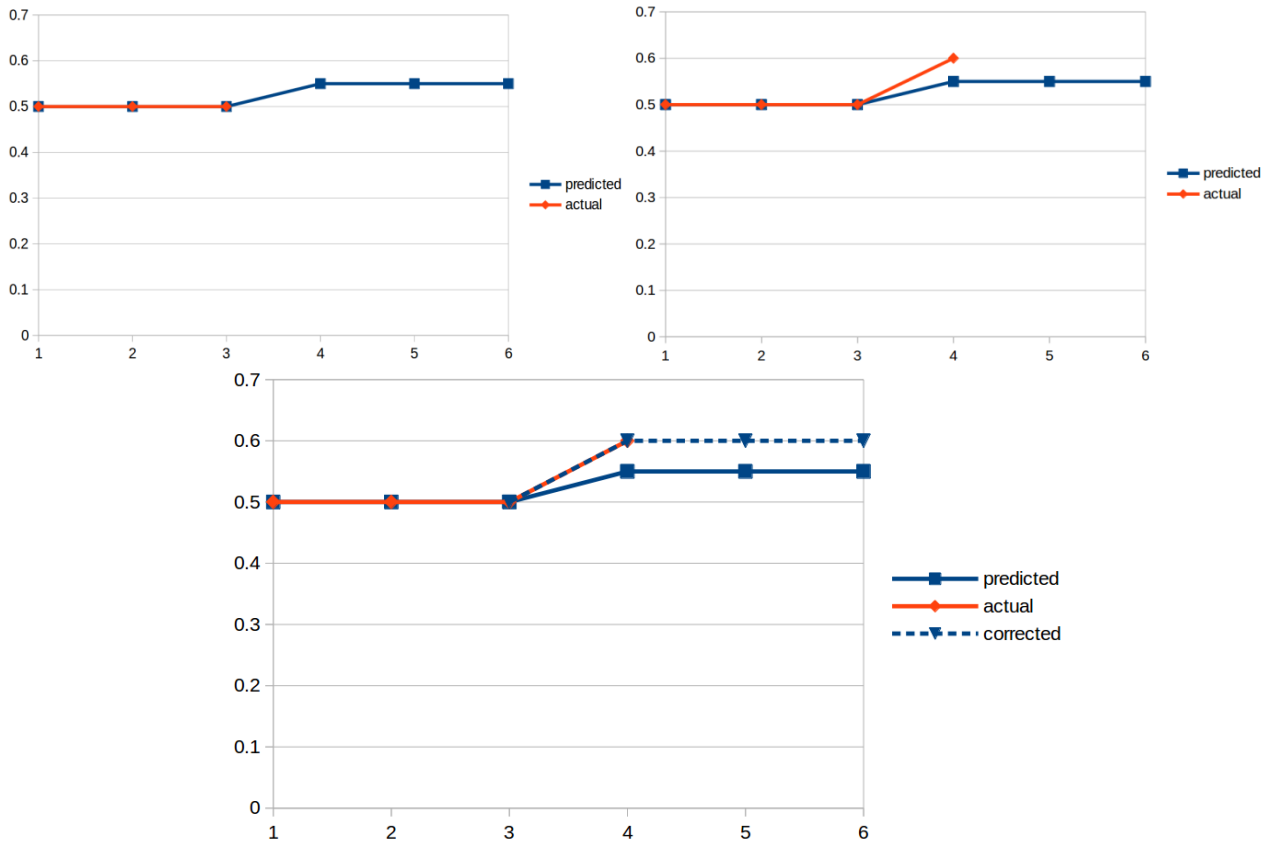

Рисунок 2.3 – Корегування прогнозованого шляху

4. Як приходять нові дані або потрібен більш точний екземпляр (або вихід алгоритму MapReduce), цей алгоритм може бути запущений заново починаючи з першого кроку.

Результатом запропонованого алгоритму є підвищення продуктивності і точності алгоритмів прогнозування,побудованих на генетичних методах, які можуть бути використані в практично необмеженій кількості областях.

2.4 Дослідження принципу розподілення обчислень

За визначенням, розподілені обчи́слення (розподілена обробка даних) — спосіб розв'язання трудомістких обчислювальних завдань з використанням двох і більше комп'ютерів, об'єднаних в мережу.

Розподілені обчислення є окремим випадком паралельних обчислень, тобто одночасного розв'язання різних частин одного обчислювального завдання декількома процесорами одного або кількох комп'ютерів. Тому необхідно, щоб завдання, що розв'язується було сегментоване — розділене на підзадачі, що можуть обчислюватися паралельно. При цьому для розподілених обчислень доводиться також враховувати можливу відмінність в обчислювальних ресурсах, які будуть доступні для розрахунку різних підзадач. Проте, не кожне завдання можна «розпаралелити» і прискорити його розв'язання за допомогою розподілених обчислень [Arora Sanjeev et all 2009, с. 334-340].

Слово «розподілений» в таких термінах, як «розподілена система», «розподілене програмування" та "розподілений алгоритм" спочатку відносилося до комп'ютерних мереж, де окремі комп'ютери були фізично розподілені в деякому географічному регіоні. Ці терміни в даний час використовуються в набагато ширшому сенсі, навіть коли стосуються автономних процесів, які працюють на одному фізичному комп'ютері і взаємодіють один з одним, посилаючи повідомлення. У той час як немає єдиного визначення розподіленої системи, використовуються такі визначальні властивості:

1. Є кілька автономних обчислювальних сутностей, кожна з яких має свою власну локальну пам'ять.
2. Об'єкти взаємодіють один з одним за допомогою передачі повідомлень.

У цій роботі, обчислювальні сутності називаються комп'ютери або вузли.

Розподілена система може мати спільну мету, наприклад, вирішення великої обчислювальної задачі. З іншої сторони, кожен комп'ютер може мати

свого власного користувача зі своїми індивідуальними потребами, і метою розподіленої системи є координація використання загальних ресурсів або надання послуг зв'язку для користувачів.

Інші типові властивості розподілених систем включають в себе наступні:

1. Система повинна бути толерантною до помилок або несправностей в окремих комп'ютерах.
2. Структура системи (топології мережі, затримки в мережі, кількість комп'ютерів) невідома заздалегідь, може складатися з різних видів комп'ютерів і мережевих зв'язків, а також може змінюватися в ході виконання розподіленої програми.
3. Кожен комп'ютер має тільки обмежене неповне уявлення про систему. Кожен комп'ютер може знати тільки одну частину вхідного сигналу.

2.3.1 Історія виникнення терміну

Використання паралельних процесів, що взаємодіють один з одним шляхом передачі повідомлень має свої корені в операційній системі архітектур що вивчались в 1960 році. Перші поширені розподілені системи представляли собою локальні мережі, такі як Ethernet, що був винайдений в 1970-х роках.

Мережа ARPANET була введена в кінці 1960-х років, а електронна пошта від цієї компанії була винайдена на початку 1970-х років. Електронна пошта стала найуспішнішим застосуванням ARPANET, і це, ймовірно, найбільш ранній приклад великомасштабного розподіленого програмного забезпечення. На додаток до ARPANET, і його наступника, Інтернет, іншими ранніми комп'ютерними мережами по всьому світу можна вважати Usenet і FidoNet з 1980-х, кожен з яких використовувався для підтримки розподілених систем зв'язку.

Дослідження розподілених обчислень перетворилось у власну гілку інформатики в кінці 1970-х років і на початку 1980-х років. Перша конференція в області, симпозіум про принципи розподілених обчислень (PODC), відбулася

в 1982 році, і його європейський аналог Міжнародний симпозіум з розподілених обчислень (DISC) вперше був проведений в 1985 році [Andrews et all 2000, с. 25-26].

2.3.2 Теоретичні основи концепції

2.3.2.1 Моделі

Багато задач, які ми хотіли б автоматизувати за допомогою комп'ютера мають тип питання-відповідь: на поставлене запитання комп'ютер повинен видавати відповідь. У теоретичній інформатиці, такі завдання називаються обчислювальні задачі. Формально, обчислювальна задача складається з екземпляру завдання та вирішення кожного з цих завданнь. Екземпляри це питання, які ми можемо задати, а вирішення це відповіді на ці питання.

Теоретична інформатика прагне зрозуміти, які обчислювальні проблеми можна вирішити за допомогою комп'ютера (теорія обчислень) і наскільки ефективно (теорії складності обчислень). За замовчуванням вона свідчить, що проблема може бути вирішена за допомогою комп'ютера, якщо ми можемо розробити алгоритм, який виробляє правильне рішення для будь-якого екземпляра. Такий алгоритм може бути реалізований у вигляді комп'ютерної програми, що виконується на комп'ютері загального призначення: програма зчитує екземпляр проблеми в якості входу, виконує деякі обчислення, і видає рішення в якості виходу.

Області паралельних і розподілених обчислень досліджують схожі питання у випадку як декількох комп'ютерів, так і комп'ютера, який виконує набір взаємодіючих процесів: які обчислювальні завдання можуть бути вирішені в такій мережі і наскільки ефективно? Проте, це зовсім не очевидно, що мається на увазі під поняттям "вирішення проблеми" у випадку паралельних або розподілених систем: наприклад, яке завдання розробника алгоритму і який еквівалент паралельних або розподілених обчислень до послідовного комп'ютеру загального призначення?

Як правило, три точки зору використовуються:

1. Паралельні алгоритми в контексті моделі загальної пам'яті

Всі комп'ютери мають доступ до загальної пам'яті. Розробник алгоритму обирає програму, що буде виконуватись на кожному комп'ютері. Використовується тільки одна теоретична модель - паралельні машини випадкового доступу (PRAM). Проте, класична модель PRAM передбачає синхронний доступ до загальної пам'яті. Програми з загальною пам'яттю можуть бути розширені до розподілених систем, якщо використовувана операційна система інкапсулює зв'язок між вузлами і практично об'єднує пам'ять усіх окремих систем.

2. Паралельні алгоритми в контексті моделі передачі повідомлень

Розробник алгоритму обирає структуру мережі, а також програми, що будуть виконуватися на кожному комп'ютері. Використовуються такі моделі, як булеві схеми і мережа сортування. Булева схема може розглядатися як комп'ютерна мережа: кожний шлюз може бути представлений як комп'ютер, що виконує надзвичайно просту програму. Аналогічно, мережу сортування можна розглядати як комп'ютерну мережу: кожен компаратор може бути представлений як комп'ютер.

3. Розподілені алгоритми в моделі передачі повідомлень

Розробник алгоритму обирає тільки комп'ютерну програму. Всі комп'ютери виконують одну і ту ж програму. Система повинна працювати правильно, незалежно від структури мережі. Зазвичай використовується модель графу із скінченим автоматом представленим у вигляді вузла.

У разі розподілених алгоритмів, обчислювальні завдання, як правило, пов'язані з графіками. Часто граф, що визначає структуру комп'ютерної мережі є екземпляром проблеми.[Andrews et all 2000, с. 87-90]

2.3.2.2 Міра складності

В паралельних алгоритмах, ще один ресурс, на додаток до часу та простору це кількість комп'ютерів. Дійсно, часто треба йти на компроміс між часом та кількостю комп'ютерів: проблема може бути вирішена швидше, якщо є кілька комп'ютерів, що працюють паралельно. Якщо проблема може бути вирішена в полілогаріфмічний час з використанням поліноміальної кількості процесорів, то кажуть, що проблема знаходиться в класі NC. Клас NC може бути визначений однаково добре за допомогою PRAM формалізму або Булевих схем - PRAM машини можуть ефективно імітувати Булеві схеми і навпаки.

При аналізі розподілених алгоритмів, більше уваги зазвичай виділяється на операції зв'язку, ніж обчислювальним крокам. Можливо, найпростіша модель розподілених обчислень є синхронною системою, де всі вузли працюють «в одну ногу». Під час кожного раунду зв'язку, всі вузли паралельно одержують останні повідомлення від сусідів, виконують певні локальний обчислення, і надіслають нові повідомлення для своїх сусідів. У таких системах, центральною мірою складності виступає кількість синхронних раундів зв'язку, необхідних для виконання завдання.

Ця міра складності тісно пов'язана з діаметром мережі. Нехай D діаметр мережі. З одного боку, будь-яка обчислювальна проблема може бути вирішена тривіально в синхронній розподілений системі приблизно за $2D$ раундів зв'язку: зібрати всю інформацію в одному місці (D раундів), вирішити проблему, і повідомити кожен вузол про рішення (D раундів).

З іншого боку, якщо час роботи алгоритму є набагато менше ніж D раундів зв'язку, то вузли мережі повинні обробляти свою задачу, не маючи можливість отримати інформацію про далекі частини мережі. Іншими словами, вузли повинні глобально правильні та стійкі рішення на основі інформації, яка доступна в їх локальній околиці. Відомі багато розподілених алгоритмів з тривалістю набагато менше, ніж D раундів, і розуміння, які проблеми можуть

бути вирішені за допомогою таких алгоритмів є одним з центральних питань дослідження області.

Інші часто вживані засоби виміреня складності - загальна кількість бітів, переданих в мережі [Andrews et all 2000, с. 140-142].

2.3.2.3 Властивості розподілених систем

До даного моменту увага була зосереджена саме на розробці розподіленої системи, яка вирішує проблему. Завдання додаткового дослідження вивчає властивості даної розподіленої системи.

Проблема зупинки - аналогічний приклад з області централізованих розрахунків: ми дали комп'ютерну програму і його задача – вирішити, зупиняється вона чи працює вічно. Проблему зупинки неможливо розв’язати в загальному випадку, і розуміння поведінки комп'ютерної мережі такеж складне, як і розуміння поведінки одного комп'ютера.

Тим не менш, є багато цікавих особливих випадків, які можна розв'язати. Зокрема, можна міркувати про поведінку мережі кінцевих автоматів. Однин із прикладів, це визначення чи дана мережа взаємодіючих (асинхронних та недетермінірованих) кінцевих автоматів може зайти в глухий кут. Ця проблема PSPACE-повна, тобто загалом вирішувана, але швидше за все не існує ефективного (централізованого, паралельного або розподіленого) алгоритму, який вирішує проблему у випадку великих мереж.

2.3.3 Архітектури розподілених мереж

Для розподілених обчислень використовуються різні апаратні і програмні архітектури. Розподілене програмування як правило потрапляє в одну з декількох основних архітектур чи категорій: клієнт-серверних, 3-рівнева архітектура, *n*-рівнева архітектура, розподілених об'єктів, слабкого зв'язку, або тісного зв'язку.

- Клієнт-сервер: клієнтський вузол зв’язується з сервером, форматує дані і відображає їх користувачеві. Введення в клієнт результує у переміщенні вводу до сервера.
- 3-рівнева архітектура: трьохрівневі системи переміщують логіку клієнта на середній рівень, так що можуть бути використані клієнти без будь-якого стану. Це спрощує розгортання додатків. Більшість веб-додатків побудовані на архітектурі 3-рівня.
- *n*-рівнева архітектура: відноситься, як правило, до веб-додатків, які надалі передають свої запроси на інші сервіси. Цей тип програм є одним з найбільш відповідальних за успіх серверів додатків.
- тісно з’єднані (кластерні): відноситься, як правило, до груп машин, що тісно працюють разом, паралельно виконуючи спільний процес. Завдання підрозділяється на частини, які розроблені індивідуально для кожного з них і потім додаються назад разом, щоб знайти остаточний результат.
- Peer-to-peer: архітектура, де немає ніяких спеціальних машин або таких, які забезпечують обслуговування або управління мережевими ресурсами. Замість цього всі обов'язки рівномірно ділиться між усіма машинами, відомих як peers. Peers можуть служити як у якості клієнтів, так і серверів.
- Просторово-орієнтовані: відносяться до інфраструктури, що створює ілюзію (віртуалізацію) одного адресного простору. Дані явно відтворені відповідно до потреб програмного забезпечення. Це дозволяє досягти незалежності у часі, просторі і посиланні.

Інший основний аспект розподіленої обчислювальної архітектури це метод спілкування та координації роботи між паралельними процесами. За допомогою різних протоколів передачі повідомлень, процеси можуть спілкуватися один з одним безпосередньо, як правило, у форматі майстер / відомий. Крім того, база даних, орієнтована на архітектуру може дозволити розподіленим обчисленням бути зробленими без будь-якої форми

безпосереднього спілкування між процесами, а шляхом використання загальної бази даних [Lind et all 2016, с. 46].

2.5 Висновки

В даному розділі було розглянуто основні алгоритми, які використовують для знаходження та тестування простих чисел. В ході дослідження було описано особливості кожного алгоритму, перелічено їх переваги та недоліки. На основі цього дослідження було вибрано алгоритм, який буде використовуватись для задачі.

Крім того були описані особливості класифікації та теоретичні основи побудови розподілених систем. Також були детально описані властивості та можливі архітектури розподілених систем, були представлені способи вирішення поставленої проблеми.

3 АРХІТЕКТУРА

3.1 Вибір архітектури програмного забезпечення реалізації

Першим питанням, що постає перед розробником є вибір архітектури програмного забезпечення. Над цим питанням замислювалися багато років назад, вже тоді, коли починали писати перші програми. Перші програми виконувалися без операційної системи. Вони були написані на мові асемблера і працювали безпосередньо з обладнанням. В 1960-их роках почали з’являтися операційні системи. Разом з їх появою, з програміста були зняті частина завдань - управління головкою жорсткого диска, організація доступу до окремих секторів і організація файлової системи. Однак питання формату зберігання даних і пошуку даних вирішувалося самим програмістом. У 1970-ті роки набувають поширення системи управління базами даних - системи, які

знімали з програміста основні питання по зберіганню та вибірки даних. Подальший розвиток СКБД дозволив працювати з однією базою даних відразу декільком користувачам. Причому ці користувачі могли працювати на різних комп'ютерах, з'єднаних локальною мережею. Зазвичай розділяють два типу клієнт - серверних додатків: з товстим клієнтом і тонким клієнтом. Відмінність полягає в тому, хто виконує основну роботу з обробки даних. Якщо основна робота проводиться на сервері, а клієнт тільки відображає дані - прийнято називати його «тонким». Якщо ж клієнтська програма займається обробкою даних, а сервер - тільки організацією доступу до бази даних - то такий клієнт називається «товстим».

Наступним кроком у розвитку архітектури програмного забезпечення була поява дворівневої архітектури. До дворівневих архітектур відносять системи, в яких є СКБД і програма для роботи з даними. В дворівневих архітектурах кожен користувач має свій екземпляр програми, але всі вони використовують єдине сховище даних.

На цьому розвиток архітектури не зупинився і була запропонована трьохрівнева архітектура програмного забезпечення.

У трьохрівневій архітектурі додається ще один додаток, який працює з базою даних і служить буфером між додатками користувачів і СКБД. У цьому додатку зручно помістити всю обробку даних (розрахунок зарплат, видачу стипендій, обробку та зберігання будь-яких даних). Якщо раптом зміняться вимоги до програми (наприклад, зміниться алгоритм розрахунку прибуткового податку), достатньо буде зробити зміни лише в одній програмі, яка знаходиться на сервері. «Тонкі» клієнти користувачів можна залишити без зміни. Також програми, що використовують трьохрівневу архітектуру є безпечними. Якщо ми надаємо клієнтського додатку доступ до бази даних (як в 2-х рівневій архітектурі), то користувач може «підмінити» клієнтську програму і провести небажані зміни в даних. В трьохрівненвій архітектурі програмісту потрібно писати лише головну частину - серверну. А клієнти використовують

стандартну програму для доступу до серверного додатку. Така стандартна програма називається ультра-тонким клієнтом. І вона є практично на кожному комп'ютері - це інтернет-браузер. Завдяки стандартним протоколам TCP/IP і HTTP і стандартнїій мови HTML інтернет-браузер може спілкуватися з нашим сервером і надавати користувачу інтерфейс. Покажемо схематично трьохрівневу архітектуру на (рис. 3.1).

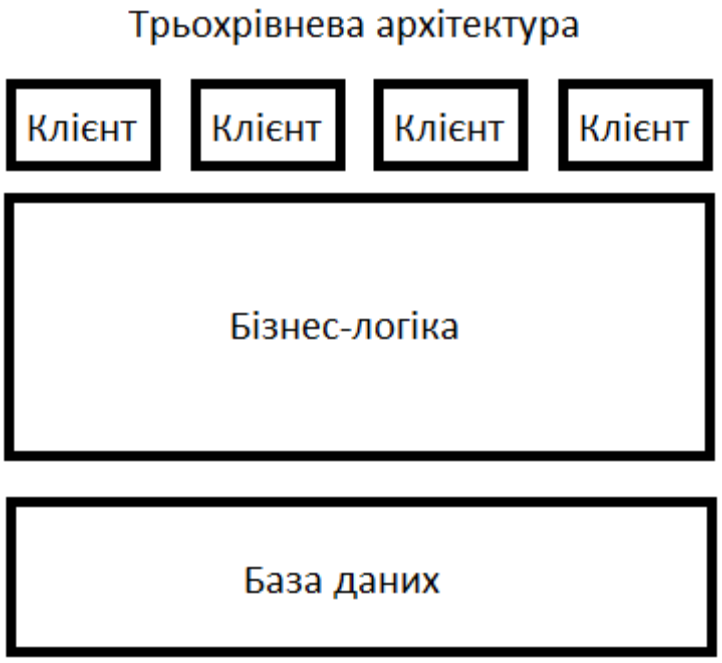

Рисунок 3.1 - Трьохрівнева архітектура

Для виконання поставленої задачі ми будемо використовувати Hadoop. Apache Hadoop — вільна програмна платформа і каркас для організації розподіленої обробки великих обсягів даних (що міряється у петабайтах) з використанням парадигми MapReduce, при якій завдання ділиться на безліч дрібніших відособлених фрагментів, кожен з яких може бути запущений на окремому вузлі кластера. До складу Hadoop входить також реалізація розподіленої файлової системи Hadoop Distributed Filesystem (HDFS), котра автоматично забезпечує резервування даних і оптимізована для роботи MapReduce-застосунків. Для спрощення доступу до даних в сховищі Hadoop розроблена БД HBase і SQL-подібна мова Hive, яка є свого роду SQL для

MapReduce і запити якої можуть бути розпаралелені і оброблені кількома Hadoop-платформами [Develop C# Hadoop streaming programs for HDInsight 2017].

Для того щоб скористатися цим інструментарієм треба встановити екземпляр (вузол) на комп’ютер з однією з підтримуваних операційних систем. Далі, треба дати цьому вузлу «роботу» на опрацювання. В деталях розробнику треба описати що буде виконуватись при розділенні задачі, та що при зборі даних назад.

Вже на цьому показовому описі ми можемо переконатись у зручності трьохрівневого подходу – розробнику не треба думати про те, що відбувається на рівень нижче: як дані розподілені, де вони знаходяться та яким чином буде зроблена передача. Все що нам потрібно – це віртуальна абстракція «дискового простору», до якого ми можемо отримати доступ через Hadoop Distributed File System. Крім того, вона має низку переваг і задовольняє всім необхідним вимогам:

- масштабованість;
- конфігурованість - ізольованість рівнів один від одного дозволяє швидко і простими засобами переконфігурувати систему при виникненні збоїв або при плановому обслуговуванні на одному з рівнів;
- висока безпека;
- висока надійність;
- низькі вимоги до продуктивності та технічним характеристикам терміналів, як наслідок зниження їх вартості.

3.2 Вибір мови програмування для створення додатків

Далі постає питання про вибір тих технологій, за допомогою яких можна реалізувати програму. Хоча найбільш поширеною мовою для створення Hadoop-додатків є Java, існує спосіб у котрий можна використати виконуваний

модуль MapReduce написаний на інших мовах. Зараз користуються популярністю дві платформи: JEE і .NET. Порівняємо їх і виберемо ту платформу, за допомогою якої ми будемо досліджувати виокопродуктивні обчислення.

3.2.1 Кросплатформеність

Однією з найбільших переваг JEE над .NET є кросплатформеність. Але, якщо вірити Microsoft, то це більше не є прерогативою JEE. Microsoft позиціонує .NET як платформу з двоступеневою компіляцією, що дозволяє створювати середовище виконання для будь-якої платформи, подібно Java. На даний момент Microsoft представила специфікації для CLI та C# до ECMA та ISO, зробивши їх стандартами, але не відомо коли треті особи презентують відповідну версію такого програмного забезпечення. Таким чином, можемо зробити висновок, що найбільш вдалим вибором буде JEE.

3.2.2 Багатомовна підтримка

Єдиною мовною основою JEE є Java, що сильно відрізняється від .NET, де підтримується велика кількість мов програмування, включаючи Fortran, COBOL, C++ та Visual Basic. Можна, звичайно, посперечатися щодо того, що єдина мова є більш елегантним рішенням, однак треба визнати, що .NET пропонує більш просте рішення для організацій, які хочуть користуватися знаннями, які вже є у їх розробників. Адже для тих розробників, які використовують мови, відмінні від Java, .NET дає можливість створювати Web-служби звичною для них мовою з мінімальними витратами на перенавчання.

3.2.3 Документація

Зробимо аналіз документації для кожної з платформ. На відміну від JEE, платформа .NET не має гарної документації. Є специфікації на мову C# і на середовище виконання CLR. Але цього недостатньо, щоб зрозуміти всі можливості .NET. Дехто може зауважити, що .NET має MSDN. Але MSDN не є

специфікацією. Це погано структурований збірник інформації, що складається з довідки для різних API, статей, прикладів, але цього недостатньо. Тому можна зробити висновок, що платформа JEE краще задокументована.

3.2.4 Інструментарій для розробки

На сьогоднішній момент єдиним середовищем розробки, який є повністю придатним для реалізації проектів промислового масштабу на платформі .NET, можна вважати лише Visual Studio.NET фірми Microsoft. Що стосується JEE, то тут існує величезна маса інструментів, у тому числі інтегрованих середовищ розробки. Також є відкриті проекти, які наразі повністю безкоштовні. Наведемо список основних або найбільш відомих IDE:

- Oracle JDeveloper;
- Borland JBuilder;
- IntelliJ IDEA;
- NetBeans IDE;
- Eclipse IDE.

Крім того, є ще одне питання, пов'язане з підтримкою інструментарію самої платформою JEE. Але і ця сторона платформи теж відкрита і стандартизована, на відміну від .NET. Також, треба прийняти до уваги те, що самі середовища крос-платформні, на відміну до Visual Studio, де існує жортка прив’язка до операційної системи Microsoft Windows.

Отже, більш гнучким та універсальним буде вибір інструментарію для розробки до платформи JEE.

3.2.5 Ціна

Розповсюджена думка, що рішення на базі JEE є дорогими. При цьому всі забувають, що є безкоштовні засоби, на базі яких можна побудувати промислову систему. Для прикладу, якщо ви не хочете платити за Oracle iAS, то ви можете використовувати JBoss разом із вихідними кодами на додачу. А якщо при цьому ви будете використовувати, наприклад, ОС Linux і СКБД MySQL, то

отримаєте безкоштовну платформу Enterprise-класу. Для Windows і .NET це нереально до тих пір, поки не буде вільної реалізації .NET і всієї необхідної інфраструктури Enterprise-класу.

На основі цього порівняння можна зробити висновок, що для написання програмного продукту більш доречно використовувати саме технологію JEE.

3.3 Вибір способу запуску фреймворку Hadoop

Далі постає питання завантаження інтрументарію та розгортання Hadoop. На вибір у нас є образи віртуальних машин від Cloudera та Hortonworks, а також варіант встановлення програмного забезпечення на певну операційну систему вручну. Зробимо зауваження, що найбільш стабільною операційною системою на яку можна поставити Hadoop – це Linux. Порівняємо можливі альтернативи і зобразимо результати у вигляді таблиці 3.1

Таблиця 3.1 – Порівняння способів

Назва методу	Час розгортання та обслуговування *	Інтерфейс користувача	Наявність додаткового ПО
Ручний	Декілька годин, в залежності ознайомленості користувача з Linux-подібними системами	Той, що вибере користувач (гнучкий вибір серед дистрибутивів ОС Linux)	Можна встановити самому, але це негатино вплине на час розгортання
Hortonworks	Після завантаження готова до користування	Текстовий, в стилі терміналу. Можливість з’єднуватися через веб-інтерфейс.	Нема
Cloudera	Після завантаження готова до користування	Графічний, на базі CentOS	Існує підтримка веб-сервісів що з будь-якого вузла відстежувати та

			керувати виконаням задач

*Зазначимо що до часу розгортання не відноситься час завантаження та встановлення програмного забезпечення для зчитування віртуальних образів операційних систем, та самих ОС.

Беручи до уваги всі плюси та мінуси кожного з методів обираємо спосіб встановлення фреймворку через віртуальну ОС від Cloudera.

3.4 Аналіз архітектури системи

Як вже зазначалось, даний програмний продукт реалізовує трьохрівневу архітектуру. Покажемо взаємозв'язок між рівнями на рис 3.2.

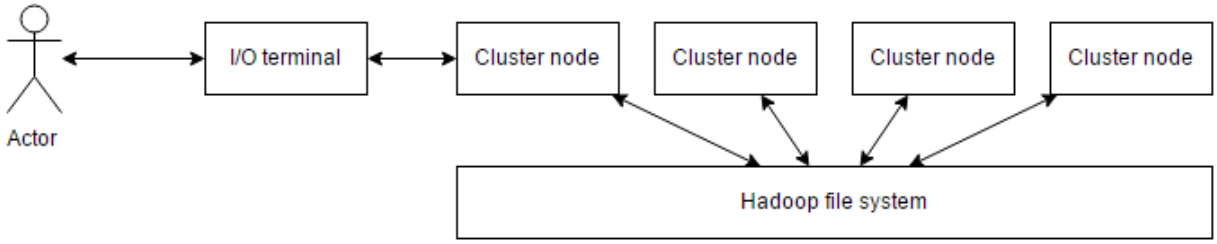

Рисунок 3.2. Взаємозв'язок між рівнями фреймворку

Для того, щоб почати працювати з дистрибутивом розподіленого обчислення задач необхідно мати доступ до будь-якого вузла середовища. При цьому компіляція може бути зроблена будь-де, за наявністю необхідних пакетів та бібліотек. Введення відкомпільованого Java байт-коду повинно бути зроблено за допомогою терміналу введення/виведення, через який можна дати завдання всьому кластеру. В якості параметру до задачі задається шлях до вхідних даних що повинні знаходитися в HDFS. Відповідно початку вхідні дані треба туди помістити за допомогою терміналу. Результати роботи будуть

збережені всередині розподіленої файлової системи, за шляхом також заданим в якоті параметру до задачі.

Для того щоб мати можливіть працювати з даними які з легкітю можуть підлягати компресійній обробці при передачі, користувачеві був предоставлений набір спеціальних типів даних що відповідають стандартним в середовищі Hadoop.

Для того щоб фреймворк мав можливість виконувати декілька компонентів однієї задачі на різних пристроях в потенційно тей самий час, треба описати за яким принципом буде проводитися розподілення, що буде виконуватися на кожному з вузлів та як буде проводитися злиття результатів. За замовчуванням виділяється по одному вузлу на кожен файл вводу. Для наших тестових прикладів цей спосіб є прийнятним. Для того щоб завдати логіку розділення/злиття потрібно створити 2 класи що наслідуються від класів фреймворку Mapper та Reducer відповідно. Результуючу архітектуру покажемо на рис 3.3

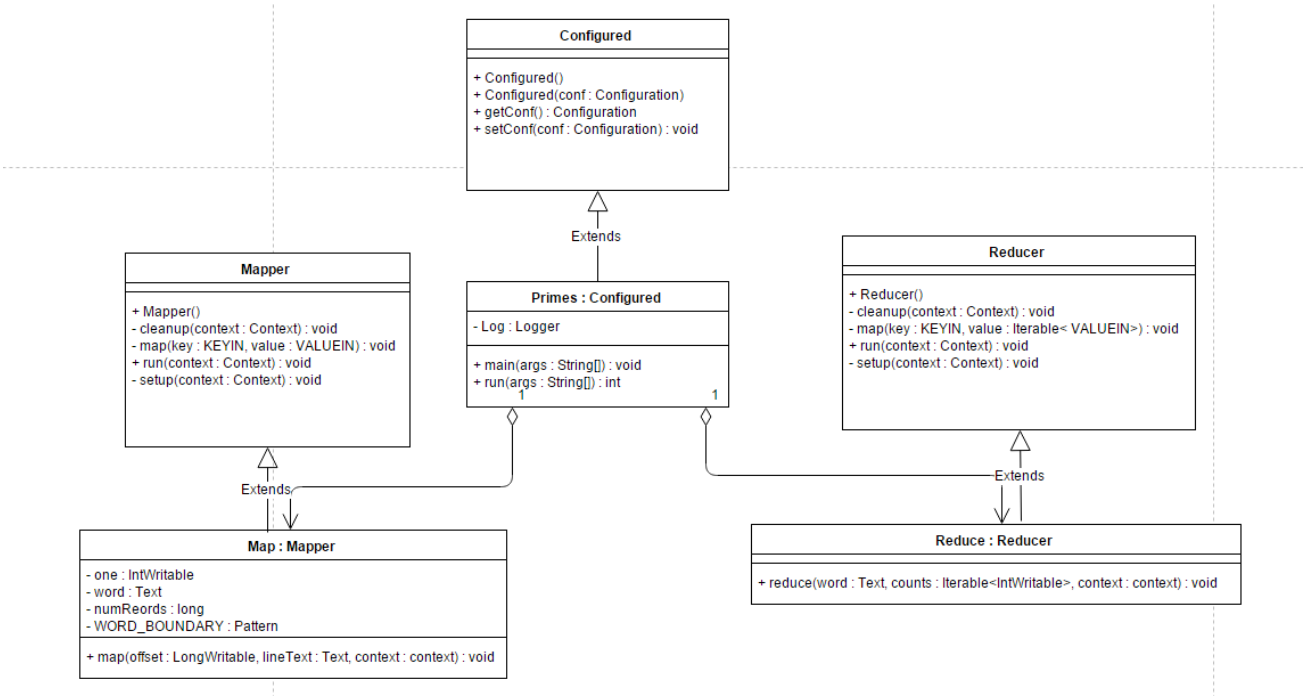

Рисунок 3.3 – Архітектура додатку

3.5 Керівництво користувача

Після того як ми встановили все необхідне програмне забезпечення ми можемо розпочинати роботу. Для цього треба запустити віртуальну машину із файлом ОС на якій встановлений фреймворк Hadoop (рис 3.4).

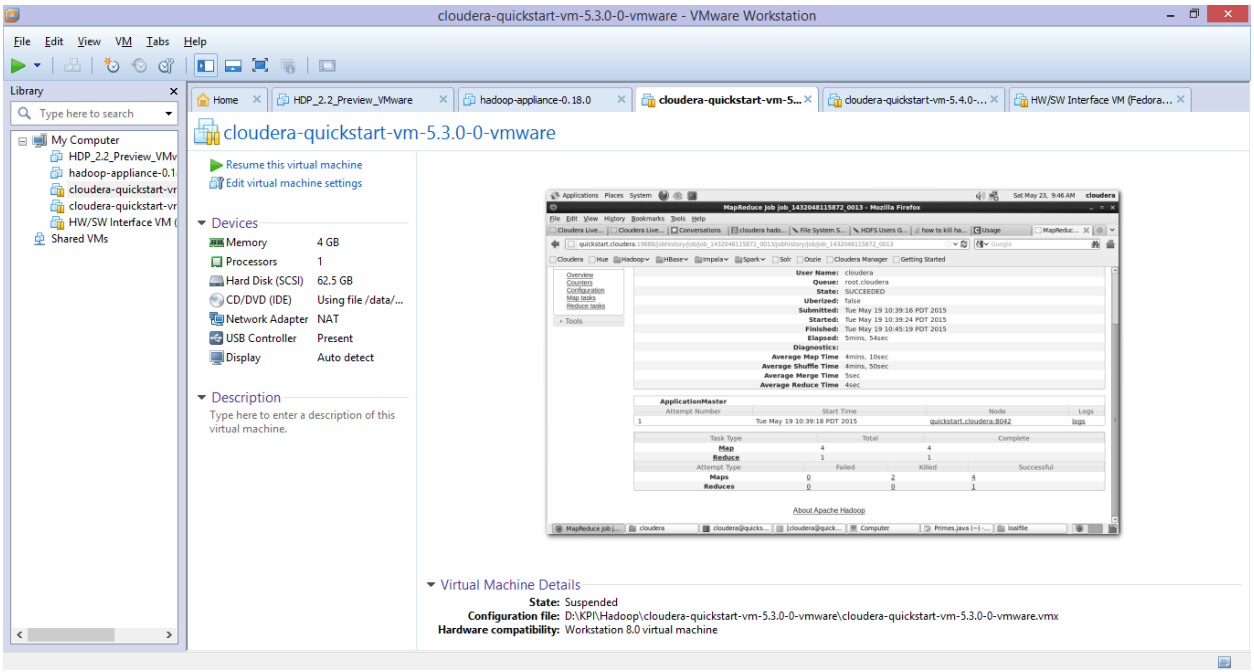

Рисунок 3.4 - Головна сторінка віртуальної машини

Якщо ми вибрали варіант з предвстановленою операційною системою, то після цього можна одразу приступати до написання коду (рис 3.5).

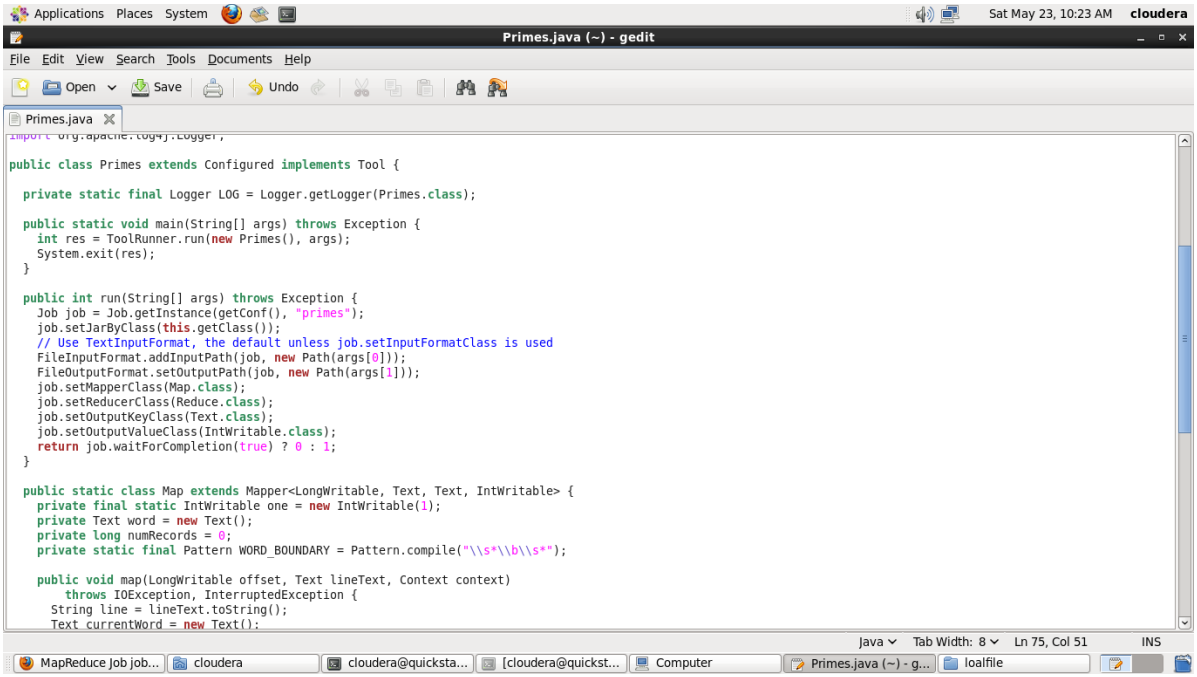

Рисунок 3.5 – Написання коду у віртуальній машині

Цілком можливо налаштувати інтегроване середовище розробки Eclipse, але для показовості прикладу наберемо код у звичайному текстовому редакторі.

Тепер, як ми написали «як» обробляти, ми повинні створити «що» обробляти. Для цього треба створити довільну кількість файлів із вхідними даними. Даний процес зображено на (рис. 3.6).

Рисунок 3.6 – Файли вхідних даних

Але самої лише наявності файлів на дисковому просторі недостатньо. Треба завантажити їх до розподіленої файлової системи фреймворку. Для того щоб ініціалізувати роботу з Hadoop треба відкрити термінал та прописати послідовність команд зображену на (рис 3.7).

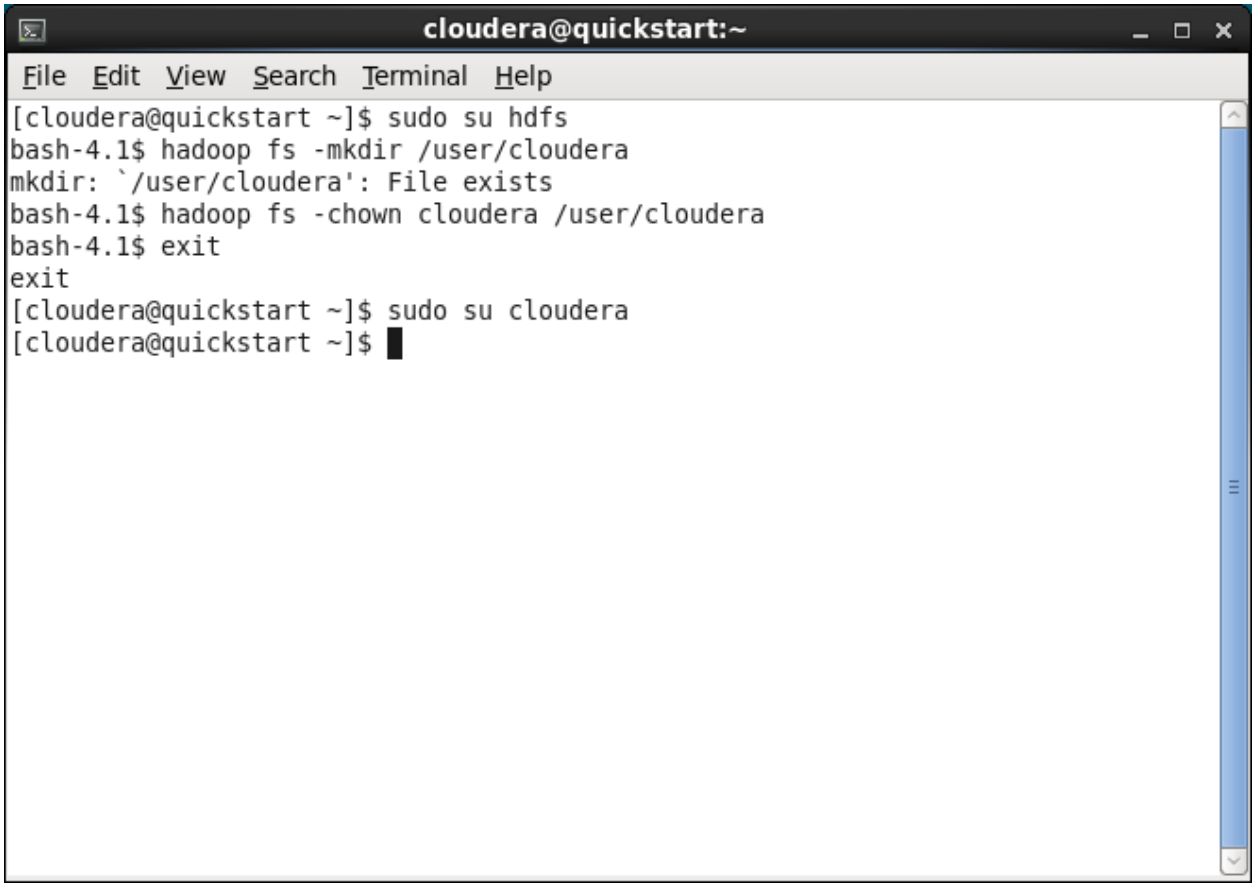

Рисунок 3.7 – Послідовність команд

Тепер треба підготовити робочий простір та створити папки вхідних та вихідних даних як зображено на (рис. 3.8)

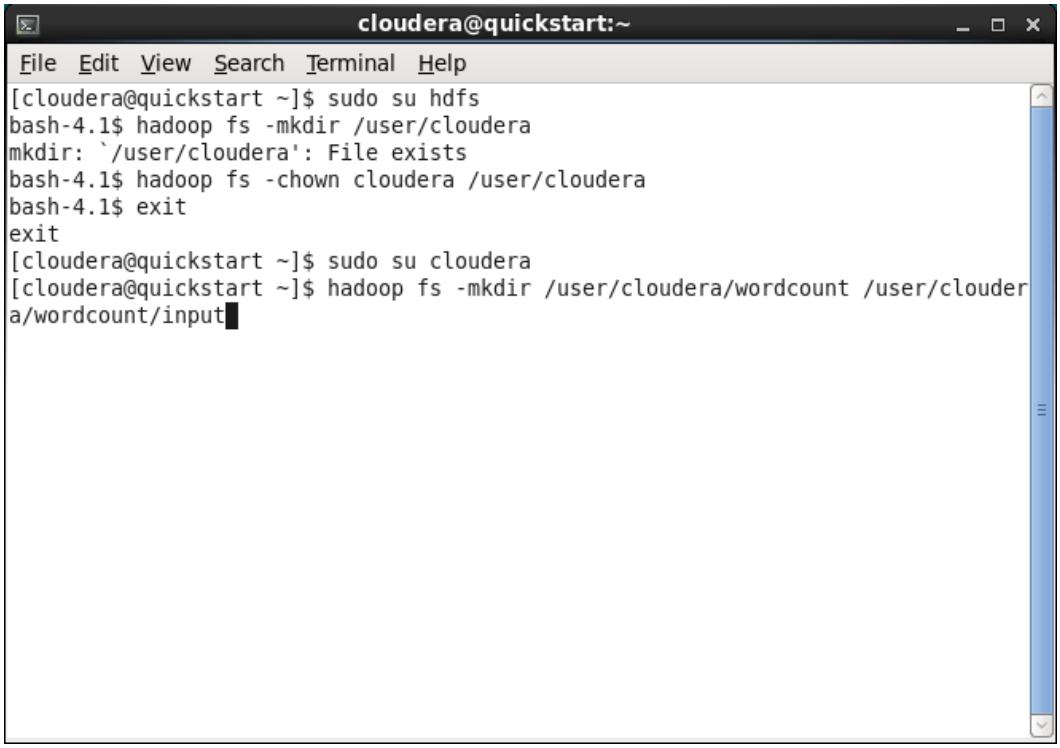

Рисунок 3.8 – Створення папок вводу/виводу

Зараз вже можемо завантажувати вхідні дані до системи (рис. 3.9).

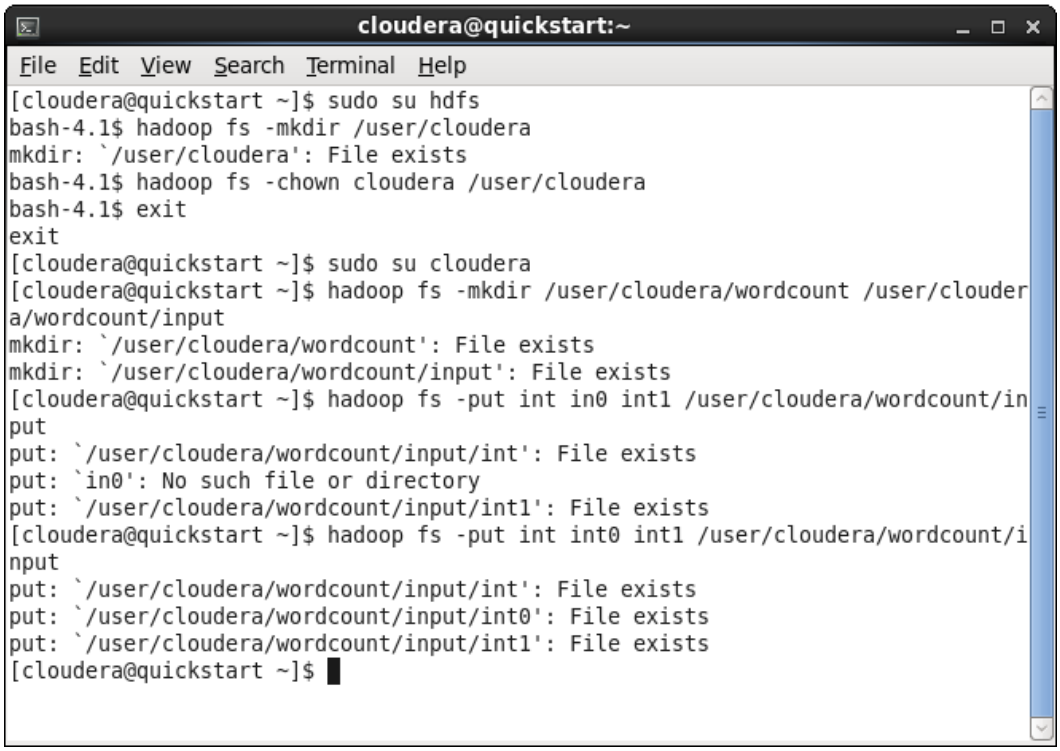

Рисунок 3.9 – Завантаження файлів до фреймворку

Перед безпосереднім виконанням задачі залишився ще один шаг – скомпілювати Java-код та згенерувати виконуючий .jar-файл (рис. 3.10).

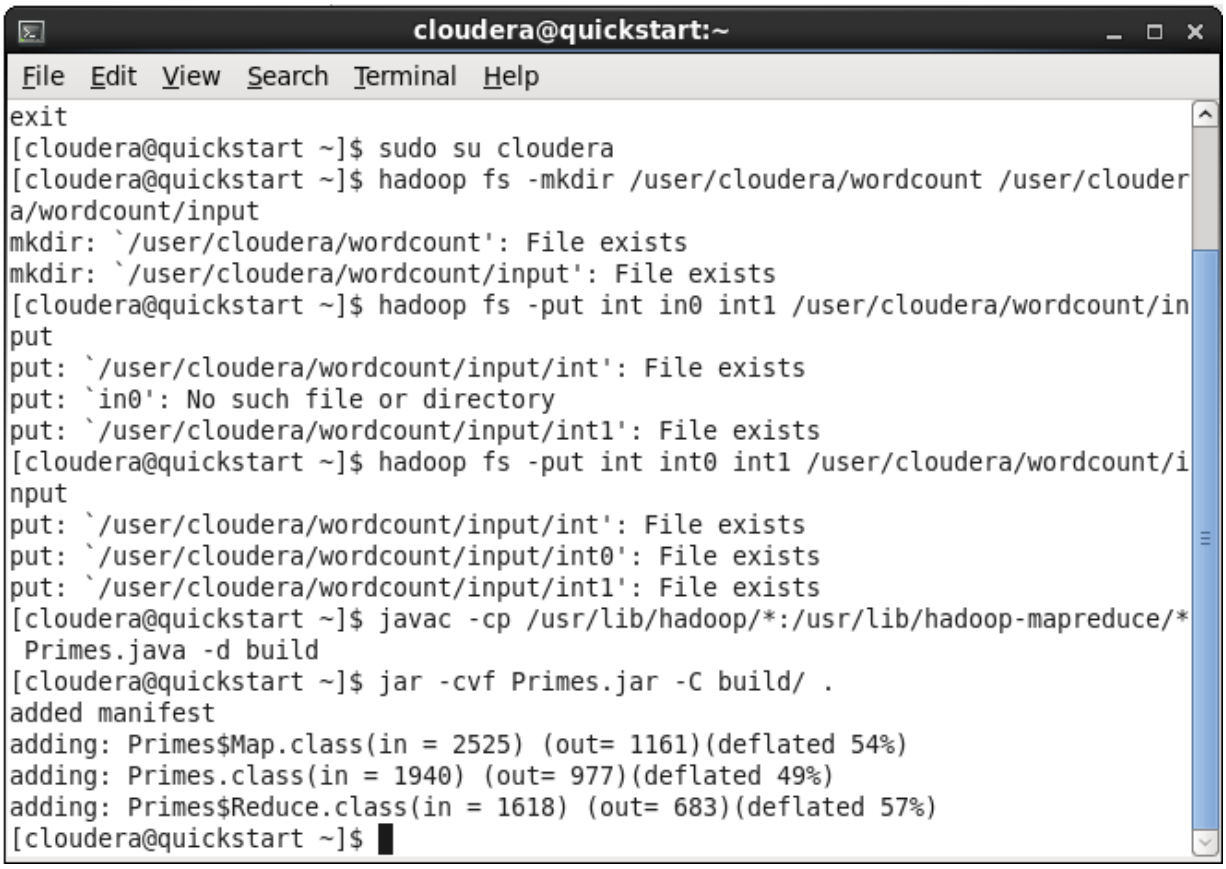

Рисунок 3.10 – Компіляція Java-коду

Для того щоб дати задачу на виконаня для системи треба ввести в термінал команду зображену на (рис. 3.11).

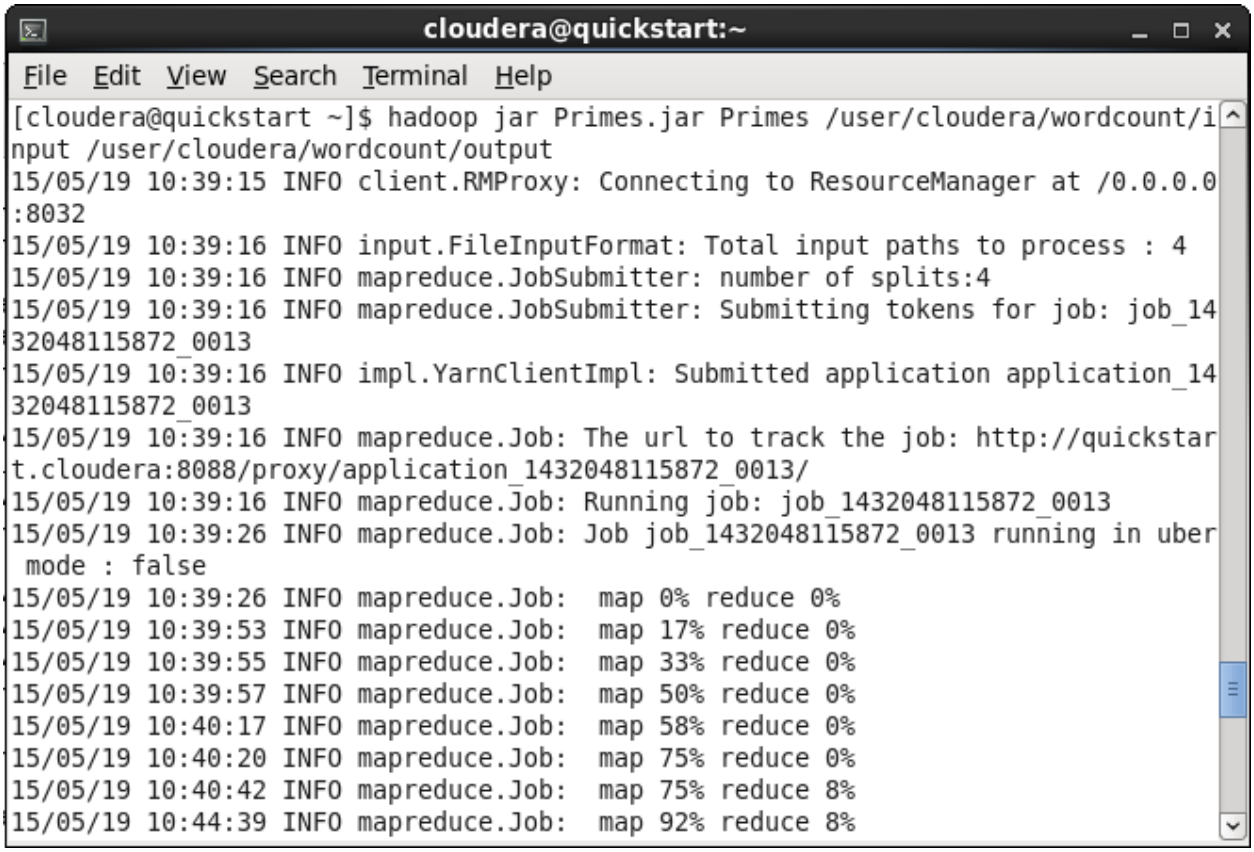

Рисунок 3.11 – Запуск задачі

Як бачимо, робота була успішно подана на опрацювання, можна помітити який статус у задачі та на якому етапі зараз проводиться виконання.

Для більш детального розгляду всіх задач за весь час треба перейти по посиланню вказаному в терміналі. Відкриється вікно браузеру, що продемонстровано на (рис. 3.12).

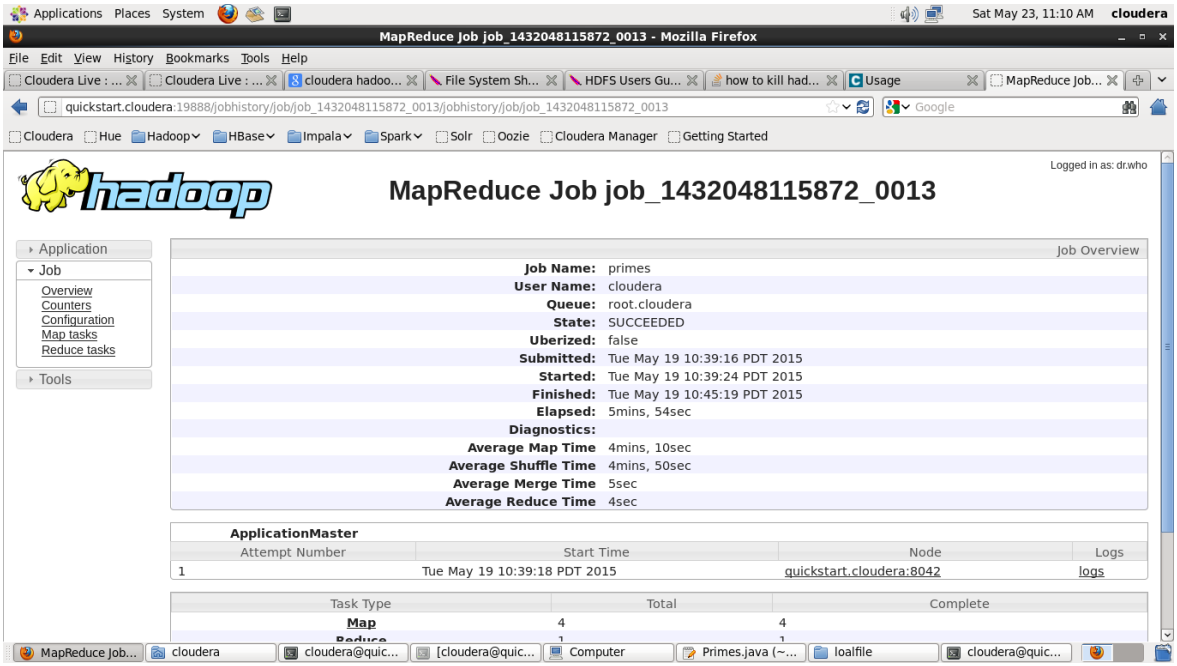

Рисунок 3.12 – Веб-сторінка з інформацією про задачу

Все що залишилося це скачати файл з вихідними даними з дискового простору розподіленої системи, що зображено на (рис. 3.13).

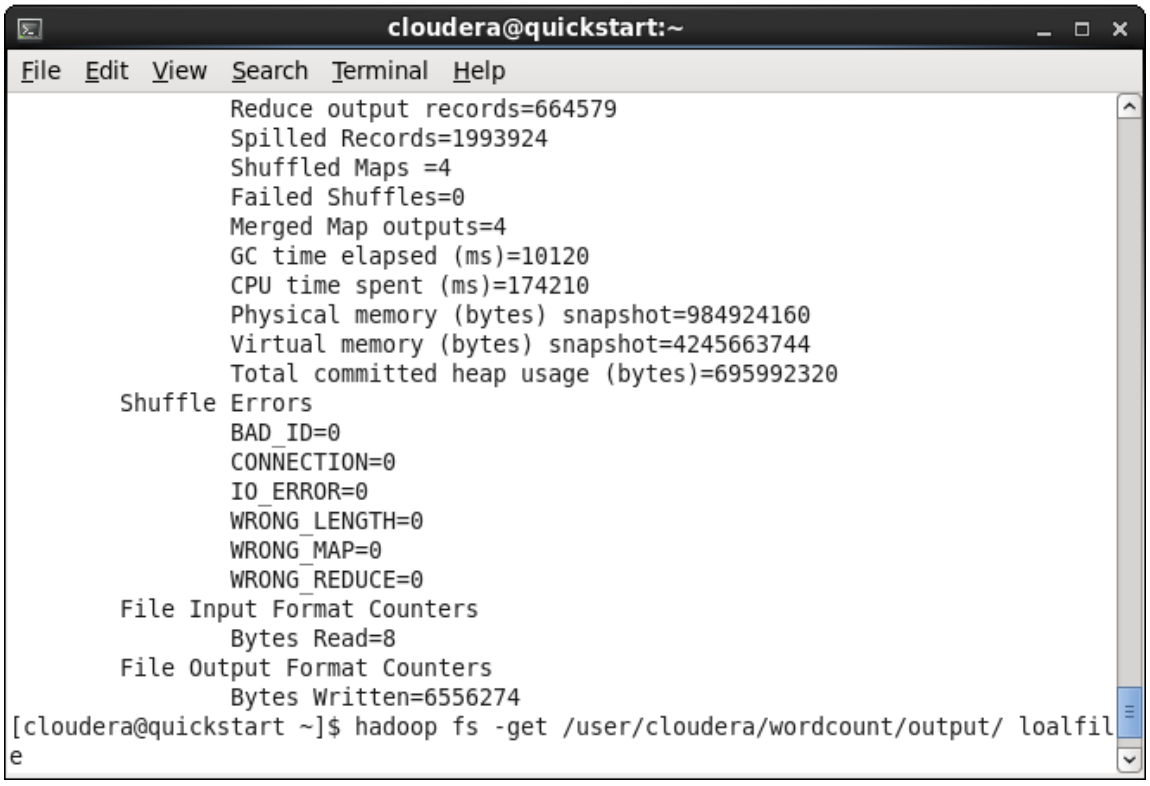

Рисунок 3.13 – Завантаження виводу назад до простору ОС

Пересвідчуємося, що новий файл дійсно з’явився в локальниі файловій системі ОС Linux (рис. 3.14).

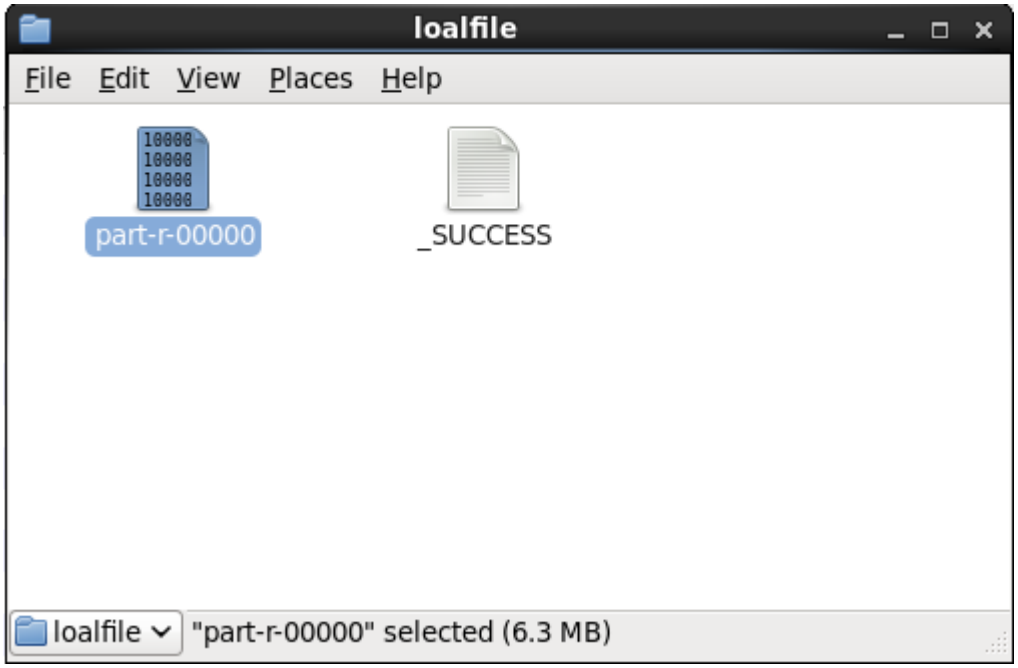

Рисунок 3.14 – Завантажений вихідний файл

3.6 Аналіз отриманих результатів

Зробимо аналіз отриманих результатів. У результаті виконання даної дипломної роботи було отриманого програмний продукт, за допомогою якого можна згенерувати таблицю простих чисел. Даний продукт являє собою додаток, що може бути виконаний на системі розподілених обчислень Hadoop. Для його розміщення було використано послуги компанії Cloudera, що надала вільний та безкоштовний доступ до образу операційної системи Linux для віртуальної машини з предвстановленним фреймворком.

Для генерації таблиці простих чисел були використані ймовірністні алгоритми з послідуючою перевіркою останнього числа, який, як було описано в статті, показав доволі гарні результати.

3.7 Висновки

В даному розділі було проаналізовано особливості архітектур програмного забезпечення та пакету розробки додатків до системи Hadoop і було з’ясовано, що в даному програмному продукті повинна використовуватись архітектура з трьома класами що наслідуються від стандартних з пакету розробки. Був вибраний метод здобуття всієї системи в цілому та реалізована покрокова інструкція по розробці програмних додатків для фреймворку. В ході дослідження було описано реалізацію даної архітектури і описано про реалізацію кожного рівня.

Крім того було написано керівництво користувача, користуючись яким, людина, яка перший раз працює з даною програмою, може легко в ній розібратись і внести необхідні зміни, після чого швидко і без надзвичайних зусиль зможе використати необхідні ресурси для вирішення проблеми. Також зроблено аналіз результатів, які були отримані в ході виконання роботи.

4 АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1 Опис задачі

Метою даної роботи є дослідження використання фреймворку Hadoop, що не э можливим без опису тестових прикладів та системи оцінювання. У цьому розділі будуть описані типи задач які мі будемо розв’язувати, безпосередньо алгоритми розв’язання, виведені результати роботи фреймворку та представлена система порівнянь.

4.2 Опис приклада що буде розв’язуватися

Отже, для показовості будемо брати задачу прогнозу значень ціни нафти, адже це питання є достатньо актуальним, та при належному рівні підготовки та визначення дозволить скласти гідну конкуренцію на ринку біржевих акцій.

Ідея доволі очевидна: маючи набір попередніх значень ціни за барель нафти треба знайти всі значення що будуть далі. Насправді поняття «попередні» є абстрагованим, адже для алгоритму не має значення за який проміжок часу були зроблені заміри – за рік або за час. Все, на що спирається алгоритм при побудові прийнятної моделі передбачення – це самі значення та їх порядок. Тим паче, як було доведено [Justin 2017], поведінка ціноутворення ринкових продуктів, зокрема нафти, має фрактальну структуру, тобто за день ціна може утворювати такий же самий графік як і за рік. Для перевірки фрактальної поведінки цін було вирішено обрати та порівняти два базових підходи до симуляції ідентичного графіку передбачення – на основі наближення ряду Фур’є для косинусів, як було описано раніше, та на основі степеневого поліному. В якості тестової вибірки було обрано використовувати набір середніх за день цін у доларах на барель нафти марки Brent з першого січня по двадцять третє квітня цього року. Усього отримуємо 93 значення що мають наступний вигляд:

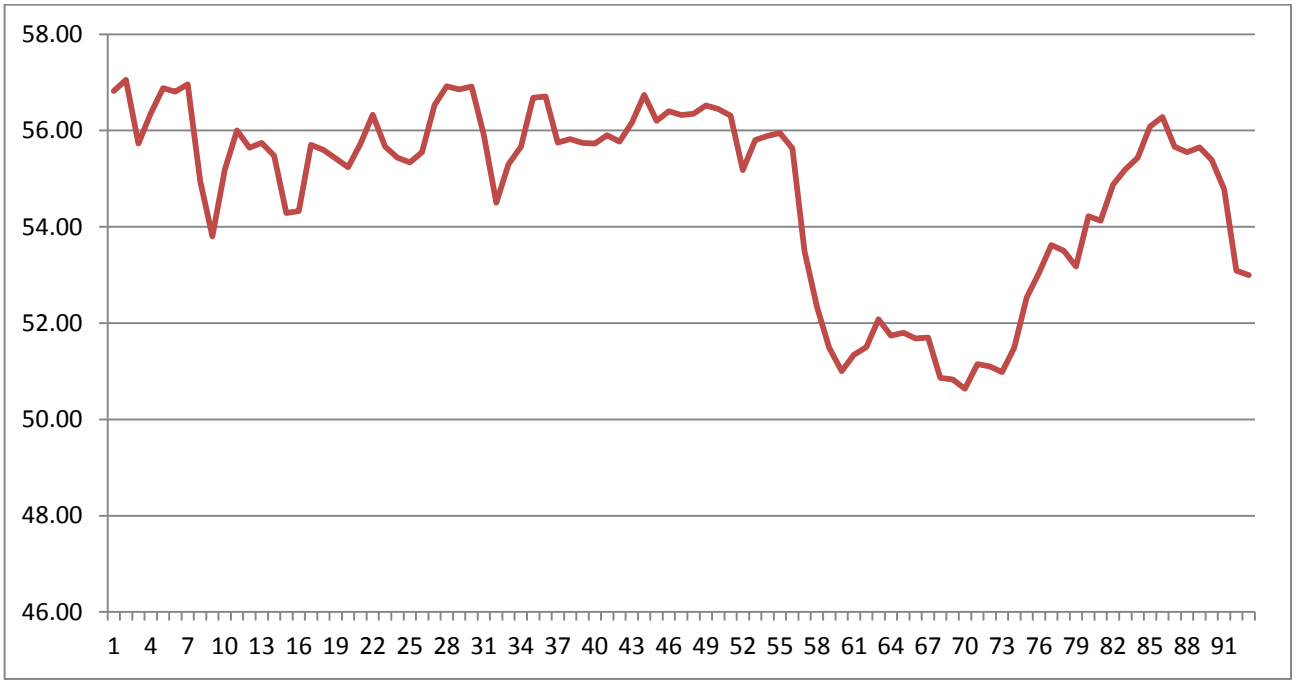

Рисунок 4.1 – Графік поведінки ціни нафти за три місяці

Технічна частина, як би там не було, не така проста. Для розв'язування будемо використовувати синтез алгоритмів запропонованих у статті Качко М. та Яременко В. «Primality test problem» [Kachko et all 2015, с. 136] та статі «Genetic algorithms over distributed systems with MapReduce model» [105] – будемо генерувати велику кількість екземплярів потомків та ітерувати їх покоління за поколінням використовуючи функцію застосованності поки не буде задовільнений критерій зупинки, а потім, обравши кращого кандидата, адаптовувати його результуючі значення для подальшого прогнозу використовуючі нові актуальні значення. Для розділення задачі на вузли будемо використовувати метод описаний Качко М. в статті «Prime table generation» [26]. Роботу способу можна продемонструвати на (рис. 4.1).

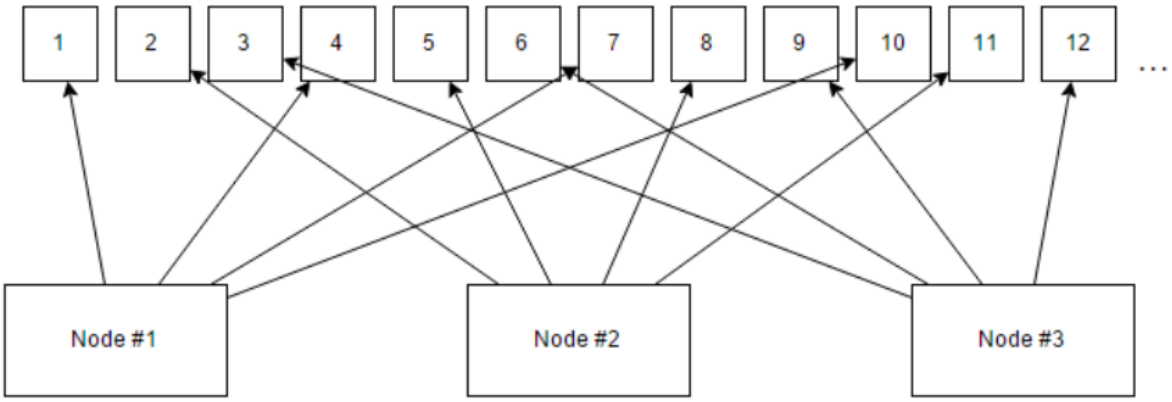

Рисунок 4.2 – Розподілення вхідних даних

Кожен вузол буде перевіряти тільки належну йому частину вхідних поколінь. У такий спосіб на кожен вузол прийдеться приблизно однакова кількість інформації що треба опрацювати. Таким чином буде забезпечене рівноміре завантаження всіх вузлів, а також оптимальний час роботи програми.

4.3 Оцінка результатів

Для того щоб мати змогу об’єктивно оцінити результати, зробимо задачі обчислювально складними, з великою кількостю даних та приблизним часом роботи у декілька хвилин. Основними критеріями за якими буде оцінена робота системи – це час та точність. Буде розроблене програмне забезпечення на мові програмування Java, що буде слідувати прямому нерозводіленому алгоритму, від так буде служити точкою порівняння. Ця програма буде написана за концепцією однопоточного програмування.

Для того щоб зовнішні фактори не впливали на чистоту тесту, обидва варіанти передбачення програмно будуть виконуватися на віртуальній операційній системі встановленій на одному і тому ж персональному комп’ютері, а вимір часу буде проводитися за допомогою вбудованого інструментарію фреймворку та віртуального середовища виконання програм JRE.

Результатом порівняння буде 4 таблиці для кожного з тестових прикладів де будуть записані часи розрахунків для двох концепцій на двох різних платформах та різної кількості даних. Також буде представлена одна кінцева зведена таблиця що буде містити показову порівняльну характеристику усіх чотирьох варіантів підрахунку. Необхідну кількість вхідних даних будемо обирати таким чином: однопоточна пряма програма на Java повинна виконувати завдання максимум за годину, часові характеристики для всіх інших значень повинні починатися в проміжку від секунди.

Для більшої точності кожний тест виконувався десять разів, а результат рахувався як середнє арифметичне серед них.

4.4 Порівняльна таблиця

Таблиця 4.1 – Точність прогнозу для поліноміальної екстраполяції

Кількість поколінь	Прямолінійний спосіб	Розподілення обчислень
10	1.6831517159635392E37	4.21E+36
100	1.0971641732396855E35	2.74E+34
1 000	9.36782384466527E34	2.34E+34
10 000	2.109876515713443E33	5.27E+32
100 000	4.04468986317093E27	1.01E+27
1 000 000	1.1053724600213966E23	2.76E+22

Таблиця 4.2 – Швидкість виконання для поліноміальної екстраполяції

Кількість поколінь	Прямолінійний спосіб, с	Розподілення обчислень, с
10	0.193	35.259
100	0.88	35.831
1 000	7.496	37.742
10 000	74.744	77.92
100 000	740.457	354.128
1 000 000	7276.355	2845.199

Таблиця 4.3 – Точність прогнозу для екстраполяції рядом Фур’є

Кількість поколінь	Прямолінійний спосіб	Розподілення обчислень
10	261723.8631681792	65430.96579
100	151035.88349620058	37758.97087
1 000	22420.05158651477	5605.012897
10 000	18645.83534393031	4661.458836
100 000	12221.89033896767	3055.472585
1 000 000	5913.1856239457	1478.296406

Таблиця 4.4 – Швидкість прогнозу для екстраполяції рядом Фур’є

Кількість поколінь	Прямолінійний спосіб	Розподілення обчислень
10	0.144	34.529
100	0.597	34.8
1 000	5.031	36.742
10 000	48.785	50.92
100 000	481.503	164.821
1 000 000	4683.145	1754.317

4.5 Графіки порівнянь

Далі наведені графіки порівнянь двох методів.

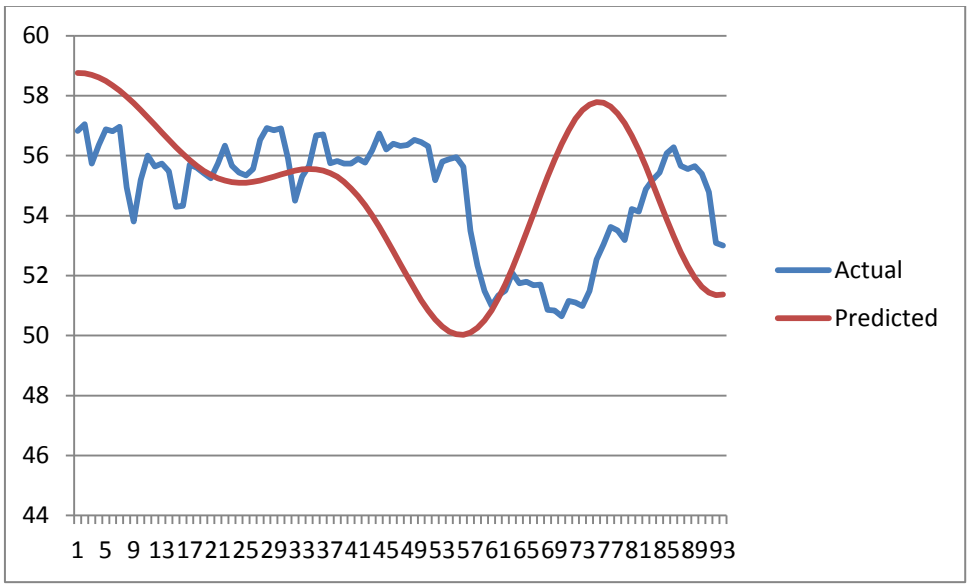

Рисунок 4.3 – Прогноз отриманий прямим методом

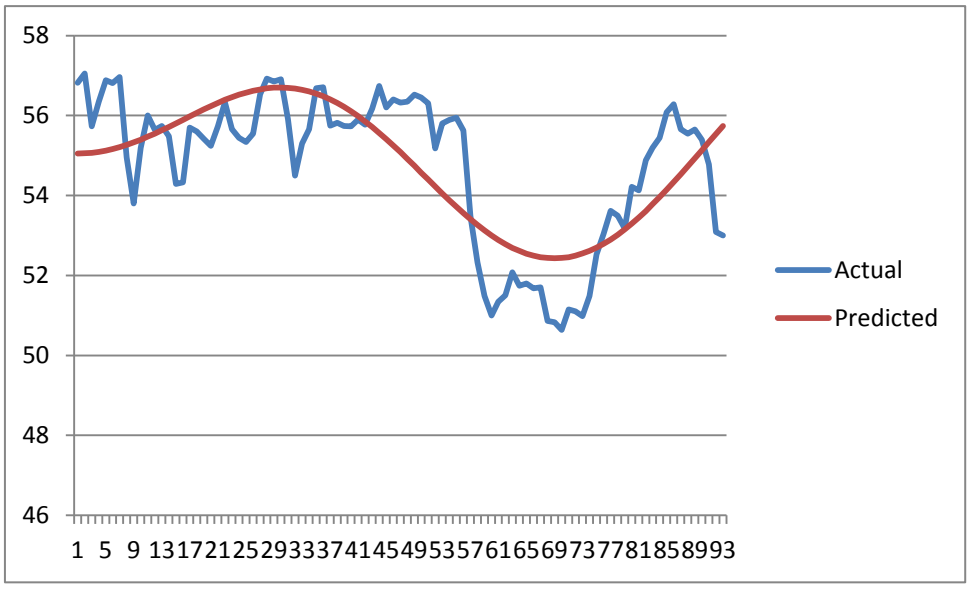

Рисунок 4.4 – Порівняння часу виконання для поліноміальної екстраполяції

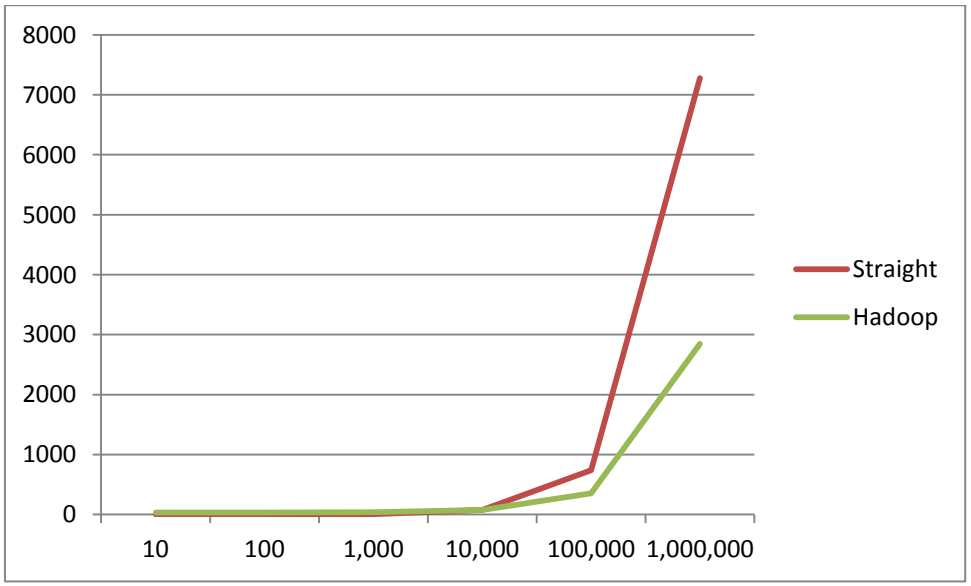

Рисунок 4.5 – Порівняння часу виконання для екстраполяції рядом Фур’є

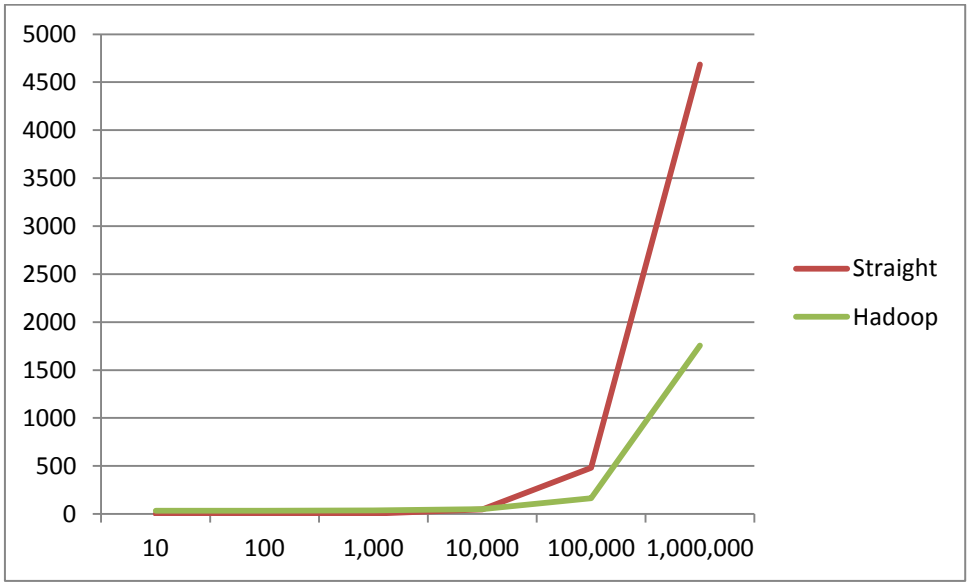

Рисунок 4.6 – Прогноз отриманий розподіленими обчисленнями

4.6 Висновки

В даному розділі проводиться аналіз отриманих результатів, а саме порівняння двох концепцій програмних додатків – з та без підтримання фреймворку HADOOP.

Виявилося, що досліджувана платформа дуже добре поводить себе в середовищі великих обчислень та задач із об’ємними вхідними або вихідними даними.

Також було з’ясовано що за допомогою генетичних алгоритмів можна доволі точно провести апроксимацію деякої величини. Було виявлено, що апроксимація що основана на степеневому поліномі не дає задовільнюючих результатів, адже при такій великій кількості відомих значень змінної (93) не можливо використовувати поліном відповідної степені. Ще однією причиною незастосованності поліноміальної версії алгоритму є суттєвий перепад значущості коефіцієнтів при членах із різним степенем – генетичний алгоритм

буде «намагатися» корегувати коефіцієнт при члені із найвищим степенем, адже він буде давати найбільше корегування точності, при цьому деталі найменших степеней будуть знехтувані.

Апроксимація побудована на базі наближення ряду Фур'є для косинусів дала позитивний результат – отриманий графік прогнозу досить точно імітував графік поведінки реальної досліджуваної величини.

Найважливішим фактом є те, що кожний раз при запуску алгоритму із прийнятною кількістю генерацій прогноз завжди правильно вказував направління зміни досліджуваної величини – зростання чи падіння, що вже само по собі є вагомим показником для такого швидкозмінного і непердбачуваного поля як міжнародна біржа нафтопродуктів.

5 ІДЕЯ СТАРТАП-ПРОЕКТУ

5.1 Опис ідеї проекту

В межах підпункту слід послідовно проаналізувати та подати у вигляді таблиць:

- зміст ідеї (що пропонується);
- можливі напрямки застосування;
- основні вигоди, що може отримати користувач товару (за кожним напрямком застосування);
- чим відрізняється від існуючих аналогів та замінників;

Перші три пункти подаються у вигляді таблиці (табл. 1) і дають цілісне уявлення про зміст ідеї та можливі базові потенційні ринки, в межах яких потрібно шукати групи потенційних клієнтів.

Таблиця 5.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Использовать технологию которая увеличит производительность вычислительных узлов	1. Биология	Вычисление модели белка
	2. Физика	Вычисление положения эл. частиц
	3. Химия	Вычисление результата химических реакций

Аналіз потенційних техніко-економічних переваг ідеї (чим відрізняється від існуючих аналогів та замінників) 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 5.2 – 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	MPI	Одноп оточно сть	Паррал лель			
1.	Скорость разработки	Медл енная	Сред няя	Быстр ая	Средн яя	X		
2.	Сложность	Слож ный	Сло жны й	Прост ой	Средн ий		X	
3.	Быстрота работы	Быстр ый	Быст рый	Медл енный	Средн ий			X

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

5.2 Технологічний аудит ідеї проекту

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 5.3 – Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Обчислення	Скачать готовый дистрибутив Hadoop	Существует	Доступна
2	Обчислення	Развернуть дистрибутив самому	Не существует	Доступна
Обрана технологія реалізації ідеї проекту: Скачать готовый дистрибутив Hadoop				

За результатами аналізу таблиці робиться висновок щодо можливості технологічної реалізації проекту: так чи ні, а також технологічного шляху, яким це доцільно зробити (з поміж названих технологій обираються такі, що доступні авторам проекту та є наявними на ринку).

5.3 Аналіз ринкових можливостей запуску стартап-проекту

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 проводиться аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку.

Таблиця 5.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	10
2	Загальний обсяг продаж, грн/ум.од	100 000

3	Динаміка ринку (якісна оцінка)	Растёт
4	Наявність обмежень для входу (вказати характер обмежень)	Нужно качественное ПО которое работает
5	Специфічні вимоги до стандартизації та сертифікації	-
6	Середня норма рентабельності в галузі (або по ринку), %	200

Середня норма рентабельності в галузі (або по ринку) порівнюється із банківським відсотком на вкладення. За умови, що останній є вищим, можливо, має сенс вкласти кошти в інший проект. За результатами аналізу таблиці робиться висновок щодо того, чи є ринок привабливим для входження за попереднім оцінюванням. Надалі визначаються потенційні групи клієнтів, їх характеристики, та формується орієнтовний перелік вимог до товару для кожної групи.

Таблиця 5.5 – 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	Возможность быстро считать и задействовать ресурсы простаивающих компьютеров	Лаборатории и исследовательские центры	Большие центры будут подвержены определенной бюрократии — долго заключать контракт	Продуктом должно быть легко пользоваться

Після визначення потенційних груп клієнтів проводиться аналіз ринкового середовища: складаються таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають. Фактори в таблиці подавати в порядку зменшення значущості.

Таблиця 5.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Использование квантового компьютера	Отпадёт необходимость в распределенных вычислениях	Строить распределенную сеть на основе квантовых компьютеров
2	Опережение	Конкурент может выпустить ПО быстрее	Реализовать более привлекательное ПО

Таблиця 5.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Использование квантового компьютера	Возникнет возможность использовать в распределенной сети квантовый компьютер	Написать ПО которое сможет использовать квантовый компьютер в сети
2	Появление быстрых сетей	Появятся более быстрые сети каналов связей	Написать протокол который сможет использовать более быструю передачу данных

Надалі проводиться аналіз пропозиції: визначаються загальні риси конкуренції на ринку.

Таблиця 5.8 – 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 - монополія/олігополія/	Много компаний с похожим продуктом	Улучшать качество ПО

монополістична/чиста		
2. За рівнем конкурентної боротьби - локальний/національний/ …	Компании в разных странах	Делать ПО без привязки к географии
3. За галузевою ознакою - міжгалузева/ внутрішньогалузева	По используется в разных сферах	Делать ПО без привязки к сфере
4. Конкуренція за видами товарів: - товарно-родова - товарно-видова - між бажаннями	Может быть применено для разных товаров	Делать ПО без приявзки к товарам
5. За характером конкурентних переваг - цінова / нецінова	ПО разного качества продается по разной цене	Продавать ПО по конкурентноспособной цене
6. За інтенсивністю - марочна/не марочна	В данный момент активно развивается	Идти в ногу со временем

Таблиця 5.8 – Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складові аналізу	Rosetta@home Boinc	Для входа на рынок делать качественно	Быстрота развертывания ПО на кластере	Количество используемых ресурсов	Клиент может перейти к конкуренту
Висновки:	Интенсивная борьба	- чвозможности входа есть - чпотенциальный конкурент PlanetQuest	Да, чем быстрее ПО будет готово к использованию тем привлекательнее	Да, чем больше необходимо ресурсов тем острее нужда в качественном ПО	Необходимо производить товар не хуже конкурентов

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.

На основі аналізу конкуренції а також із урахуванням характеристик ідеї проекту, вимог споживачів до товару та факторів маркетингового середовища визначається та обґрунтовується перелік факторів конкурентоспроможності.

Таблиця 5.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Возможность работы в распределенном режиме	Такая возможность ускорит работу всего процесса, вычислительная способность возрастёт
2	Быстрота развертки	То же самое
3	Цена	Более доступная цена увеличит количество потенциальных клиентов

За визначеними факторами конкурентоспроможності проводиться аналіз сильних та слабких сторін стартап-проекту.

Таблиця 5.11 – Порівняльний аналіз сильних та слабких сторін «назва проекту»

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з Hadoop (назва підприємства)						
			–3	–2	–1	0	+1	+2	+3
1	Возможность работы в распределенном режиме	10				X			
2	Быстрота развертки	10	X						
3	Цена	10		X					

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.

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 5.12 – SWOT- аналіз стартап-проекту

Сильні сторони: качество ПО	Слабкі сторони: сложность работы с ПО
Можливості: распределенные вычисление	Загрози: появление схожего продукта быстрее

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 5.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Переход на другую модель распределения вычислений	75%	3 месяца
2	Добавление возможности использования суперкомпьютеров	10%	5 месяцев

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

5.4 Розроблення ринкової стратегії проекту

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

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

Таблиця 5.13 – 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	Домашнее пользование	Готовы	10 000 клиентов	Низкая	Просто
2	Исследователи	Готовы	1 000 клиентов	Средняя	Просто
3	Научные институты	Готовы	100 клиентов	Высокая	Просто
Які цільові групи обрано: 2 и 3					

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

- якщо компанія зосереджується на одному сегменті – вона обирає стратегію концентрованого маркетингу;
- якщо працює із кількома сегментами, розробляючи для них окремо програми ринкового впливу – вона використовує стратегію диференційованого маркетингу;

- якщо компанія працює із всім ринком, пропонуючи стандартизовану програму (включно із характеристиками товару/послуги) – вона використовує масовий маркетинг.

4.2) 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). Зокрема, ця стратегія припускає, що за рахунок великих можливостей по об'ємах збуту товарів (портфеля укладених контрактів на постачання) і продуктивності підприємство може добитися менших витрат. Ця стратегія зазвичай тісно пов'язана з можливістю досягнення ефекту масштабу і досвіду.

Компанії, що вибирають цю стратегію, проводять ретельний контроль за постійними витратами, знижують виробничі, збутові і рекламні витрати, проводять інвестиції, спрямовані на зменшення витрат, ретельне опрацювання конструкції нових товарів.

Переваги стратегії за Ж.-Ж. Ламбеном:

- фірма здатна протистояти своїм прямим конкурентам навіть у разі цінової війни і в змозі отримувати прибуток при ціні, мінімально допустимій для конкурентів;
- сильні клієнти не можуть добитися зниження ціни нижче рівня, прийнятного для найбільш сильного конкурента;
- низькі витрати забезпечують захист проти сильних постачальників, оскільки дають фірмі велику гнучкість у разі підвищення вхідних витрат;
- низькі витрати створюють бар'єр входу для нових конкурентів і одночасно хороший захист проти товарів-замінників.

В ході конкурентної боротьби з використанням цієї стратегії з ринку вимушені будуть піти фірми, менш ефективні з точки зору величини і структури витрат, нездібні до проведення технологічних новацій, спрямованих на зниження витрат.

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.

Переваги стратегії за Ж.-Ж. Ламбеном:

- по відношенню до прямих конкурентів диференціація знижує ступінь замінності товару, посилює прихильність марці, зменшує чутливість до ціни і тим самим підвищує рентабельність;
- прихильність клієнтів послабляє їх тиск на фірму і перешкоджає приходу на ринок нових конкурентів;
- підвищена рентабельність збільшує стійкість до можливого зростання витрат в результаті дій сильного постачальника;
- відмітні властивості товару і завойована прихильність клієнтів захищають фірму і від товарів-замінників.

Реалізація цієї стратегії вимагає, як правило, більш високих витрат. Проте успішна диференціація дозволяє компанії домогтись більшої рентабельності за рахунок того, що ринок готовий прийняти більш високу ціну (цінову премію бренду).

При веденні конкурентної боротьби з використанням цієї стратегії на ринку в першу чергу терплять фіаско фірми, що не здатні визначати потреби цільових ринків, оперативно реагувати на зміни в ринковому попиті, проводити ефективну політику маркетингових комунікацій, не мають необхідних навичок в області брендингу. Найважливішими здібностями, які повинна мати компанія, що приймає цю стратегію, є з генерування маркетингових ноу-хау, здійснення продуктових новацій.

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 5.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	Специализации	Превосхождение по параметрам	Вычислительная мощность	Специализации

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. Розширюючи таким чином ринковий попит, лідер надає допомогу усім підприємствам, що «йдуть за ним», несучи при цьому основні фінансові

витрати, проводячи найбільш революційні НДДКР. Така стратегія можлива тільки на початкових стадіях життєвого циклу товару, коли попит ще є розширюваним, а взаємний тиск конкурентів ще невеликий. Інакше фірмі лідерові необхідно приймати оборонну або наступальну стратегію.

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. Оборона може бути:

- інновації з метою постановки технологічних бар'єрів для входу в ринок нових конкурентів, подальшого збільшення відриву від них;
- ліквідація ніш для проникнення конкурентів за допомогою розширення товарного асортименту, цінових парасольок, захоплення каналів збуту;
- ведення цінової війни і/або проведення масованої рекламної атаки.

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. Проте, існує межа, при перевищенні якої подальше зростання частки ринку стає невигідним. Це або чисто економічна недоцільність відвойовування добре захищених часток, що сильно захищаються, у дрібніших виробників або ж попадання під дію антимонопольного законодавства.

Наступальна стратегія припускає активну інноваційну політику компанії. Вона постійно атакує власні ж досягнення, збільшуючи розрив між собою і основними конкурентами. Постійні техніко-економічні вдосконалення, модифікація розміру і форми упаковки, використання event- маркетингу – типові складові арсеналу фірм-лідерів.

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.

Проте у більшості випадків найпривабливішою стратегією для компаній-лідерів є диверсифікація, що дозволяє використати переваги масштабу виробництва, know – how.

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. Тому реалізація цієї стратегії вимагає детального опрацювання по наступних напрямах:

- аналіз сильних і слабких сил своїх і фірми-лідера;
- виявлення можливих напрямів атаки;
- ревізія власних сил і ресурсів;
- аналіз можливих дій конкурентів і розробка методів захисту.

Залежно від цього компанія може вибрати одну з альтернативних стратегій: фронтальної або флангової атаки.

Фронтальна атака припускає атаку на сильні сторони конкурента. Така стратегія вимагає наявності у фірми значної переваги над тим, що атакує. У військовій стратегії це співвідношення зазвичай складає 3: 1. При цьому складно не лише стати першим, а мати можливості утримати першість в подальшому. Нового лідера атакуватиме не лише програвша фірма, але і треті, четверті в надії переділити ринок. В силу цього ця стратегія є найбільш ризиковою і у разі невдачі відбувається « виснаження» компанії, що може привести до значного відкидання підприємства. У разі ж успіху компанія стає лідером ринку з усіма перевагами цієї позиції.

Флангова атака передбачає атаку на слабкі сторони фірми-лідера, наприклад, неосвоєні або погано відстежувані регіональні ринки або ринкові сегменти, ціну, значущий для споживача сервіс або показники якості продукції. Особливо б'є по лідерові цінова атака, оскільки, маючи велику ринкову частку, при зниженні ціни в абсолютному вираженні лідер терпить великі втрати, а недостатній приплив фінансових ресурсів відразу ж оголяє раніше приховані латентні слабкі місця компанії, може привести до системної кризи.

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 наслідування лідеру найчастіше має місце у випадку олігополії, коли кожен конкурент прагне уникнути боротьби, особливо цінової, а також у випадку, коли слабо виражений ефект масштабу, що не дозволяє отримати переваги від об'ємів продажів або ж він не грає істотної ролі. Стратегію наслідування лідеру приймають також фірми, які не змогли реалізувати стратегію виклику лідерові.

Компанії, що приймають таку стратегію, зазвичай випускають товари-імітатори, займаючи ринкову частку, яку з різних причин не можуть охопити фірми лідери. Вибір такої стратегії може також бути обумовлений також перевагою локалізації (краще знання ринку, налагоджені зв'язки з клієнтами тощо).

Для ефективної реалізації цієї стратегії компанії повинні задовольняти наступним основним умовам:

- систематичний аналіз сегментації ринку з метою виділення нових ринкових сегментів або таких, що незадовільно обслуговуються;

- ефективне використання НДДКР з метою вдосконалення технологічних процесів і незначних продуктових новацій;
- концентрація на прибутковості, а не на простому зростанні об’ємів продажів;
- постійний аналіз витрат на всіх стадіях виробництва і логістики;
- залишатися досить малим, щоб не бути досить цікавим для фірм-лідерів;
- сильний керівник, здатний не лише формулювати стратегію, але і тримати усю діяльність компанії під власним контролем.

Якщо врахувати, що лідерами ринку можуть бути лише декілька компаній, то ця стратегія є наймасовішою.

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.

Ніша, для того, щоб вона була привабливою для компанії, повинна задовольняти таким умовам:

- бути досить прибутковою, щоб робити доцільним процес виробництва і обслуговування;
- залишатися стабільною упродовж тривалого проміжку часу;
- має бути добре захищеною, мати високі вхідні бар’єри;
- бути непривабливою для конкурентів;
- відповідати цілям і ресурсам компанії, її специфічним можливостям.

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 5.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати	Чи буде компанія копіювати основні характеристики	Стратегія конкурентної поведінки*

		існуючих у конкурентів?	товару конкурента, і які?	
1	Нет	И новых и старых	Нет	Стратегия лидера

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 5.17 – 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	Вычислительная мощность	По конкурентной обстановке на рынке	Более быстрая передача данных по каналам связи — больше вычислительная мощность	Скорость, надёжность, стабильность
2	Простота разработки	По конкурентной обстановке на рынке	ПО которое простое в обращении	Быстрота, простота, ненавязчивость

Результатом виконання підрозділу має стати узгоджена система рішень щодо ринкової поведінки стартап-компанії, яка визначатиме напрями роботи стартап-компанії на ринку.

5.4 Розроблення маркетингової програми стартап-проекту

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

Концепція товару - письмовий опис фізичних та інших характеристик товару, які сприймаються споживачем, і набору вигод, які він обіцяє певній групі споживачів.

Таблиця 5.18 – Визначення ключових переваг концепції товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Быстрота подсчёта данных	Более быстрая передача данных	Будет передавать данные по каналам связи быстрее остальных
2	Простота создания распределенной среды	Более простой пользовательский интерфейс	Простой интерфейс позволит разрабатывать ПО быстрее

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

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.

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

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)

Таблиця 5.19 – Техніко-економічні характеристики товару

№ п/п	Група показників	Склад показників
1	Економічні	Вартість обслуговування, експлуатації, утилізації, витратних матеріалів, ремонту, знижки
2	Призначення (технічні)	Показники, що визначають головний напрямок використання товару та можливу сферу його застосування: класифікаційні показники, складу і структури, технічної досконалості. Приклад: маса, розміри, кількість елементів, види та характеристики матеріалу (для меблів); густина, калорійність, відсоткове співвідношення цукру, жиру, солі (для харчових продуктів); ємність салону, кількість пасажирських місць, потужність двигуна (для автобуса); % вуглеводню (для нафти); структура плетіння, склад пряжі (для тканин) тощо)
3	Надійності	Характеризують здатність товару безвідмовно функціонувати: безвідмовність, довговічність, ремонтопридатність. Приклад: кількість разів використання (запуску в роботу) товару, строк безвідмовної праці, гарантійний термін.
3	Технологічні	Показники, що характеризують можливість оптимізації витрат матеріалів, праці, коштів, часу під час технологічної підготовки виробництва, виготовлення та використання товару. Приклад: трудомісткість виготовлення та технологічна собівартість товару.
4	Ергономічні	Показники ступеню адаптованості технічних та конструктивних рішень виробу до біологічних властивостей людини та середовища використання товару: гігієнічні, антропометричні, фізіологічні та психологічні. Приклад: рівень освітлення, температури, вологості, токсичності, шуму, вібрацій, достатність робочого простору, раціональність розміщення, зручність спостереження за сигнальними елементами, відповідність зросту людини, рівень статичного напруження м’язів робочого органу людини, відповідність виробу можливостям сприйняття інформації користувачем, зручність користування під час виконання основних та допоміжних операцій, зручність управління, простота набуття навичок.
5	Органолептичні	Показники, що визначають властивості товару, які людина може визначити за допомогою своїх органів чуття. Приклад: смак, присмак, запах, забарвлення, каламутність.

№ п/п	Група показників	Склад показників
6	Естетичні	Показники, що оцінюють зовнішній вигляд товару. Приклад: інформаційна виразність, раціональність форми, цілісність композиції, досконаліть виробничого виконання, стабільність товарного вигляду, відповідність стилю, відповідність моді.
7	Транспортабельності	Показники, що визначають пристосованість продукції до транспортування, підготовчих, початкових і кінцевих операцій перевезення. Приклад: Середню трудомісткість підготовки одиниці продукції до перевезень (із навантаженнями та закріпленнями включно), середню вартість пакування в тару для перевезення, середню тривалість завантаження/розвантаження партії товару з одиниці рухомого складу.
8	Екологічності	Показники, що характеризують рівень негативного впливу на довкілля. Приклад: вміст шкідливих домішок у викидах, ймовірність викидів шкідливих домішок під час транспортування, збереження, експлуатації.
9	Безпеки	Показники безпечності та нешкідливості споживання товару. Приклад: можливість безпечної праці протягом визначеного часу, час спрацювання захисних пристроїв, електрична міцність високовольтних мереж, наявність блокуючих пристроїв, ременів безпеки, ізоляції, аварійної сигналізації.

Таблиця 5.20 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Необходимость быстро считать большие данные и использовать вычислительные мощности незадействованых компьютеров		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Скорость вычислений 2. Быстрота разработки		
	Якість: скорость вычислений должна быть больше чем у конкурентов		
	Пакування: онлайн-дистрибутив		
	Марка: Hadoop		
III. Товар із підкріпленням	До продажу: использование быстрых протоколов передачи данных		
	Після продажу: поддержка квантовых компьютеров		

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. Аналіз проводиться експертним методом.

Таблиця 5.21 – 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	Около 100 000	Около 100 000	Около 1 000 000	120 000 — 150 000

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення:

- проводити збут власними силами або залучати сторонніх посередників (власна або залучена система збуту);
- вибір та обґрунтування оптимальної глибини каналу збуту;
- вибір та обґрунтування виду посередників.

Таблиця 5.22 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Будут покупать по одной единице на центр	Розничная торговля	Одноуровневый	Собственными силами

5.5 Висновки

На даний момент є можливість ринкової комерціалізації проекту. Наявний попит, динаміка ринку, було виявлено що робота буде рентабельною. Також було виявлено що є перспективи впровадження з огляду на потенційні групи клієнтів, бар’єри входження, стан конкуренції, конкурентоспроможність проекту. Доцільно обрати варіант впровадження позиціювання за показниками якості для ринкової реалізації проекту. За результатами дослідження було виявлено що подальша імплементація проекту є доцільною.

ВИСНОВКИ

Дипломна робота присвячена задачі розподілення обчислень у складних системах для вирішення комплексних задач.

У результаті виконання даної дипломної роботи було отриманого програмний продукт, за допомогою якого можна дослідити роботу системи розподілених обчислень HADOOP.

Також був запропонований та досліджений алгоритм апроксимації графіку поведінки деякої величини на базі генетичних алгоритмів що виявляють найбільш пристосований поліном або ряд Фурє'.

Було з'ясовано що за допомогою генетичних алгоритмів можна доволі точно провести апроксимацію деякої величини. Було виявлено, що апроксимація що основана на степеневому поліномі не дає задовільнюючих результатів, адже при такій великій кількості відомих значень змінної не можливо використовувати поліном відповідної степені. Ще однією причиною незастосованності поліноміальної версії алгоритму є суттєвий перепад значущості коефіцієнтів при членах із різним степенем – генетичний алгоритм буде «намагатися» корегувати коефіцієнт при члені із найвищим степенем, адже він буде давати найбільше корегування точності, при цьому деталі найменших степеней будуть знехтувані.

Апроксимація побудована на базі наближення ряду Фур'є для косинусів дала позитивний результат – отриманий графік прогнозу досить точно імітував графік поведінки реальної досліджуваної величини.

Найважливішим фактом є те, що кожний раз при запуску алгоритму із прийнятною кількістю генерацій прогноз завжди правильно вказував направління зміни досліджуваної величини – зростання чи падіння, що вже само по собі є вагомим показником для такого швидкозмінного і непердбачуваного поля як міжнародна біржа нафтопродуктів.

Для розробки програмного забезпечення було розроблено іноваційний алгоритм екстраполяції та розподілення задач між вузлами, роботу яких було проаналізовано та виведені результати в четвертому розділі. Виявилося, що фреймворк в цілому справляється з поставленою задачею, але є такі випадки або такі задачі де використання більш простої системи буде доцільнішим. Також, є алгоритми де HADOOP в зв'язку із специфічним алгоритмом роботи платформи на рівень ефективніше виконує програму.

У першому розділі було наведено аналіз існуючих систем для вирішення заданої проблеми, описано про особливості та проблематики задачі розподілення обчислень, була сформульована постановка і актуальність даної задачі.

Другий розділ було присвячено дослідженню існуючих методів для розв'язання проблеми, побудовано математичну модель, описано за якими критеріями визначалась якість результату роботи системи, наведено алгоритм роботи програми для тестуваня даної системи.

У третьому розділі було обґрунтовано вибір платформи та мови реалізації програмного продукту, було зроблено аналіз архітектури системи. Крім того, були розроблені експерименти за допомогою яких можна було проводити аналіз результатів, отриманих в роботі.

У четвертому розділі проводився аналіз отриманих результатів, а саме порівняння двох концепцій програмних додатків – з та без підтримання фреймворку HADOOP.

П'ятий розділ був присвячений аналізу можливостей ринкової комерціалізації проекту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Andrews, Gregory R. Foundations of Multithreaded, Parallel, and Distributed Programming. / Andrews, Gregory R. – Addison–Wesley: 2000. p. 140-142
2. Arnault F. "Rabin-Miller Primality Test: Composite Numbers Which Pass It." / Arnault F. // Math. Comput. - New York, 1995 355-361.
3. Arora Sanjeev and Barak Boaz. Computational Complexity – A Modern Approach. / Arora Sanjeev and Barak Boaz. – Cambridge: 2009.
4. Black F. The Pricing of Options and Corporate Liabilities / F. Black, M. Scholes. // Journal of Political Economy. – 2013. – С. 637–659.
5. Develop C# Hadoop streaming programs for HDInsight — Режим доступа : https://azure.microsoft.com/en-us/documentation/articles/hdinsight-hadoop-develop-deploy-streaming-jobs/ — Дата доступа : 06.05.17.
6. Evaluating MapReduce for Multi-core and Multiprocessor Systems / [C. Ranger, R. Raghuraman, A. Penmetsa та ін.]. // IEEE 13th International Symposium on High Performance Computer Architecture. – 2007. – С. 13.
7. Goldberg D. E. Genetic Algorithms in Search, Optimization & Machine Learning / Goldberg. // Addison-Wesley. – 2015.
8. Holger J. Pletsch Deepest All-Sky Surveys for Continuous Gravitational Waves. / Holger J. Pletsch. –Hannover. : Leibniz Universität Hannover, July 25, 2010. – 5 p.
9. Ingber L. Genetic Algorithms and Very Fast Simulated Reannealing: A Comparison / L. Ingber, B. Rosen. // J. of Mathematical and Computer Modeling 16. – 2012.
10. Justin K. A Trader's Guide to Using Fractals [Електронний ресурс] / Kuepper Justin // Investopedia. – 2017. – Режим доступу до ресурсу: http://www.investopedia.com/articles/trading/06/fractals.asp.

11. Kachko N. A. Genetic algorithms over distributed systems with MapReduce model / Kachko. // Institute for Applied System Analysis at the Igor Sikorsky Kyiv Polytechnic Institute. – 2017. – С. 199–200.
12. Kachko N. Prime table generation / Kachko N.: міжнародна науково-технічна конференція «САІТ-2015», 23 червня 2015, Київ, Україна : матеріали. – К. : НТУУ «КПІ», 2015.
13. Lakhmi C. J. Fuzzy Systems and Genetic Algorithms: Industrial Applications / C. J. Lakhmi, N. M. Martin., 2012. – (CRC Press LLC).
14. Lämmel R. Google's Map Reduce programming model — Revisited / Lämmel. // Science of Computer Programming. – 2008. – С. 1–30.
15. Lind P. and Alm M. A. database-centric virtual chemistry system. / Lind P. and Alm M. A. – Huddinge: 2006. p. 46
16. MapReduce [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/MapReduce.
17. Marozzo F. P2P-MapReduce: Parallel data processing in dynamic Cloud environments / F. Marozzo, D. Talia, P. Trunfio. // Journal of Computer and System Sciences. – 2012. – С. 1382–1402.
18. Misco: a MapReduce framework for mobile systems / [A. Dou, V. Kalogeraki, D. Gunopulos та ін.]. // Proceedings of the 3rd International Conference on PErvasive Technologies Related to Assistive Environments. – 2010.
19. Muehlenbein H. Predictive Models for the Breeder Genetic Algorithm / H. Muehlenbein, Schlierkamp-Vosen. // Evolutionary Computation 1. – 2013.
20. Nedunchelian R. Dynamic task scheduling using parallel genetic algorithms for heterogeneous distributed computing / Nedunchelian. // Sri Venkateswara College, Pennalur, Sriperumbudhur. – 2005.
21. Peng D. Large-scale Incremental Processing Using Distributed Transactions and Notifications / D. Peng, F. Dabek. // OSDI. – 2010. – №10.
22. Poli R. A Field Guide to Genetic Programming / R. Poli, W. B. Langdon, N. F. McPhee., 2008.

23. Price K. Differential Evolution: A Fast and Simple Numerical Optimizer / Price. // NAFIPS’16. – 2016. – С. 524–527.
24. Prime Number. — Режим доступа : http://mathworld.wolfram.com/PrimeNumber.html. — Дата доступа : 26.04.17.
25. Richard Crandall and Carl Pomerance. Prime Numbers: A Computational Perspective. / Richard Crandall and Carl Pomerance. – Springer : 2005. p.159–190.
26. Richard Crandall and Carl Pomerance. Prime Numbers: A Computational Perspective. / Richard Crandall and Carl Pomerance. – Springer : 2005. p.334–340.
27. Scott Guthrie: Silverlight and the Cross-Platform CLR— Режим доступа : https://channel9.msdn.com/shows/Going+Deep/Scott-Guthrie-Silverlight-and-the-Cross-Platform-CLR — Дата доступа : 06.05.17.
28. Séroul R. "The Sieve of Eratosthenes." / Séroul, R. // Programming for Mathematicians. - Berlin: Springer-Verlag, 2000, pp. 169-175.
29. Spark: Cluster Computing with Working Sets / [M. Zaharia, M. Chowdhury, M. Franklin та ін.]. // HotCloud. – 2010.
30. Storn R. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces / R. Storn, K. Price // Journal of Global Optimization. – 1997.
31. The Prime Glossary — Режим доступа : http://primes.utm.edu/glossary/home.php. — Дата доступа : 02.05.17.
32. Ullman J. D. Designing good MapReduce algorithms / Ullman. // The ACM Magazine for Students. – 2012. – С. 30.
33. What is the purpose of Einstein@Home? — Режим доступа : http://einstein.phys.uwm.edu/PartialS3Results/node3.html — Дата доступа : 01.05.17.

34. Wickham. The split-apply-combine strategy for data analysis / Wickham, Hadley. // Journal of Statistical Software. – 2011. – С. 1–29.
35. Yaremenko V., Kachko N. Primality test problem / Yaremenko V., Kachko N. : Матеріали XIV всеукраїнської науково – практичної студентської конференції, 07 квітня 2015, Київ, Україна : матеріали. – К. : НТУУ «КПІ», 2015. – С. 136.
36. Yaremenko V., Kachko N. Primality test problem / Yaremenko V., Kachko N. : Матеріали XIV всеукраїнської науково – практичної студентської конференції, 07 квітня 2015, Київ, Україна : матеріали. – К. : НТУУ «КПІ», 2015. – С. 136.
37. Ziny F. Optimization of routing control with Differential Evolution / Ziny. // private communication. – 2015.
38. Баудер Е. Социальный маркетинг: особенности развития соци- ального маркетинга в сфере малого бизнеса / Е. Баудер., 2012. – 272 с.
39. Богданов В.Л. Концепція програми наукових досліджень НАН України. / Богданов В.Л.. - К. : Президія НАН України, 2012. -7 с.
40. Волощук В. С. Евристичні генетичні алгоритми пошуку : автореф. дис. на здобуття наук. ступеня канд. техн. наук / Волощук В. С. – Одеса, 2010.
41. Гладков Л. А. Генетичні алгоритми: Навчальний посібник / Л. А. Гладков, В. В. Курейчик, В. М. Курейчик. – Москва: Фізматліт, 2006. – 320 с.
42. Кобець О. І. Моделі і методи розподілу ресурсів в системах, побудованих на хмарних обчисленнях. / Кобець О. І.// Міжнародна наукова інтернет-конференція Соціум. Наука. Культура., - К.: Національний технічний університет України "Київський політехнічний інститут", 2012 - С. 28-33.
43. Крицун К. Перспективи застосування фрактального аналізу на валютному ринку україни / К. Крицун. // Bulletin of Taras Shevchenko National University of Kyiv. Economics.. – 2014. – С. 48–53.

44. Курейчик В. М. Пошукова адаптація: теорія і практика / В. М. Курейчик, Б. К. Лебедєв, О. К. Лебедєв. – Москва: Фізматліт, 2013. – 272 с.
45. Організація інтелектуальних обчислень [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: http://www.victoria.lviv.ua/html/oio/html/theme10.htm#10_4.
46. Офіційний сайт PrimeGrid. — Режим доступа : http://www.primegrid.com. — Дата доступа : 26.04.17.
47. Паралельні бази даних. — Режим доступа : http://www.simulation.kiev.ua/dbis/lection27.html. — Дата доступа : 26.04.17.
48. Пользователей интернета пригласили подтвердить теорию Эйнштейна — Режим доступа : http://lenta.ru/news/2005/02/21/einstein/ — Дата доступа : 01.05.17.
49. Розподілені Обчислення — Режим доступа : http://www.simulation.kiev.ua/dbis/lection27.html. — Дата доступа : 01.05.17.
50. Розподілені обчислення в Україні. — Режим доступа : http://kirdey.com/rozpodileni-obchislennya-v-ukrayini — Дата доступа : 25.04.17.
51. Сайт української команди розподілених обчислень. — Режим доступа : http://distributed.org.ua/index.php?newlang=ua. — Дата доступа : 25.04.17.
52. Семчишин Ю. Б. Методи та засоби розподілення обчислень в задачах теплового проектування електронних пристроїв. / Семчишин Ю. Б. - Львів : Львівська політехніка, 2010. - 24 с.