КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій
(факультет)

інформаційних технологій
(кафедра)

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

на тему: «Розподілена обробка даних для великомасштабних обчислень»

ЛИФИРЕНКО РОСТИСЛАВ ОЛЕКСАНДРОВИЧ
(прізвище, ім’я та по батькові студента повністю)

Київ, 2024 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій

(кафедра)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТ
к.т.н., доцент Гончаренко Т.А.

„___” _____2024 року

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

на тему: «Розподілена обробка даних для великомасштабних обчислень»

Виконав: студент 4-го курсу, групи КН-20-2

Спеціальності: 122 «Комп`ютерні науки

Освітня програма: «Інформаційні управляючі системи та технології»

(шифр і назва напряму підготовки, спеціальності)

Лифиренко Р.О.
(прізвище та ініціали)

Керівник к.т.н., доц. Гончаренко Т.А.

(прізвище та ініціали)

Рецензент к.т.н., доц. Шабала Є.Є.
(прізвище та ініціали)

Київ, 2024 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: автоматизації і інформаційних технологій
Кафедра: інформаційних технологій
Освітній рівень: «бакалавр» за ОП
Спеціальність: 122 «Комп`ютерні науки»
Спеціалізація: Інформаційні управляючі системи і технології

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТ
к.т.н., доцент Гончаренко Т.А.

_____
„___” _____2024 року

З А В Д А Н Н Я
ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

Лифиренко Ростислав Олександрович
Тема роботи: Розподілена обробка даних для великомасштабних обчислень

затверджена наказом ректора КНУБА №433/2 від 29.02.2024.
2. Керівник роботи: Гончаренко Тетяна Андріївна, к.т.н, доцент
кафедри інформаційних технологій
3. Строк подання студентом роботи до захисту:
4. Зміст пояснювальної записки за розділами:
Р.1. Аналіз предметної області та постановка задачі
Р.2. Детальний опис розподіленої обробки даних
Р.3. Проектні рішення

5. Інформаційні слайди:
С.1. Характеристики Big Data
С.1. Обчислювальний кластер
С.2. Топологія мереж
С.3. Демонстрація VPN
С.5. Обмін пакетами даних між двома комп’ютерами через Hamachi
С.6. Інформація про клієнтів у терміналі

6. Календарний план виконання атестаційної випускної роботи

Види робіт та їх зміст	Дата виконання
Р. 1. Аналіз предметної області та постановка задачі	Лютий 2024 р.
Р. 2. Детальний опис розподіленої обробки даних	Березень 2024 р.
Р. 3. Проектні рішення	Квітень 2024 р.
Тестовий приклад програми	Квітень 2024 р.
Остаточне оформлення роботи	Травень 2024 р.
Направлення роботи на рецензування	Червень 2024 р.
Попередній захист роботи на кафедрі	Червень 2024 р.

7. Консультанти розділів атестаційної випускної роботи

Розділ	Прізвище, ініціали та посада консультанта, представника комісії	дата	підпис
Ергономіка інформаційних технологій	доц. Ачкасов І.А.		
Прийом програмного продукту	доц. Рябчун Ю.В.		

8. Дата видачі завдання: 29.02.2024

Завідувач _____ Гончаренко Т.А.
(підпис) (прізвище та ініціали)

Керівник _____ Гончаренко Т.А.
(підпис) (прізвище та ініціали)

Здобувач _____ Лифиренко Р.О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Лифиренко Р.О. Розподілена обробка даних для великомасштабних обчислень.

Атестаційна випускна робота бакалавра за спеціальністю: 122 «Комп’ютерні науки», освітня програма: «Інформаційні управляючі системи і технології». – Київський національний університет будівництва та архітектури. – Київ, 2024.

Робота присвячена дослідженню методів та технологій розподіленої обробки даних, які застосовуються для ефективного управління та аналізу великих обсягів інформації. У роботі розглянуто основні концепції розподілених систем, їх архітектурні особливості, а також переваги та недоліки використання таких систем у різних галузях.

Ключові слова:Наука про дані,Великі дані,VPN,JavaScript,HTML,LAN,IP.

SUMMARY

Lyfyrenko. R.O. Distributed data processing for large-scale computing.

Bachelor's thesis in the specialty: 122 "Computer Science", specialization: "Information managing systems and technologies". - Kyiv National University of Construction and Architecture. - Kyiv, 2024.

This work is dedicated to the study of methods and technologies for distributed data processing, which are applied for efficient management and analysis of large volumes of information. The paper examines the main concepts of distributed systems, their architectural features, as well as the advantages and disadvantages of using such systems in various fields.

Keywords:Data Science,Big Data,VPN,JavaScript,HTML,LAN,IP.

ЗМІСТ

Вступ

Розподілена обробка даних є ключовим аспектом сучасних інформаційних технологій, що революціонізує спосіб, яким ми збираємо, зберігаємо, обробляємо і аналізуємо великі обсяги даних. Цей підхід дозволяє розподілено виконувати обчислення на декількох вузлах або комп'ютерах у мережі, замість традиційного централізованого оброблення.

В умовах стрімкого зростання обсягів даних, які генеруються компаніями та організаціями, розподілена обробка стає важливою стратегією для забезпечення швидкодії, масштабованості і надійності систем. Вона дозволяє розподіляти завдання між різними частинами обладнання, що прискорює обробку даних та знижує час відповіді на запити користувачів.

Основні переваги розподіленої обробки даних включають збільшення продуктивності, зменшення витрат на обладнання, збільшення надійності за рахунок дублювання даних та автоматизованого резервного копіювання. Крім того, такий підхід дозволяє легко масштабувати систему з ростом потреб.

Перелік умовних позначень

IDC – International Data Corporation
NIST – National Institute of Standards and Technology
РОД – Розподілена Обробка Даних
PRAM – Parralel Random-Access Machines
CAS – Compare-And-Swap
SLinCA - Scaling Laws in Cluster Aggregation
MultiScaleIVideoP - Multiscale Image and Video Processing
CPDynSG - City Population Dynamics and Sustainable Growth
LAMMPS - Large-scale Atomic/Molecular Massively Parallel Simulator
HDFS - Hadoop Distributed File System
RDD – Resilient Distributed Datasets
CPU – Central Processing Unit
GPU – Graphics Processing Unit
LLR – Lucas–Lehmer–Riesel
AVX - Advanced Vector Extensions
FMA - Fused Multiply-Add
AWS – Amazon Web Services
EMR – Elastic MapReduce
API – Application Programming Interface
ППЗ – Проміжне Програмне Забезпечення
VIA – Virtual Interface Architecture
PCI – Peripheral Component Interconnect
МКІ – Масштабований Координований Інтерфейс
TCP – Transmission Control Protocol
IP – Internet Protocol
SMP – Symmetric MultiProcessing
MPI – Message Passing Interface
NIC – Network Interface Card
КМ – Комп’ютерна Мережа
DNS – Domen Name System
PC – Personal Computer
HTTP – HyperText Transfer Protocol
FTP – File Transfer Protocol
PDU – Protocol Data Unit
RFC – Request for Comments
VPN – Virtual Private Network
LAN – Local Area Network
WAN – Wide Area Network
OSI – Open Systems Interconnection
ISO – International Organization for Standardization
ЛКМ – Локальна Комп’ютерна Мережа
NNTP – Network News Transfer Protocol
SMTP – Simple Mail Transfer Protocol

SNMP – Simple Network Management Protocol
NAT – Network Address Translation
UDP – User Datagram Protocol
JS – JavaScript
DOM – Document Object Model
ПЗ – Програмне Забезпечення
SIMD –Single Instruction, Multiple Data
URL – Uniform Resource Locator

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної області

Предметною областю даної дипломної роботи є наука про дані(Data Science).

Data Science - це галузь, яка зосереджена на використанні аналітичних методів, статистики та машинного навчання для отримання знань та виявлення патернів в даних. Аналізуючи структуровані та неструктуровані дані, Data Science допомагає виготовленні висновків, розробці прогностичних моделей та покращенні прийняття рішень.

Data Science використовується для розгляду питань, таких як рекомендації для користувачів, прогнозування тенденцій у бізнесі, аналізу соціальних мереж та багатьох інших областей. Предметна область Data Science дозволяє виявляти корисні відомості та отримувати цінний інсайт з даних.

Після відомостей, отриманих у Data Science, приходить час розглядати Big Data. Експонентне зростання даних створило нову сферу інтересів у галузі

технологій та бізнесу під назвою "Big Data". Загалом, набір даних або бізнес-проблема належать до класифікації Big Data, коли її дані настільки великі або складні, що їх стає неможливим зберігати, обробляти та аналізувати, використовуючи традиційні підходи до зберігання та аналізу даних.

Скільки даних потрібно, щоб стати Big Data? Чи достатньо 100 терабайт або 1000 петабайт? Обсяг є лише одним із критеріїв, оскільки потреба в обробці даних у режимі реального часу (також це називають даними в русі) або потреба в інтеграції структурованих і неструктурованих даних може кваліфікувати проблему як велику проблему даних. Наприклад, Міжнародна корпорація даних IDC використовує 100 терабайт як розмір набору даних, який визначається як Big Data. Якщо дані потокові, розмір набору даних може бути меншим, ніж 100 терабайт, але все ще вважається Big Data до тих пір, поки дані, що створюються, збільшуються на понад 60% на рік. Згідно NIST: "Парадигма великих даних складається з розподілу систем даних по горизонтально пов'язаних незалежних

ресурсах для досягнення масштабованості, необхідної для ефективної обробки великих наборів даних".

Щоб вирізнити дані від великих даних, використовуються чотири Vs:

1.Об'єм (Volume) - кількість даних, що передаються та зберігаються. Поточним завданням є пошук способів найбільш ефективно обробити зростаючий обсяг даних.

2.Швидкість (Velocity) - швидкість, з якою формуються дані. Наприклад, дані, згенеровані мільярдом акцій, проданих на Нью-Йоркській фондовій біржі, не можуть бути просто збережені для подальшого аналізу.

3.Різноманітність (Variety) - тип даних, який рідко знаходиться в стані, який ідеально готовий до обробки та аналізу. Велика частка Big Data –неструктуровані дані, які, за оцінками, становлять від 70 до 90% світових даних.

4.Достовірність (Veracity) - процес запобігання неточного опису наборів даних. Наприклад, люди можуть створювати онлайн-акаунт та використовувати неправдиву контактну інформацію. Підвищена правдивість у зборі даних зменшує необхідну кількість очищення даних.

Хоча тут перераховано чотири V, більшість дискусій, інструментів та документів стосуються лише перших трьох (об'єм, швидкість, різноманітність).[1]

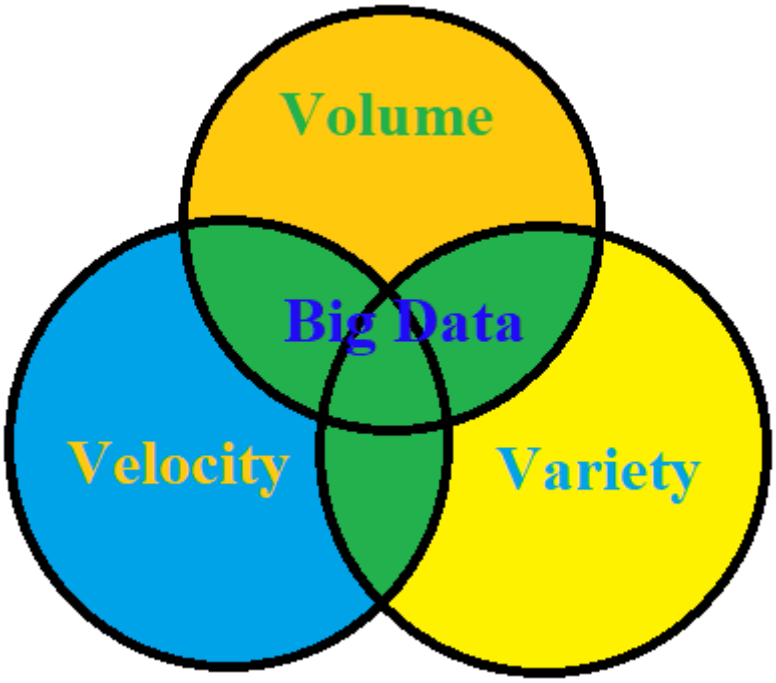

Рисунок 1.1.1 - Характеристики Big Data

1.2 Аналіз об’єкта дослідження

Обробка даних - це процес збору, аналізу, перетворення, інтерпретації та збереження інформації з метою отримання корисних знань, прийняття рішень або виконання певних завдань. Цей процес може включати різноманітні дії, такі як очищення даних від помилок, структурування їх для подальшого аналізу, використання алгоритмів машинного навчання для виявлення закономірностей та здійснення передбачень, а також забезпечення захисту особистих даних. Обробка даних важлива в багатьох галузях, таких як бізнес, наука, медицина, технології та інші.

Основні операції,з обробки даних:

1. Збір даних: Це процес накопичення інформації з різних джерел з метою забезпечення достатньої повноти даних для прийняття рішень. Це може включати збір даних з баз даних, опитування користувачів, моніторинг датчиків тощо.
2. Формалізація даних: Це процес приведення даних з різних джерел до однакової форми, щоб полегшити подальшу обробку та аналіз. Це може включати переведення даних в стандартні формати, стандартизацію одиниць виміру тощо.
3. Фільтрація даних: Це видалення зайвих або несуттєвих даних, які не потрібні для прийняття рішень. Це може включати видалення дублікатів, аномальних значень, а також відсів зайвої інформації.
4. Сортування даних: Це впорядкування даних за певною ознакою для полегшення їхнього подальшого використання. Наприклад, дані можуть сортуватися за алфавітом, числовим значенням, часом тощо.
5. Архівація даних: Це процес збереження даних у зручній та доступній формі для подальшого використання. Це може включати стиснення даних, створення архівних копій, розміщення даних на зовнішніх носіях тощо.
6. Захист даних: Це комплекс дій, спрямованих на запобігання втраті, відтворенню та модифікації даних. Це може включати шифрування

даних, встановлення прав доступу, резервне копіювання даних та інші заходи безпеки.

7. Транспортування даних: Це прийом та передача даних між віддаленими користувачами інформаційного процесу. Джерело даних прийнято називати сервером, а споживача — клієнтом. Це може включати використання мережних протоколів, передачу через інтернет або локальні мережі тощо.

8. Перетворення даних: Це процес перетворення даних з однієї форми в іншу або з однієї структури в іншу, або зміни типу носія. Це може включати конвертацію файлових форматів, перекодування даних з одного типу в інший, зміну структури баз даних тощо.[2]

Розподілені обчислення (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.[3]

Слово “розподілений” у таких термінах, як «розподілена система», «розподілене програмування» та «розподілений алгоритм» спочатку стосувалося комп’ютерних мереж, де окремі комп’ютери були фізично розподілені в певній географічній зоні. Зараз ці терміни використовуються в набагато ширшому значенні, навіть посилаючись на автономні процеси, які виконуються на одному фізичному комп’ютері та взаємодіють один з одним шляхом передачі повідомлень.

Хоча немає єдиного визначення розподіленої системи, зазвичай використовуються наступні властивості визначення:

- Існує кілька автономних обчислювальних об'єктів (комп'ютерів або вузлів), кожен з яких має власну локальну пам'ять.
- Об’єкти спілкуються один з одним шляхом передачі повідомлень.

Розподілена система може мати спільну мету, таку як вирішення великої обчислювальної проблеми; тоді користувач сприймає набір автономних процесорів як одиницю. Крім того, кожен комп’ютер може мати власного користувача з індивідуальними потребами, а мета розподіленої системи полягає в координації використання спільних ресурсів або наданні комунікаційних послуг для користувачів.

Інші типові властивості розподілених систем включають наступне:

- Система повинна терпіти збої в окремих комп'ютерах.
- Структура системи (топологія мережі, затримка мережі, кількість комп'ютерів) невідома заздалегідь, система може складатися з різних типів комп'ютерів і мережевих з'єднань, і система може змінюватися під час виконання розподіленої програми.
- Кожен комп’ютер має лише обмежене, неповне уявлення про систему. Кожен комп'ютер може знати лише одну частину введення.

Для розподілених обчислень використовуються різні апаратні та програмні архітектури. На нижчому рівні необхідно з’єднати кілька процесорів з якоюсь мережею, незалежно від того, чи ця мережа надрукована на друкованій платі, чи складається з слабо з’єднаних пристроїв і кабелів. На більш високому рівні необхідно з’єднати процеси, що виконуються на цих процесорах, з якоюсь системою зв’язку.

Розподілене програмування зазвичай належить до однієї з кількох базових архітектур: клієнт-сервер, трирівнева, n-рівнева або однорангова; або категорії: слабке зчеплення або жорстке зчеплення.

1. Клієнт-сервер: архітектури, де інтелектуальні клієнти звертаються до сервера для отримання даних, а потім форматують і відображають їх користувачам. Вхідні дані клієнта повертаються на сервер, коли вони представляють постійні зміни.

2. Трирівневі: архітектури, які переміщують клієнтський інтелект на середній рівень, щоб можна було використовувати клієнтів без стану. Це спрощує розгортання програми. Більшість веб-додатків є трирівневими.

3. n-tier: архітектури, які зазвичай посилаються на веб-додатки, які далі пересилають свої запити до інших корпоративних служб. Цей тип додатків є найбільш відповідальним за успіх серверів додатків.

4. Одноранговий: архітектури, де немає спеціальних машин, які надають послуги або керують мережевими ресурсами. Натомість усі обов’язки рівномірно розподіляються між усіма машинами, відомими як однорангові. Піри можуть служити як клієнтами, так і серверами. Прикладами такої архітектури є BitTorrent і мережа біткойн.

Іншим основним аспектом розподіленої обчислювальної архітектури є метод обміну даними та координації роботи між паралельними процесами. За допомогою різних протоколів передачі повідомлень процеси можуть спілкуватися безпосередньо один з одним, як правило, у зв’язку основний/підпорядкований. Альтернативно, «орієнтована на базу даних» архітектура може забезпечити розподілене обчислення без будь-якої форми прямого зв’язку між процесами, використовуючи спільну базу даних.Архітектура, орієнтована на базу даних, зокрема, забезпечує аналітику реляційної обробки в схематичній архітектурі, що дозволяє ретранслювати живе середовище. Це дає можливість розподілених обчислювальних функцій як у межах, так і за межами параметрів мережевої бази даних.

Причини використання розподілених систем і розподілених обчислень можуть включати:

1. Сама природа програми може вимагати використання комунікаційної мережі, яка з’єднує кілька комп’ютерів: наприклад, дані створюються в одному фізичному місці, а потрібні в іншому місці.

2. Є багато випадків, коли використання одного комп'ютера було б принципово можливим, але використання розподіленої системи є вигідним з практичних міркувань. Наприклад:

- Він може забезпечити набагато більший обсяг пам’яті та пам’яті, швидші обчислення та вищу пропускну здатність, ніж одна машина.
- Вона може забезпечити більшу надійність, ніж нерозподілена система, оскільки немає єдиної точки відмови. Крім того, розподілену систему може бути легше розширювати та керувати, ніж монолітну однопроцесорну систему.
- Можливо, рентабельніше отримати бажаний рівень продуктивності за допомогою кластера з кількох комп’ютерів низького класу порівняно з одним комп’ютером високого класу.[4]

1.3 Опис предмету дослідження

Розподілені системи — це групи об’єднаних у мережу комп’ютерів, які мають спільну мету для роботи. Терміни «паралельні обчислення», «паралельні обчислення» та «розподілені обчислення» багато в чому збігаються, і чіткої різниці між ними немає. Одну і ту ж систему можна охарактеризувати як «паралельну», так і «розподілену»; процесори в типовій розподіленій системі працюють одночасно і паралельно. Паралельні обчислення можна розглядати як особливо тісно пов’язану форму розподілених обчислень, а розподілені обчислення можна розглядати як слабко пов’язану форму паралельних обчислень. Тим не менш, можна грубо класифікувати паралельні системи як «паралельні» або «розподілені» за такими критеріями:

1. У паралельних обчисленнях усі процесори можуть мати доступ до спільної пам’яті для обміну інформацією між процесорами.

2. У розподілених обчисленнях кожен процесор має власну приватну пам’ять (розподілену пам’ять). Обмін інформацією відбувається шляхом передачі повідомлень між процесорами.

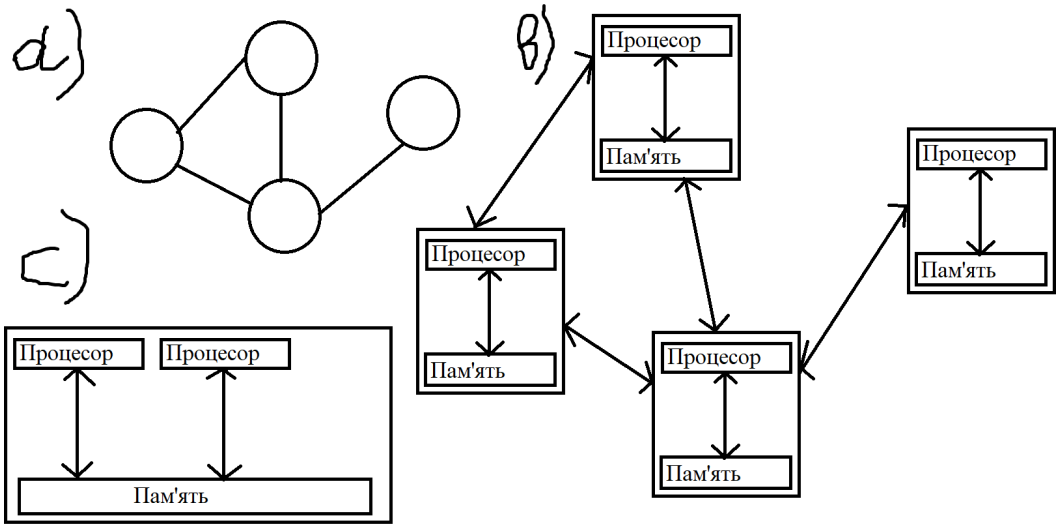

Рисунок 1.3.1- (a), (b): розподілена система; (c): паралельна система

Рисунок 1.3.1 ілюструє різницю між розподіленими та паралельними системами. На малюнку (a) представлено схематичне зображення типової розподіленої системи; система представлена як топологія мережі, в якій кожен вузол є комп'ютером, а кожна лінія, що з'єднує вузли, є лінією зв'язку. На малюнку (b) більш детально показана та сама розподілена система: кожен комп’ютер має власну локальну пам’ять, і обмін інформацією може здійснюватися лише шляхом передачі повідомлень від одного вузла до іншого за допомогою доступних каналів зв’язку. На рисунку (c) показана паралельна система, в якій кожен процесор має прямий доступ до спільної пам'яті.

Ситуація ще більше ускладнюється традиційним використанням термінів «паралельний» і «розподілений алгоритм», які не зовсім відповідають наведеним вище визначенням паралельних і розподілених систем (див. нижче для більш детального обговорення). Тим не менш, як правило, високоефективне паралельне обчислення в мультипроцесорі зі спільною пам'яттю використовує паралельні алгоритми, тоді як координація великомасштабної розподіленої системи використовує розподілені алгоритми.

Багато завдань, які ми хочемо автоматизувати за допомогою комп’ютера, мають тип запитання-відповідь: ми хочемо поставити запитання, а комп’ютер має дати відповідь. У теоретичній інформатиці такі задачі називаються обчислювальними. Формально обчислювальна задача складається з екземплярів разом із рішенням для кожного екземпляра. Приклади — це запитання, які ми можемо поставити, а рішення — бажані відповіді на ці запитання.

Теоретична інформатика прагне зрозуміти, які обчислювальні проблеми можна вирішити за допомогою комп’ютера (теорія обчислюваності) і наскільки ефективно (теорія обчислювальної складності). Традиційно кажуть, що проблему можна розв’язати за допомогою комп’ютера, якщо ми можемо розробити алгоритм, який дає правильне рішення для будь-якого конкретного випадку. Такий алгоритм може бути реалізований як комп’ютерна програма, яка виконується на комп’ютері загального призначення: програма зчитує екземпляр проблеми з вхідних даних, виконує деякі обчислення та створює рішення як вихід. Формалізми, такі як машини з довільним доступом або універсальні машини Тюрінга, можуть бути використані як абстрактні моделі послідовного комп'ютера загального призначення, що виконує такий алгоритм.

Сфера паралельних і розподілених обчислень вивчає схожі питання у випадку або кількох комп’ютерів, або комп’ютера, який виконує мережу взаємодіючих процесів: які обчислювальні проблеми можна вирішити в такій мережі та наскільки ефективно? Однак зовсім не очевидно, що мається на увазі під «вирішенням проблеми» у випадку паралельної або розподіленої системи: наприклад, що є завданням розробника алгоритму, і що є паралельним або розподіленим еквівалентом послідовного комп'ютер загального призначення?

Наведене нижче обговорення зосереджено на випадку кількох комп’ютерів, хоча багато проблем однакові для паралельних процесів, що виконуються на одному комп’ютері.

Зазвичай використовуються три точки зору:

1.Паралельні алгоритми в моделі спільної пам'яті

- Всі процесори мають доступ до спільної пам'яті. Розробник алгоритму вибирає програму, яку виконує кожен процесор.
- Однією з теоретичних моделей є паралельні машини з випадковим доступом PRAM,які використовуються. Однак класична модель PRAM передбачає синхронний доступ до спільної пам'яті.
- Програми зі спільною пам’яттю можна розширити до розподілених систем, якщо базова операційна система інкапсулює зв’язок між вузлами та фактично об’єднує пам’ять у всіх окремих системах.
- Модель, яка ближче до поведінки реальних багатопроцесорних машин і враховує використання машинних інструкцій, таких як CAS — це модель асинхронної спільної пам’яті. Існує велика кількість робіт щодо цієї моделі, короткий виклад яких можна знайти в літературі.

2.Паралельні алгоритми в моделі передачі повідомлень

- Розробник алгоритму вибирає структуру мережі, а також програму, яку виконує кожен комп'ютер.
- Використовуються такі моделі, як булеві схеми та мережі сортування. Булеву схему можна розглядати як комп’ютерну мережу: кожен шлюз — це комп’ютер, на якому виконується надзвичайно проста комп’ютерна програма. Подібним чином мережу сортування можна розглядати як комп’ютерну мережу: кожен компаратор є комп’ютером.

3.Розподілені алгоритми в моделі передачі повідомлень

- Розробник алгоритму лише вибирає комп’ютерну програму. На всіх комп’ютерах працює одна програма. Система повинна працювати коректно незалежно від структури мережі.
- Зазвичай використовуваною моделлю є граф з одним кінцевим автоматом на вузол.

Традиційні обчислювальні задачі розглядають таку перспективу, що користувач задає запитання, комп’ютер (або розподілена система) обробляє запитання, потім дає відповідь і зупиняється. Однак існують також проблеми, де система повинна не зупинятися, включно з проблемою обідаючих філософів та

іншими подібними проблемами взаємного виключення. У цих проблемах розподілена система повинна постійно координувати використання спільних ресурсів, щоб не виникало конфліктів або взаємоблокувань.

Існують також фундаментальні проблеми, які є унікальними для розподілених обчислень, наприклад ті, що стосуються відмовостійкості. Приклади пов’язаних проблем включають проблеми консенсусу, візантійську відмовостійкість і самостабілізацію.

Багато досліджень також спрямовані на розуміння асинхронної природи розподілених систем:

1. Синхронізатори можна використовувати для запуску синхронних алгоритмів в асинхронних системах.
2. Логічні годинники забезпечують причинно-наслідковий зв'язок, що відбувся до того, як упорядковуються події.
3. Алгоритми синхронізації годинника забезпечують глобально послідовні фізичні позначки часу.[4]

1.4 Аналіз актуальності

Багато великих та інноваційних компаній використовують РОД для аналізу великих обсягів інформації та вирішення складних завдань обробки даних. Ось кілька прикладів компаній, які активно використовують РОД:

- **Folding@Home** — проєкт розподілених обчислень, що проводиться під егідою Стенфордського університету. Суть проєкту полягає в моделюванні процесу згортання білків з метою виявлення потенційних помилок у природній конформації. Помилки конформації спричиняють ряд клінічних синдромів, серед яких: хвороба Альцгеймера, хвороба Паркінсона, діабет типу II, хвороба скрепі, Хвороба Кройцфельда—Якоба, коров'ячий сказ, склероз і деякі типи раку. Розуміння механізмів виникнення дефектів на молекулярному рівні допоможе з'ясувати точну картину виникнення даних захворювань і дозволить розробити методи протидії їм.

Коли спалах COVID-19 став пандемією, команда проекту Folding@home переключила свої зусилля на дослідження коронавірусу SARS-CoV-2. Вони розробили спеціалізовані програми для моделювання важливих білкових структур вірусу, зокрема "шипів", які використовуються для взаємодії з клітинами людини.

Ці моделі дозволили дослідникам краще зрозуміти, як вірус взаємодіє з клітинами людини, та ідентифікувати потенційні точки втручання для розвитку ліків та вакцин. Зусилля волонтерів Folding@home допомогли в швидкому розвитку наукових знань про коронавірус і забезпечили важливі дані для розробки противірусних засобів та лікування COVID-19.[5]

- **SLinCA@Home** — дослідницький проєкт, який використовує підключені до Інтернету комп’ютери для проведення досліджень у таких галузях, як фізика та матеріалознавство.

SLinCA@Home базується в Інституті металофізики імені Г. В. Курдюмова (ІМФ) Національної академії наук України (НАНУ) у Києві, столиці України. Він працював на програмній платформі Berkeley Open Infrastructure for Network Computing (BOINC), платформі SZTAKI Desktop Grid і Distributed Computing API (DC-API) від SZTAKI. SLinCA@Home містить декілька наукових додатків, присвячених дослідженню масштабно-інваріантних залежностей в експериментальних даних у фізиці та матеріалознавстві.

Програми SLinCA@Home:

1. **SLinCA** - перша програма, портована на DG інфраструктуру Лабораторії фізики деформаційних процесів ІМФ НАН України, призначена для виявлення законів масштабної інваріантності в агрегації мономерів у кластерах різних видів та галузях науки. Процеси агрегації кластерів досліджуються в матеріалознавстві, біології, соціології тощо. Теорії пояснюють формування ієрархічних структур та їх масштабоінваріантні властивості. Для перевірки таких теорій необхідні потужні обчислювальні ресурси, і SLinCA використовує IPO для швидкого моделювання різних сценаріїв.

2. Програма **MultiScaleIVideoP** призначена для обробки записаної еволюції матеріалів під час механічної деформації. Ці розрахунки включають безліч параметрів фізичного процесу та обробки зображення, що робить їх трудомісткими і повільними. Використання інфраструктури розподілених обчислень дозволяє прискорити цей процес, обробляючи зображення та відео в більш широкому діапазоні масштабів та за коротший час.

3. **CPDynSG** - це програма, яка призначена для аналізу та прогнозування динаміки населення у містах з метою забезпечення стійкого зростання. Програма використовує великий обсяг експериментальних даних про розмір та розподіл населення в різних місцевостях і порівнює їх з різними теоріями, які пояснюють динаміку міського населення.

4. **LAMMPS** - це некомерційний пакет з відкритим кодом, розроблений Sandia National Laboratories, призначений для молекулярно-динамічного моделювання процесів нановиробництва. Програма використовується для ретельного вибору і налаштування критичних параметрів атомної самоорганізації в розроблюваних моделях і структурах для наномасштабних функціональних пристроїв.[6]

- **PrimeGrid** — проєкт добровільних розподілених обчислень на платформі BOINC і PRPNet, метою якого є пошук дуже великих простих чисел різного виду, водночас прагнучи вирішити давні математичні гіпотези. PrimeGrid пропонує низку підпроєктів з відсіву й пошуку простих чисел. Більшість з них доступні через клієнт BOINC, у якому повністю автоматизовано завантаження, обробку й повернення результатів. Решта проєктів доступні через клієнт PRPNet, вони потребують запуску вручну. Є також активності, що зараховуються в PRPNet, які потребують завантаження, запуску і вивантаження результатів вручну. Різні підпроєкти можуть бути запущені на різноманітних операційних системах, є підпроєкти, що можуть бути виконанні із застосуванням CPU, GPU, або

CPU та GPU одночасно. Під час виконання тестів LLR CPU з набором інструкцій AVX і FMA дають кращий результат без використання GPU.

PrimeGrid досягнув кількох значних результатів в пошуку простих чисел та інших числових структурах. Деякі з найважливіших досягнень включають:

1. Знаходження великих простих чисел: PrimeGrid знайшов багато найбільших відомих простих чисел, таких як мерсенні та факторіалізовані числа, відомі як супер-прості числа.
2. Пошук великих відомих шаблонів простих чисел: Крім простих чисел, PrimeGrid також виявив великі патерни у розподілі простих чисел, допомагаючи краще зрозуміти їхню розподіленість.
3. Розширення знань про числа Ферма: PrimeGrid внесла значний внесок у пошук чисел Ферма, які є потенційно простими числами у вигляді:$2^{2(\hat{}n)}+1$.
4. Виявлення нових рекордів обчислення простих чисел: Проект також встановив деякі рекорди у швидкості обчислення простих чисел за допомогою різних методів та алгоритмів.[7]

1.5 Стан вже існуючих рішень

На сьогоднішній день існує безліч рішень для розподіленої обробки даних, серед яких найпопулярніші відкриті та комерційні платформи. Ось деякі з них:

Apache Hadoop:

- Apache Hadoop - це відкритий фреймворк для обробки великих обсягів даних, що дозволяє розподілену обробку та збереження даних на кластерах серверів.
- Основними компонентами Hadoop є HDFS для збереження даних та MapReduce для обробки даних у розподіленому середовищі.
- Крім того, у екосистемі Hadoop є також інші корисні інструменти, такі як Apache Hive для роботи з даними у вигляді таблиць, Apache Pig для складання скриптів обробки даних та Apache Spark для швидкого обчислення.[8]

Apache Spark:

- Apache Spark - це швидкодіючий фреймворк для обробки даних у реальному часі та пакетної обробки.
- Він надає API для роботи з даними у вигляді RDD, DataFrame та Dataset, що дозволяє розробникам працювати з даними у зручному форматі.
- Spark може працювати з різними джерелами даних, такими як HDFS, Amazon S3, Apache Kafka тощо.
- Окрім того, Spark пропонує багато високорівневих бібліотек для обробки даних, машинного навчання та глибинного навчання, таких як Spark SQL, MLlib та GraphX.[9]

AWS EMR:

- Amazon EMR - це керована служба на базі хмарних ресурсів Amazon Web Services для розподіленої обробки даних.
- EMR надає можливості для швидкого створення та масштабування кластерів Hadoop, Spark, HBase, Flink тощо.
- Користувачам не потрібно вести процес налаштування та керування інфраструктурою, оскільки Amazon бере на себе ці завдання.
- EMR може інтегруватися з іншими службами AWS, такими як Amazon S3, Amazon Redshift тощо, для обробки та аналізу даних з різних джерел.[10]

Google Cloud Dataflow:

- Google Cloud Dataflow - це керована служба для розподіленої обробки даних, яка пропонує можливості для створення потокових та пакетних обробок даних.
- Dataflow надає можливості для автоматичного масштабування залежно від потреб користувача.
- Користувачі можуть розробляти програми обробки даних з використанням Apache Beam API, яке підтримує різні мови програмування, такі як Java та Python.[11]

Apache Flink:

- Apache Flink - це потужний фреймворк для обробки потокових даних, який пропонує високу швидкість обробки та підтримку складних операцій з даними.
- Flink може працювати з різними джерелами даних, включаючи Kafka, HDFS, Amazon S3 тощо.
- Цей фреймворк також має багато вбудованих бібліотек для розробки потокових програм, машинного навчання та аналізу графів.[12]

1.6 Визначення цілей дослідження та постановка задачі

Основна мета проєкту, полягає, у створенні програми великомасштабного обчислення для розподіленої обробки даних, яка оптимально використовує два пристрої (комп'ютери) для збільшення продуктивності та швидкості обчислень.

1.6.1 Дерево цілей

1.Проектування (рис.1.6.1):

1.1.Розробити архітектуру програмного рішення для великомасштабного обчислення з використанням розподіленої обробки даних.

1.2.Визначити протоколи комунікації між пристроями та способи синхронізації обчислень.

2.Реалізація:

2.1.Розробити програмний код для розподіленої обробки даних, використовуючи обрану архітектуру та протоколи комунікації.

2.2.Забезпечити ефективну роботу програми на двох пристроях з можливістю паралельних обчислень та оптимізації ресурсів.

3.Відмітки прогресу:

3.1.Відстеження прогресу на кожному етапі розробки.

3.2.Періодичні звіти та оновлення з команди проекту.

4.Тестування:

4.1.Провести тестування програмного продукту для впевненості у його коректності та ефективності.

4.2.Перевірити відповідність результатів реальним очікуванням та специфікаціям проекту.

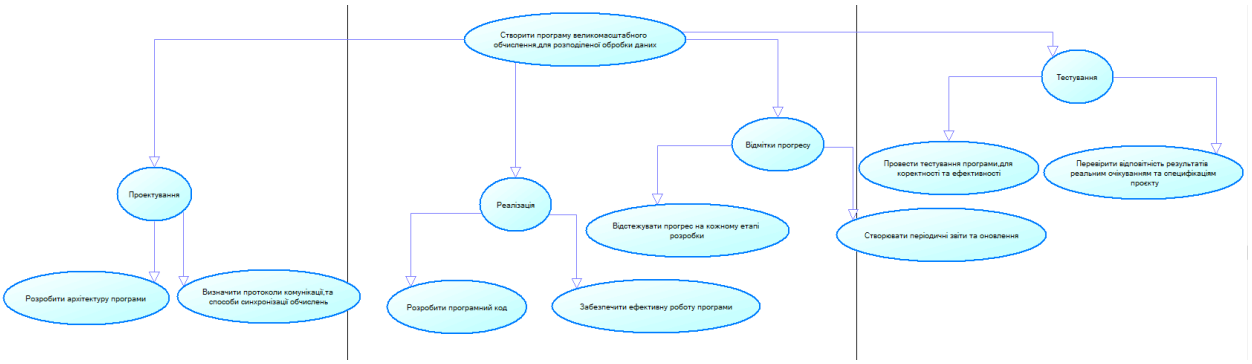

Рисунок 1.6.1 - Дерево цілей створення програми розподіленої обробки даних

Об'єктом дослідження є процес розподіленої обробки даних. В цьому контексті я досліджую метод, алгоритм, технологію та стратегію, яку буду використовувати для ефективної обробки великих обсягів даних на розподілених системах. Основними аспектами дослідження, є розробка архітектури системи, вибір оптимальних алгоритмів обробки даних та оптимізація продуктивності системи.

Розподілена система, як **предмет дослідження** та реалізації, буде включати в себе розробку та налагодження комплексу програмних і апаратних засобів, які забезпечують розподілену обробку даних. З, ключових аспектів,є:архітектура системи,програмний код,оптимізація ресурсів,та масштабованість.

Системою є комплекс технологій, інструментів та методів, які використовуються для реалізації розподіленої обробки даних. Це, включає в себе фреймворки та платформи для обробки даних, бази даних, системи кластерного обчислення, а також інші компоненти, які необхідні для побудови та функціонування вашої системи.

Локальна мережа виступає як **зовнішнє середовище** для дослідження та реалізації системи розподіленої обробки даних. Це включає встановлення фізичного та програмного забезпечення, налаштування мережі, забезпечення

безпеки та моніторингу. Ця локальна інфраструктура дозволяє ефективно тестувати та вдосконалювати систему перед її розгортанням у реальних умовах.

2. ДЕТАЛЬНИЙ ОПИС РОЗПОДІЛЕНОЇ ОБРОБКИ ДАНИХ

2.1 Мережеве з’єднання

Мережеве з'єднання - це процес або стан, у якому комп'ютери, пристрої або системи спільно використовують ресурси та обмінюються даними між собою через мережу зв'язку. Це включає встановлення зв'язку між пристроями за допомогою фізичних або бездротових засобів передачі даних, налаштування протоколів комунікації та обмін інформацією, що дозволяє взаємодіяти та спільно працювати пристроям у мережі. Мережеве з'єднання є основою для спільного використання ресурсів, передачі даних, комунікації та реалізації різноманітних мережевих послуг та додатків.

2.1.1 Концепція мережевого з'єднання/Інтернет(рис 2.2.1.1)

- Спочатку було реалізовано у Мережі агентства з передових досліджень оборони (ARPANET) у 1966 році в США.
- Складається з підключення кількох комп'ютерних мереж, що базуються на різних протоколах.
- Вимагає визначення спільного протоколу з'єднання поверх локальних протоколів.
- Роль цього відіграє Протокол Інтернету (IP), визначаючи унікальні адреси для мережі та комп'ютера-хоста.[13]

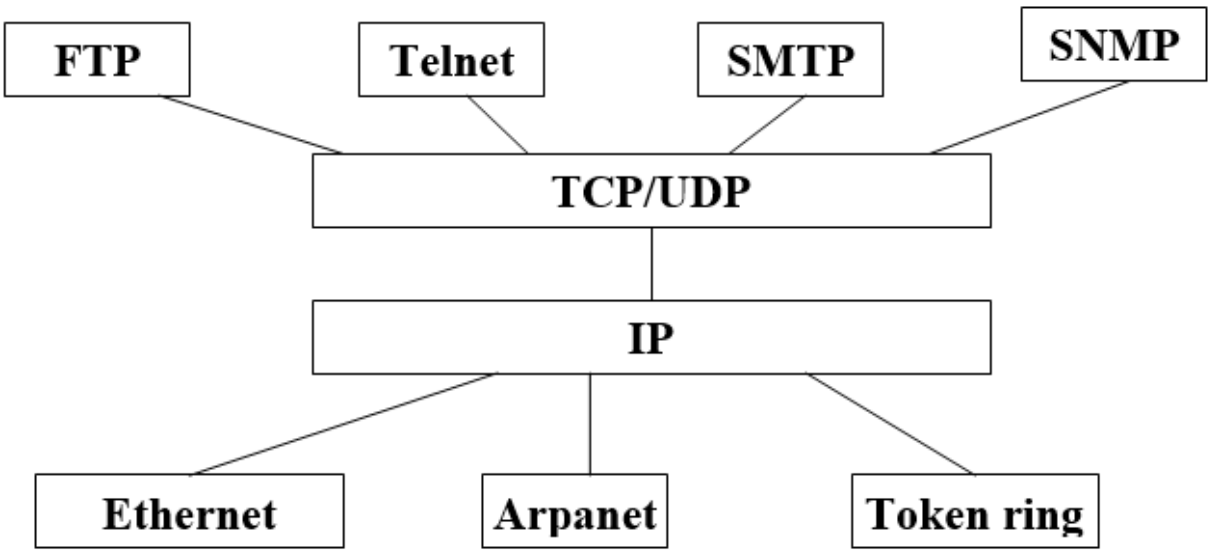

Рисунок 2.2.1.1 – Концепція мережевого з’єднання

Мережеве з'єднання може значно полегшити розподілену обробку даних, що є однією з ключових переваг мережевих систем. Ось декілька способів, як мережеве з'єднання допомагає в цьому:

1. Обмін ресурсами: У мережевій системі різні пристрої можуть обмінюватися даними та ресурсами, такими як обчислювальна потужність, пам'ять, зберігання даних тощо. Це дозволяє розподіляти завдання обробки даних між різними пристроями в мережі для ефективного використання ресурсів.

2. Паралельна обробка: Мережеве з'єднання дозволяє використовувати паралельну обробку даних, коли різні пристрої в мережі можуть обробляти частини даних одночасно. Це збільшує швидкодію та ефективність обробки, оскільки завдання може бути розподілене між багатьма пристроями.

3. Доступ до великих об'ємів даних: Мережеве з'єднання дозволяє звертатися до великих об'ємів даних, які можуть бути збережені на різних пристроях у мережі. Це дає можливість використовувати дані з різних джерел для обробки та аналізу.

4. Резервне копіювання та відновлення: У мережевій системі дані можуть бути резервно копійовані на різні пристрої, що забезпечує захист від втрати даних в разі збою на одному з пристроїв. Це важливо для забезпечення надійності та цілісності даних у розподілених системах.

2.2 GRID - система

Система сіток (Grid system): Це система, що складається з сітки ліній або точок, яка використовується для розміщення та організації елементів (рис. 2.2.1).

Ґрід — це географічно розподілена інфраструктура, яка об'єднує множину різних типів, доступ до яких користувач може отримати з будь-якої точки, незалежно від місця їх розміщення. Ґрід надає колективний розподілений режим доступу до ресурсів і до зв'язаних з ними послуг в рамках глобально-розподілених

організацій (підприємства які спільно використовують глобальні ресурси, бази даних, спеціалізоване програмне забезпечення).[14]

Особливості GRID-системи:

- Масштабованість: GRID-системи можуть легко масштабуватися за рахунок додавання нових вузлів до мережі.
- Відмовостійкість: Завдяки розподіленим ресурсам, GRID-системи можуть продовжувати працювати, навіть якщо один або кілька вузлів виявляться недоступними.
- Висока продуктивність: Завдяки паралельним обчисленням та розподіленню завдань, GRID-системи забезпечують високу швидкість обробки даних.

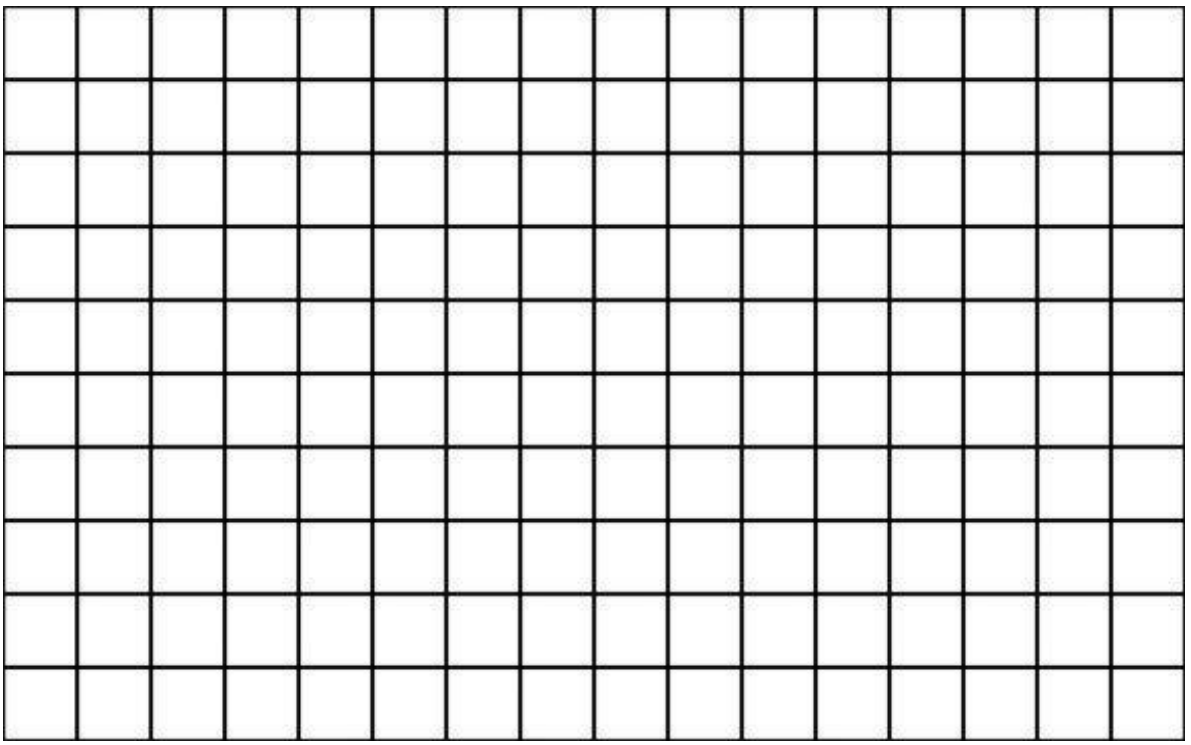

Рисунок 2.2.1 – Модель сітки

2.2.1 Елементи GRID-інфраструктури

Апаратне забезпечення/Ресурси:

- Постачаються з різних географічно розподілених місць.
- Центральний процесор (CPU)/Пам'ять/Інструменти/Бази даних.

Програмне забезпечення:

- Щось, що поєднує всі ці ресурси: проміжне програмне забезпечення (middleware).
- Деякі програми для використання обчислювальної потужності, доступної для використання.[15]

ППЗ — в інформатиці, шар програмного забезпечення, що складається з агентів, які є посередниками між різними компонентами великого застосунка. Найчастіше проміжне ПЗ використовується в розподілених застосунках, причому агентів, що становлять цей шар, може бути кілька.[16]

Таблиця 2.2.1.1

Категорії IT-ресурсів,та послуг

Інструменти та програми	Користувацькі додатки
Посередництво в каталогах, діагностика та моніторинг	Колективні послуги
Безпечний доступ до ресурсів і послуг	Протоколи ресурсів і підключення
Різноманітні ресурси, такі як комп’ютери, носії інформації, мережі та датчики	Фабрика

Основні елементи проміжного програмного забезпечення системи-сітки:

- Безпека: Цей елемент відповідає за захист інформації та ресурсів, що перебувають у системі сітки. Він забезпечує аутентифікацію, авторизацію та конфіденційність даних, що передаються і оброблюються. Це може включати застосування шифрування, аутентифікаційних протоколів та засобів контролю доступу.
- Управління ресурсами: Цей елемент відповідає за ефективне розподілення та використання ресурсів, доступних у грід-системі. Він включає механізми для моніторингу, планування та керування використанням обчислювальних, мережевих та сховищ даних, що складають інфраструктуру сітки.
- Управління даними: Цей елемент забезпечує ефективне зберігання, передачу та обробку даних у грід-системі. Він включає

механізми для реплікації, розподіленого зберігання, синхронізації та управління версіями даних.

- Інформаційні сервіси: Цей елемент надає інформацію про ресурси та стан системи, необхідну для ефективного використання грід-інфраструктури. Він може включати каталоги ресурсів, метадані, сервіси відкритого доступу до даних та інші інструменти для пошуку, аналізу та моніторингу ресурсів.[15]

2.3 Обчислювальний кластер

Кластер (скупчення, рій) - об'єднання кількох однорідних елементів, яке може розглядатися як самостійна одиниця, що має певні властивості.(рис. 2.3.1)

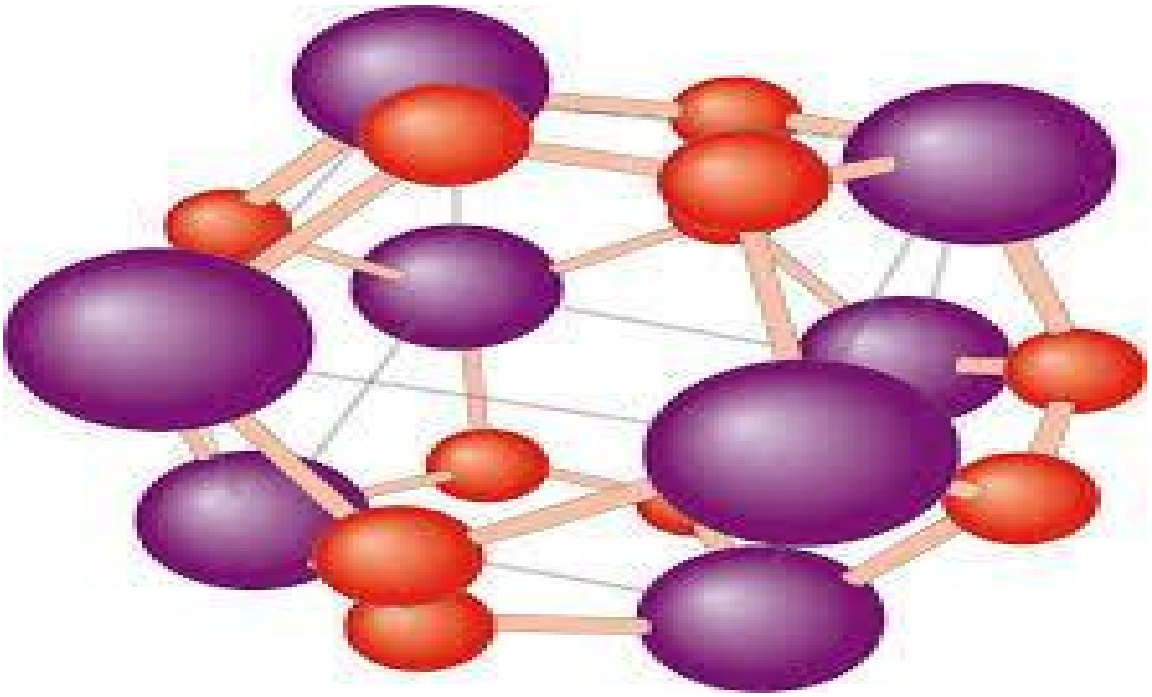

Рисунок 2.3.1 – Кластер

Кластер — група комп'ютерів, об'єднаних високошвидкісними каналами зв'язку, і що представляє з точки зору користувача єдиний апаратний ресурс.[17]

Особливості обчислювального кластера:

- Локальність: У кластері всі комп'ютери фізично розташовані поруч один з одним, що сприяє швидкому обміну даними та зменшенню затримок.

- Висока продуктивність: Обчислювальні кластери забезпечують високу продуктивність завдяки паралельним обчисленням та розподіленню завдань між вузлами.
- Легкість управління: Управління кластером зазвичай є простим завдяки спеціалізованим програмним забезпеченням для керування кластером.

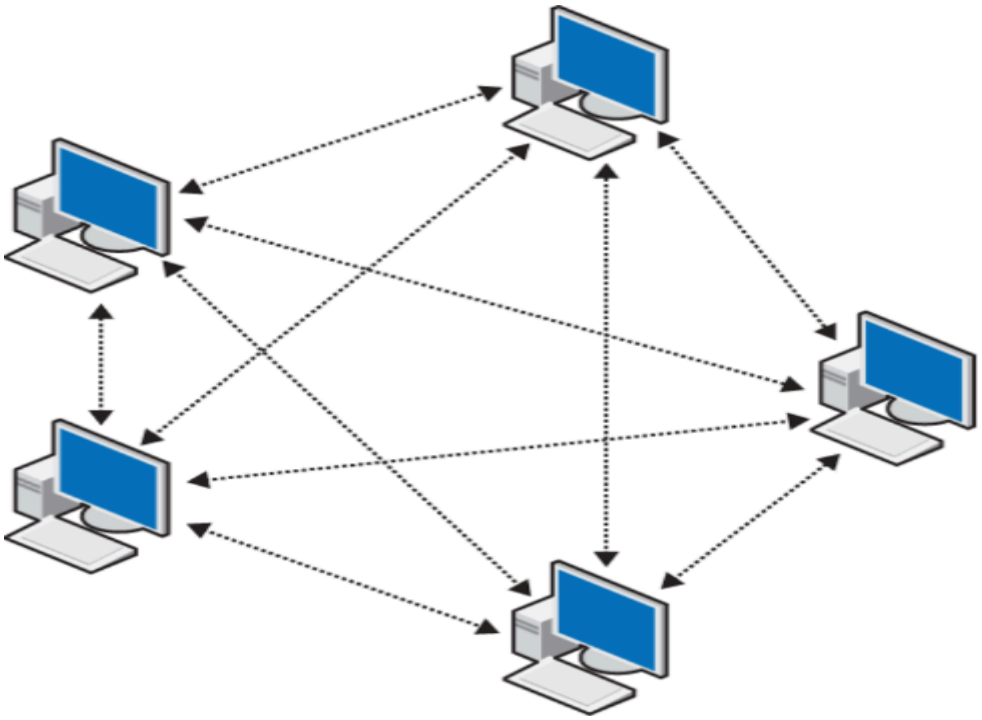

Рисунок 2.3.2- Обчислювальний кластер

Типова архітектура кластера показана на рисунку 2.3.2. Основні компоненти кластера включають в себе кілька автономних комп'ютерів (ПК, робочі станції або SMP), операційні системи, високопродуктивні міжкомп'ютерні зв'язки, проміжне програмне забезпечення, середовища паралельного програмування та додатки. У решті цієї розділу акцентується на компонентах, специфічних для кластера, їх функціональності разом із представниками. Передбачається, що читач знає стандартні апаратні та програмні компоненти, такі як автономні комп'ютери, операційні системи, такі як Linux і Windows, та стандартне програмне забезпечення для зв'язку, таке як TCP/IP.

Кластери повинні використовувати швидкі технології міжз'єднання, щоб підтримувати високосмугову та низьколатентну міжпроцесорну комунікацію між

вузлами кластера. Повільні технології міжз'єднання завжди були критичним фактором обмеження продуктивності для кластерних обчислень. Сьогодні вдосконалені мережеві технології допомагають зреалізувати побудову більш ефективних кластерів.

Вибір технології мережевого з'єднання кластера залежить від кількох факторів, таких як сумісність з апаратним забезпеченням та операційною системою кластера, ціна та продуктивність. Існують дві метрики для вимірювання продуктивності мережевих з'єднань: пропускна здатність та латентність. Пропускна здатність - це кількість даних, які можуть бути передані по мережевому з'єднанню протягом фіксованого періоду часу, тоді як латентність - це час підготовки та передачі даних від вихідного вузла до призначеного вузла.

Таблиця 2.3.1

Приклади деяких технологій взаємозв'язку

Технологія міжз’єднання	Опис
Gigabit Ethernet	Надає досить велику пропускну здатність за невелику ціну, але має відносно високу затримку, що обмежує Gigabit Ethernet як хороший вибір. Однак низька ціна Gigabit Ethernet приваблива для створення кластерів.
Giganet cLAN	Giganet cLAN розроблено з метою підтримки VIA у апаратному забезпеченні та має низьку затримку. Проте він надає лише низьку пропускну здатність менше 125 Мбайт/с, тому він не є життєздатним вибором для реалізації швидких кластерних мереж.
Infiniband	Останній промисловий стандарт, заснований на концепціях VIA і випущений у 2002 році, Infiniband підтримує підключення різних компонентів системи всередині системи, таких як мережі міжпроцесорного зв'язку, підсистеми введення/виведення або мережеві комутатори з кількома протоколами.

	Це робить Infiniband незалежним від будь-якої конкретної технології.
Myrinet	Найбільш широко використовуваний наразі для швидких кластерних мереж. Головною перевагою Myrinet є те, що він працює в просторі користувача, тим самим обходячи втручання та затримки операційної системи.
QsNet II	Наступне покоління версії QsNet, яке базується на інтерфейсі високої продуктивності PCI-X, порівняно з PCI-інтерфейсом QsNet. QsNetII може досягати швидкості 1064 Мбайт/с, підтримувати 4096 вузлів та забезпечувати 64-бітну архітектуру віртуальних адрес.
МКІ	Перший стандарт технології міжз'єднання, визначений для кластерного обчислення. МКІ визначає схему кешування на основі каталогів, яка може забезпечити збереження кешів підключених процесорів у співвідношенні, тим самим здатна реалізувати віртуальну спільну пам'ять.

Таблиця 2.3.1 надає короткий огляд деяких технологій міжз'єднання, які потім порівнюються, як показано в таблиці 2.3.2. У порівняннях розглядаються такі фактори, як: пропускна здатність, затримка, наявність обладнання, підтримка для Linux, максимальна кількість вузлів кластера, яку підтримується, як протокол реалізований, підтримка VIA та підтримка MPI. VIA є стандартом для програмного інтерфейсу низької затримки, який був розроблений консорціумом виробників обладнання та академічних установ і був прийнятий для використання більшістю виробників кластерів. MPI забезпечує обмін повідомленнями через набір бібліотек, які користувачі можуть використовувати для розробки паралельних та розподілених програм. Це означає, що MPI надає шар зв'язку для користувацьких програм і тим самим забезпечує переносимість коду програм між усіма розподіленими та паралельними платформами.

З врахуванням поточної популярності кластерного обчислення стає все важливіше розуміти можливості та потенційну продуктивність різних мережевих міжз'єднань для кластерів. Крім того, через низьку вартість кластерів та їх зростаючу популярність в науковій спільноті багато останніх збирачів кластерів не є комп'ютерними вченими або інженерами і, отже, мають обмежені технічні обчислювальні навички. Ця нова група збирачів кластерів менше зацікавлена у таких функціях, як програмованість мережевих інтерфейсних карт (NIC) та спеціальні бібліотеки обміну повідомленнями. Замість цього, вони цікавляться двома основними факторами: вартість та продуктивність. Хоча вартість легко визначається та порівнюється, продуктивність важко оцінити, особливо для користувачів, які можуть бути новими у кластерному обчисленні.[18]

Відмінності між GRID-системою та обчислювальним кластером в основному полягають у масштабованості, локалізації та способах управління. GRID-системи зазвичай більш гнучкі та масштабовані, в той час як обчислювальні кластери можуть бути більш простими в управлінні та легше підтримувати локально.

Таблиця 2.3.2

Порівняння деяких технологій взаємозв'язку

Критерії порівняння	Gigabit Ethernet	Giganet cLAN	Infiniband	Myrinet	QsNet II	MKI
Пропускна здатність (Мбайт/с)	< 100	< 125	850	230	1064	< 320
Затримка (мкс)	< 100	7 - 10	< 7	10	< 3	1 - 2
Наявність обладнання	Має	Має	Має	Має	Має	Має
Підтримка Linux	Має	Має	Має	Має	Має	Має
Макс. Кількість вузлів	1000	1000	> 1000	1000	4096	1000
Реалізація протоколу	Обладнання	Прошивка на адаптері	Обладнання	Прошивка на адаптері	Прошивка на адаптері	Прошивка на адаптері
Архітектура віртуального інтерфейсу (VIA)	NT/Linux	NT/Linux	Програмне забезпечення	Linux	Немає	Програмне забезпечення
Інтерфейс передачі повідомлень	MVICH через M-VIA, TCP	3rd Party	MPI/Pro	3rd Party	Quadrics	3rd Party

2.4 Адреса мереж

- Мережу можна визначити як групу комп'ютерів та інших пристроїв, що з'єднані якимось чином, щоб мати можливість обмінюватися даними.

- Кожен пристрій в мережі можна розглядати як вузол; кожен вузол має унікальну адресу.
- Адреси - це числові величини, з якими комп'ютерам легко працювати, але не так просто запам'ятати людині.
- Приклад: 204.160.241.98
- Деякі мережі також надають назви, які людина може легше запам'ятати, ніж числа.
- Приклад: www.javasoft.com, що відповідає вищезазначеній числовій адресі.

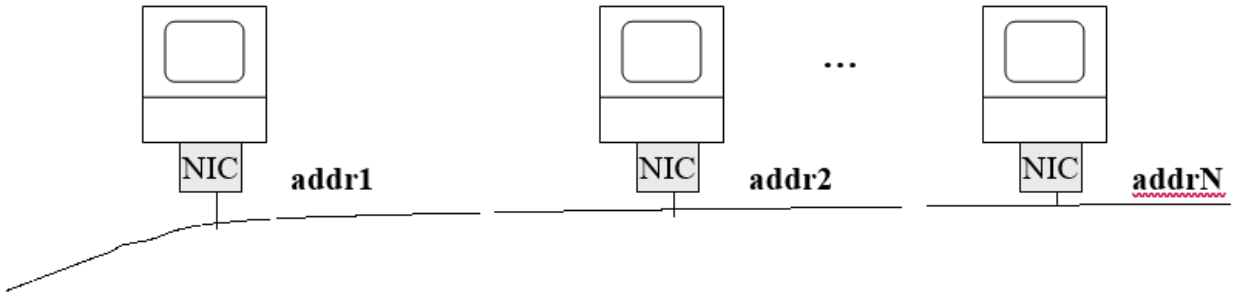

Рисунок 2.4.1 – Адреса мереж,через NIC

Інтернет-адреса складається з 4 байтів, розділених крапками.

Наприклад: 136.102.233.49.

- Перші R байтів (де R = 1, 2, 3) відповідають мережевій адресі;
- Решта H байтів (де H = 3, 2, 1) використовуються для хост-машини.
- InterNIC Register - це організація, відповідальна за розподіл діапазонів адрес, що відповідають мережам. (рис. 2.4.1)
- Критерії, які враховуються:
  - Географічна область (країна);
  - Організація, підприємство;
  - Відділ;
  - Хост.

Система доменних імен (DNS):

- Забезпечує надання мнемонічних текстових адрес для полегшення маніпуляції інтернет-адресами.
- Сервери DNS відповідають за перетворення мнемонічних текстових інтернет-адрес в тверді числові інтернет-адреси.[13]

2.5 Порти

- IP-адреса ідентифікують хост-машину в Інтернеті.
- IP-порт ідентифікує конкретний додаток, що працює на Інтернет-хост-машині.
- Порт ідентифікується числом, номером порта. (рис. 2.5.1)
- Кількість портів не має функціональних обмежень, на відміну від послідовних комунікацій, де дозволено лише 4 порти.
- Існують деякі номери портів, які відведені для конкретних додатків.[13]

Applications	Port numbers
HTTP	80
FTP	20 and 21
Gopher	70
SMTP (e-mail)	25
POP3 (e-mail)	110
Telnet	23
Finger	79

Рисунок 2.5.1 – Додатки,та їхні порти

2.6 Передача даних

- У сучасних мережах дані передаються за допомогою комутації пакетів.
- Повідомлення розбиваються на одиниці, які називаються пакетами, і відправляються з одного комп'ютера на інший.
- На призначенні дані вилучаються з одного або кількох пакетів і використовуються для відновлення початкового повідомлення.
- Кожен пакет має максимальний розмір і складається з заголовка та області даних.
- Заголовок містить адреси вихідного та призначеного комп'ютерів, а також інформацію про послідовність, необхідну для збирання повідомлення на призначенні[13] (рис. 2.6.1).

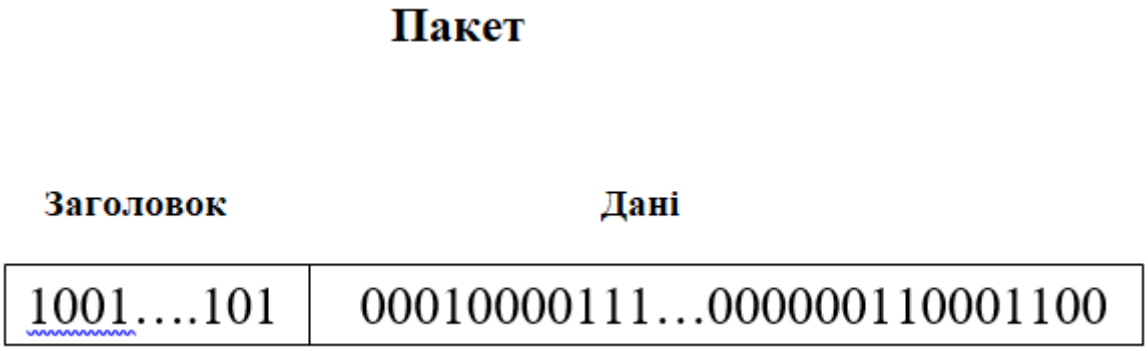

Рисунок 2.6.1 – Представлення пакету даних

2.7 LAN

ЛКМ (LAN) є об'єднанням певного числа комп'ютерів на відносно невеликій території. В порівнянні з глобальною мережею (WAN), локальна мережа зазвичай має більшу швидкість обміну даними, менше географічне покриття та відсутність потреби використовувати запозичену телекомунікаційну лінію зв'язку.

До складу локальної мережі входять:

1. Комп'ютери
2. Мережеві адаптери
3. Периферійні пристрої
4. Передавальне середовище
5. Мережеві пристрої

За допомогою локальної мережі один комп'ютер отримує доступ до ресурсів іншого, таких, як дані та периферійні пристрої (принтери, модеми, факси тощо). Використання комп'ютерних мереж дає можливість розподілу ресурсів великої вартості, покращення доступу до інформації, виконувати швидке та якісне прийняття рішень. Прикладом застосування цієї технології може бути E-mail.

Перевагами об'єднання комп'ютерів у локальну мережу є:

1. Розподіл даних (Data Sharing). Дані в мережі зберігаються на центральному PC та можуть бути доступні для будь-якого PC, підключеного до мережі, тому не потрібно на кожному робочому місці мати накопичувач для зберігання однієї й тієї ж інформації.
2. Розподіл ресурсів (Resource Sharing). Периферійні пристрої можуть бути доступні для всіх користувачів мережі (наприклад, факс або лазерний принтер).
3. Розподіл програм (Software Sharing). Усі користувачі мережі можуть мати доступ до програм, які були один раз централізовано встановлені. При цьому повинна працювати мережна версія відповідних програм.
4. Електронна пошта (Electronic Mail). Усі користувачі мережі можуть передавати або приймати повідомлення.

У локальних мережах застосовують такі мережеві пристрої:

1. Концентратори
2. Комутатори
3. Повторювачі
4. Мости
5. Маршрутизатори

Локальні мережі вирішують такі задачі:

1. Радіус дії обмежується невеликими географічними відстанями.
2. Надає множинний доступ до спільного передавального середовища.
3. Права користувача надаються локальним адміністратором.

4. Надає постійний доступ до сервісів локальної мережі.
5. Фізично з'єднує пристрої на невеликій відстані.[18]

2.8 Топологія мереж

Загальне поняття топології мереж означає схему розміщення і з'єднання вузлів без врахування їх фізичних розмірів. Для топології комп'ютерних мереж крім того не має значення територіальне розміщення комп'ютерів, а враховується лише конфігурація їх логічних зв'язків. Топологію комп'ютерної мережі слід розуміти, як схему з'єднання вузлів без врахування відстані між ними і їх територіального розміщення.

Крім поняття топології КМ існує поняття, топології фізичних зв'язків КМ, куди включають у якості вузлів не тільки комп'ютери, але й з'єднувальне обладнання, наприклад, комутатори або концентратори. При цьому територіальне розміщення обладнання також не враховується. Головною ознакою, за якою легко відрізнити вузол КМ від з'єднувального обладнання, є наявність унікальної адреси, без котрої вузол не здатен забезпечити обмін даними. (рис. 2.8.1)

1.Шина (Bus Topology):

Опис: У цій топології всі пристрої підключені до одного спільного кабелю. Кожен пристрій підключений до цього кабелю через розгалужувачі або концентратори.

Основні властивості:

- Простота в установці і налаштуванні.
- Висока вартість експлуатації при збільшенні кількості пристроїв.
- Загальний канал передачі даних, тому конфлікти можуть виникати, коли багато пристроїв намагаються передавати дані одночасно.
- Вразливість до збоїв: якщо кабель перерваний або пошкоджений в одному місці, це може призвести до втрати зв'язку з усіма пристроями, які підключені до цього кабелю.[19]

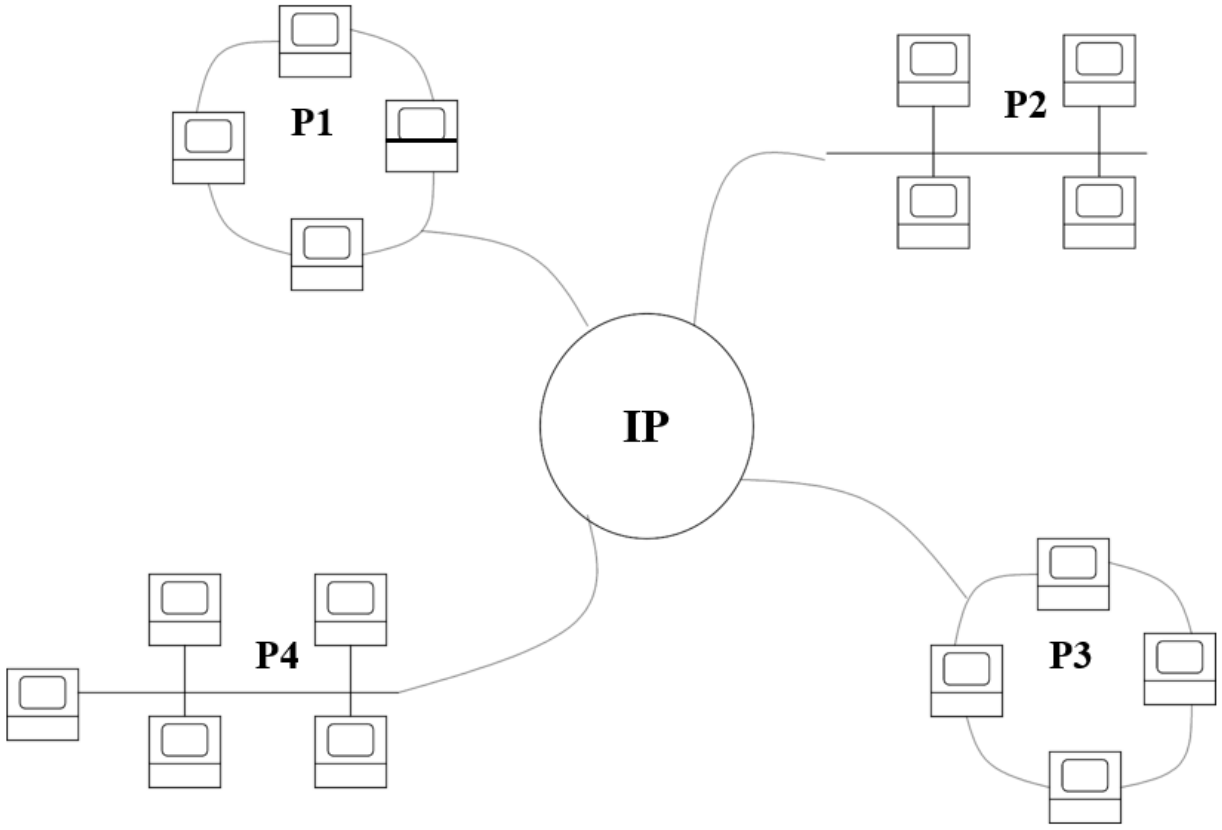

Рисунок 2.8.1 – Топологія мереж

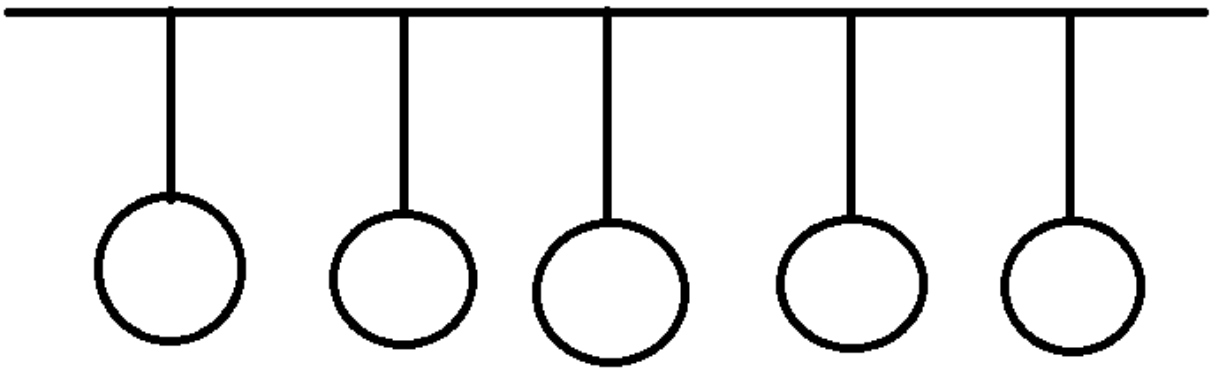

Рисунок 2.8.1 – Bus Topology

2.Кільцева (Ring Topology):

Опис: У цій топології пристрої підключені один до одного в формі кільця. Кожен пристрій підключений до двох сусідів, утворюючи кільце.

Основні властивості:

- Простота в установці.
- Ефективне використання пропускної здатності мережі.
- Вразливість до збоїв: якщо один пристрій або кабель у кільці пошкоджений, це може призвести до втрати зв'язку усієї мережі.
- Повільна швидкість передачі даних у порівнянні з іншими топологіями.[20]

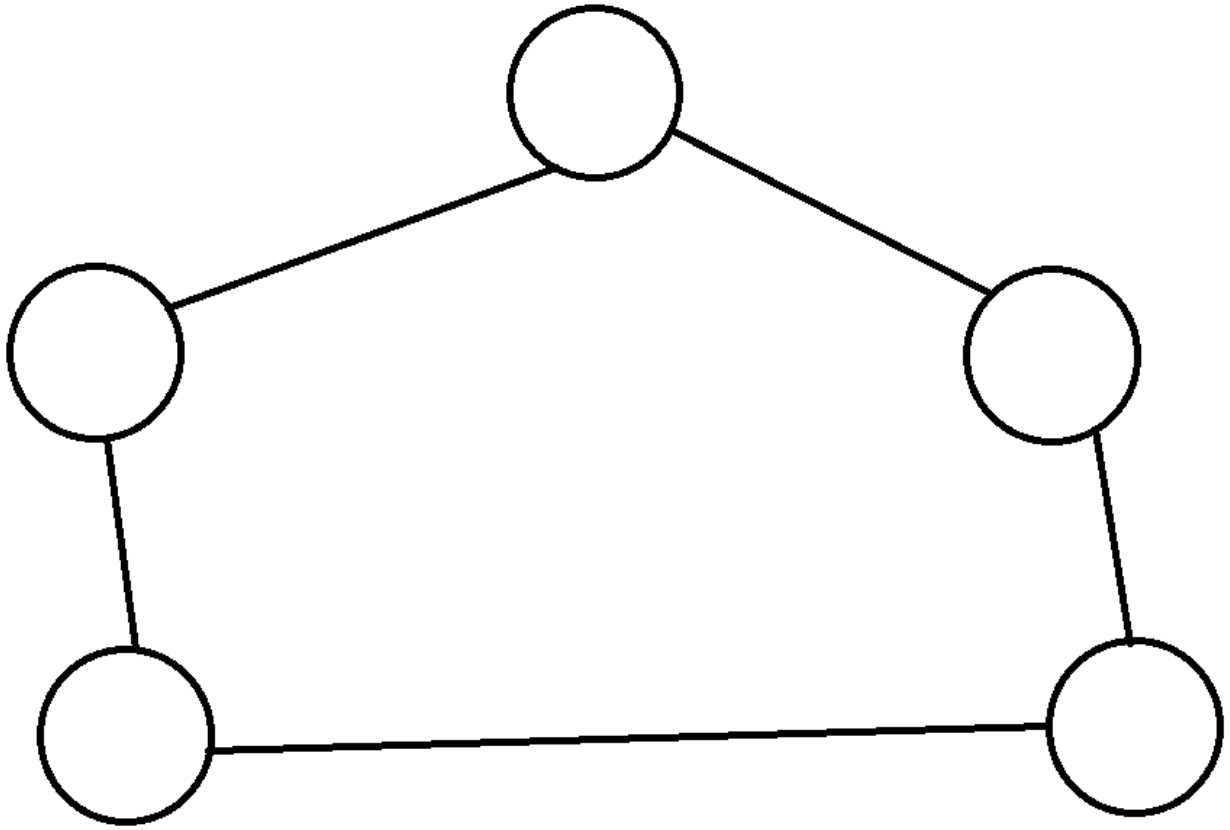

Рисунок 2.8.2 – Ring Topology

3.Зіркоподібна (Star Topology):

Опис: У цій топології всі пристрої підключені до центрального пристрою, який може бути комутатором або концентратором.

Основні властивості:

- Простота у встановленні та налаштуванні.
- Кожен пристрій має свою власну точку доступу до мережі.

- При збої центрального пристрою, тільки пристрої, підключені до нього, втрачають зв'язок, інші залишаються активними.
- Збільшення кількості пристроїв може призвести до перенавантаження центрального пристрою.[21]

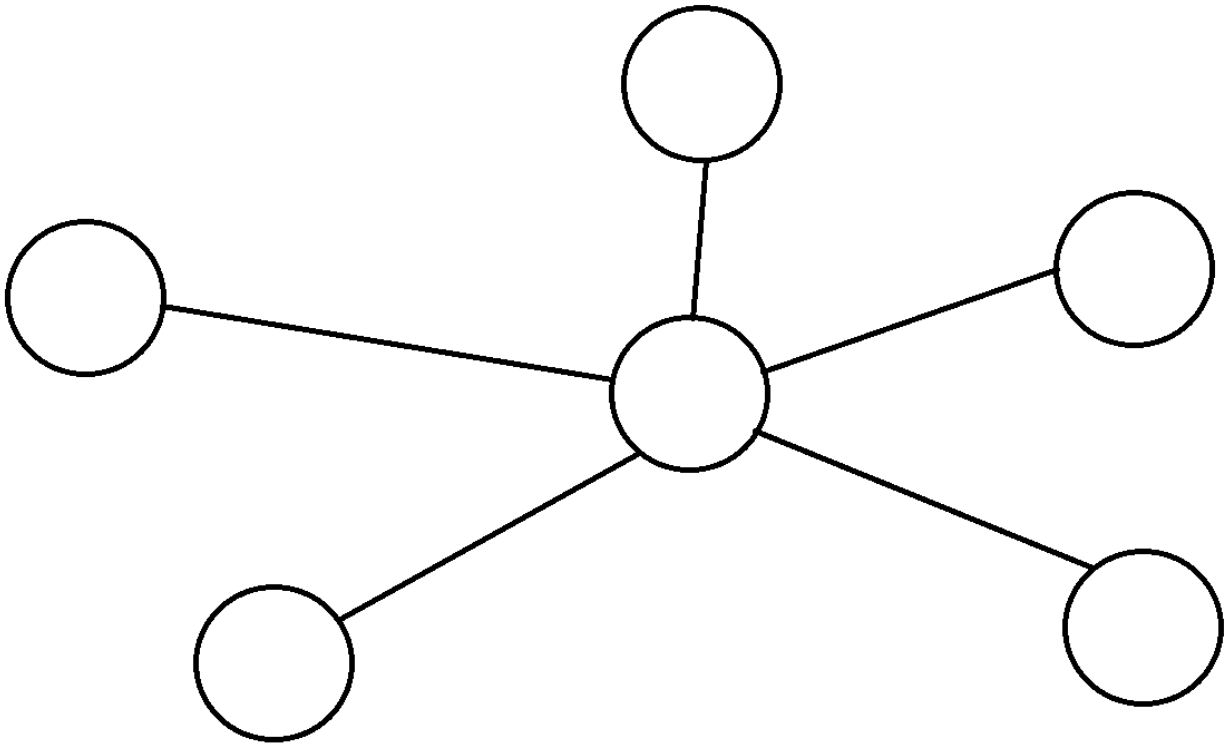

Рисунок 2.8.3 – Star Topology

4.Сітчаста (Mesh Topology):

Опис: У цій топології кожен пристрій підключений безпосередньо до кожного іншого пристрою.

Основні властивості:

- Висока надійність і доступність: якщо один шлях відомчається, дані можуть бути маршрутизовані через інший шлях.
- Висока пропускна здатність, оскільки кожен пристрій має безпосереднє підключення до кожного іншого.
- Складність у встановленні та налаштуванні.
- Велика кількість кабелів, необхідних для побудови такої мережі, що призводить до високої вартості.[22]

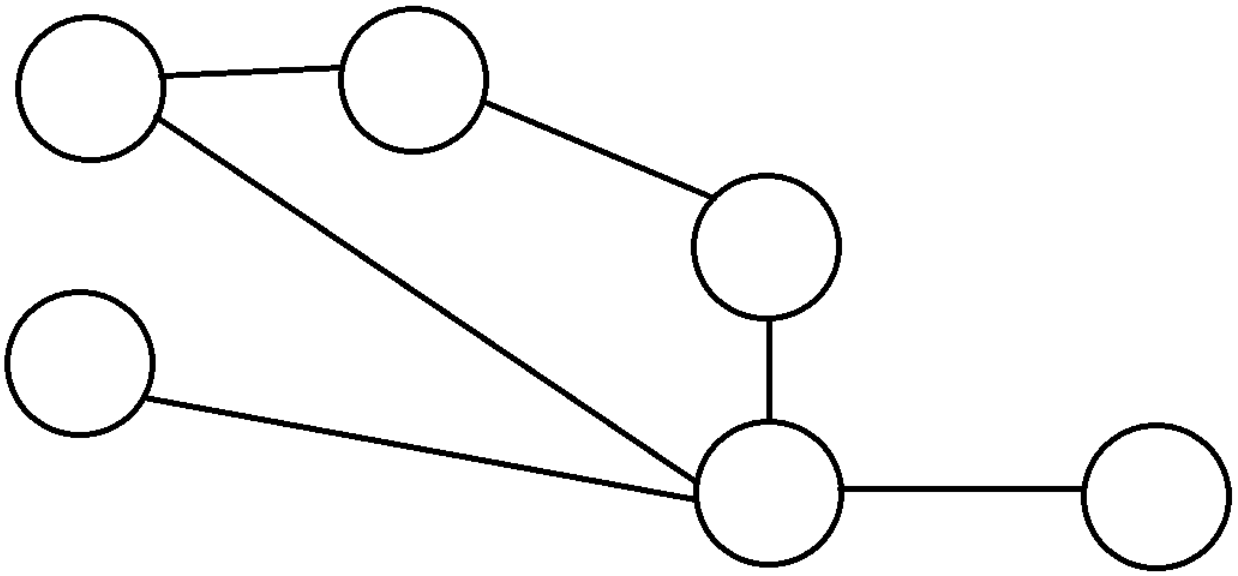

Рисунок 2.8.4 – Mesh Topology

2.9 Протоколи

Мережевий протокол — це обумовлені наперед правила передачі даних між двома пристроями. До основних параметрів, які описує протокол, відносяться:

- тип перевірки помилок, що використовується
- метод компресії (стискання) інформації (якщо такий є)
- спосіб визначення передаючим пристроєм завершення передачі

Мережеві протоколи визначають правила комунікації між двома комп'ютерами, які з'єднані в мережу.

Основні ролі протоколів:

- Адресація та маршрутизація повідомлень: Протоколи визначають способи призначення адреси кожному комп'ютеру в мережі та маршрутизації повідомлень між ними.
- Виявлення та відновлення помилок: Вони включають механізми для виявлення та виправлення помилок, які можуть виникнути під час передачі даних через мережу.
- Контроль послідовності та потоку: Протоколи регулюють порядок, в якому повідомлення відправляються та приймаються, а також контролюють потік даних між комп'ютерами.

Протокол включає специфікацію синтаксису та семантики:

- Синтаксис визначає види та формати повідомлень, які обмінюються між комп'ютерами у мережі.
- Семантика визначає дії, які виконуються кожним з комп'ютерів при виникненні конкретних подій або станів.

Прикладом може бути протокол HTTP для комунікації між веб-браузерами та серверами.

RFC (Request For Comments) - це специфікації протоколів, що використовуються в інтернет-комунікаціях.

Наприклад: RFC 821 - це зразок специфікації, що описує комунікацію між сервером і клієнтом SMTP.

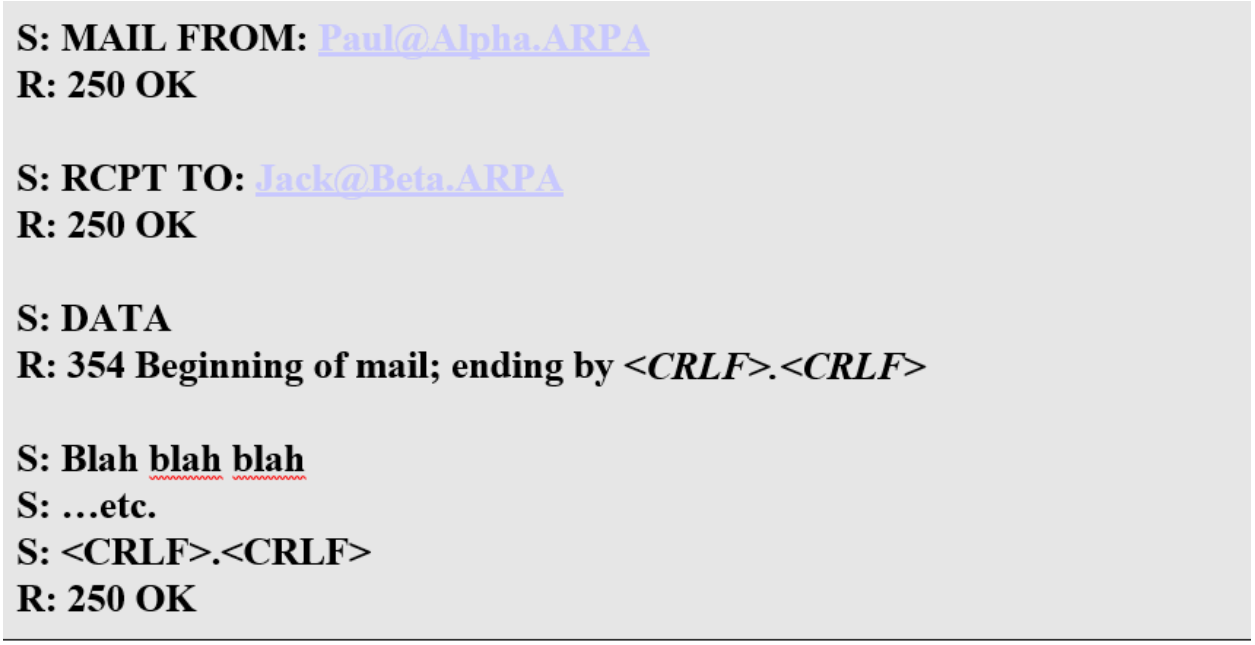

Рисунок 2.9.1 – Інтернет – комунікація

Протоколи розробляються на основі шарової архітектури, такої як модель посилання OSI.

Кожна сутність на шарі n спілкується лише з сутностями на шарі n-1.

Дані, які обмінюються, відомі як Протокольна Дана Одиниця (PDU), проходять через шари в обидва боки; кожен шар додає або видаляє свій власний заголовок і навпаки. Таким чином, PDU на шарі n може стати даними шару n-1.[13]

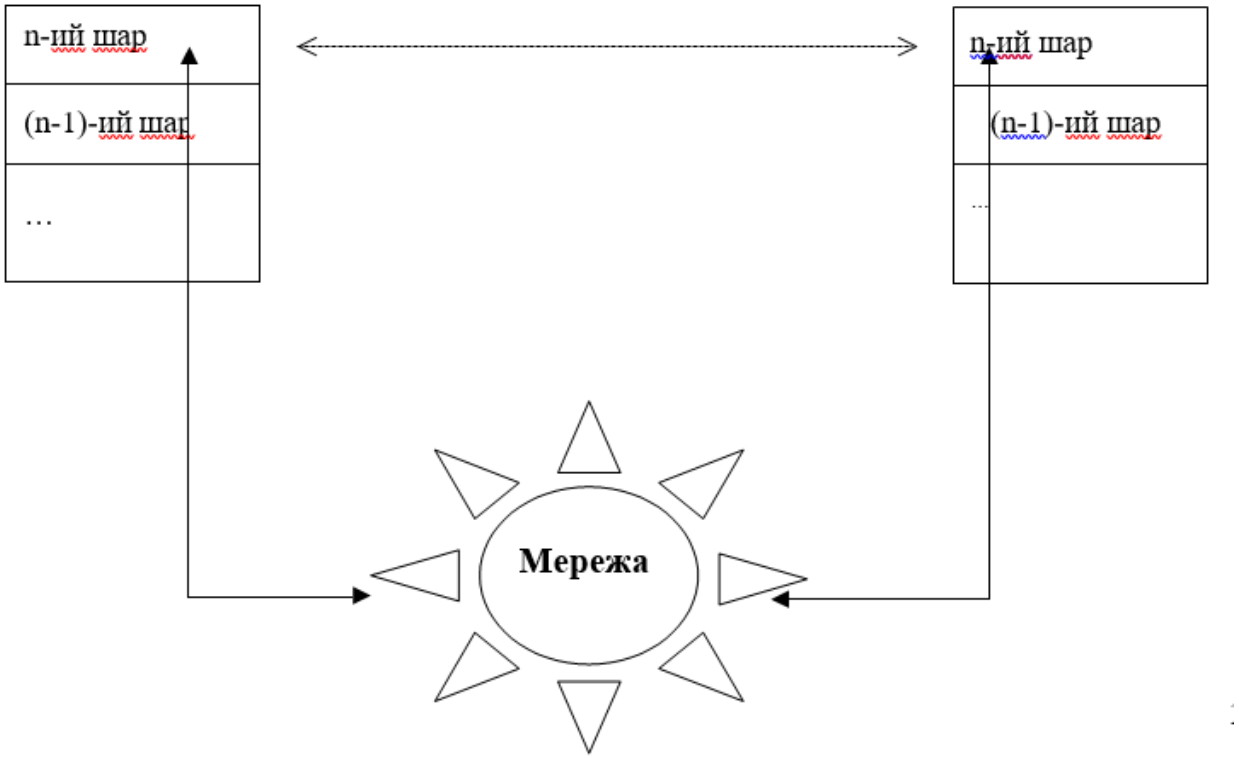

Рисунок 2.9.2 – Протоколи,на основі шарової архітектури

2.10 Шари протоколів

Модель даних OSI - це стандарт ISO для проектування та функціонування комп'ютерних мереж.

- Включає щонайменше 7 шарів, кожен з яких виконує певну роль під час взаємодії програм через мережу.
- Під час передачі кожен шар (від верхнього до нижнього) додає до оброблюваних даних певний заголовок.
- При отриманні дані обернено проходять крізь шари, заголовки видаляються, поки дані не дійдуть до приймаючої програми.

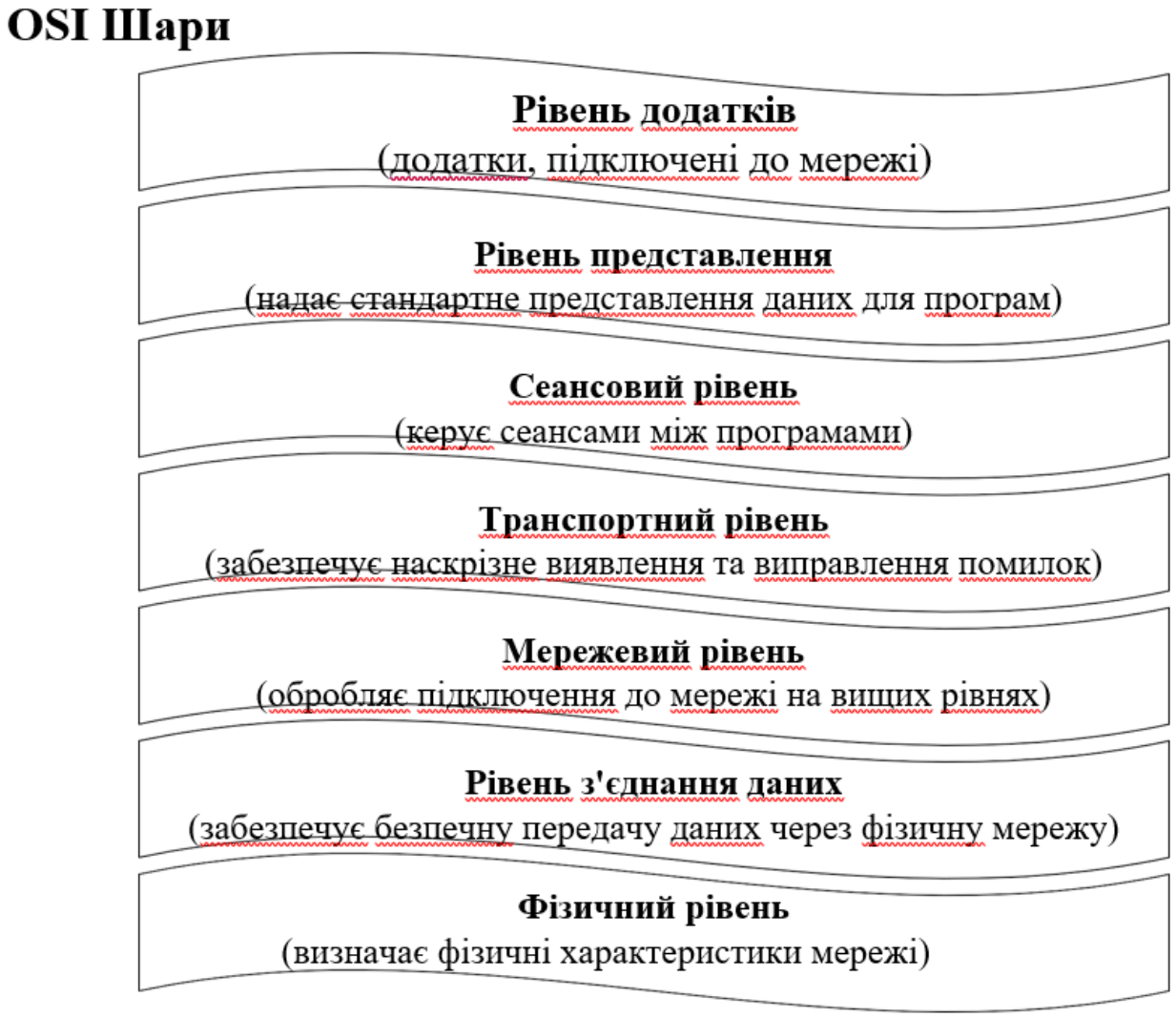

Рисунок 2.10.1 – OSI Шари

OSI Шари:

1. Фізичний рівень: забезпечує безпечну та ефективну передачу даних; складається з електронних схем для передачі даних тощо.

2. Рівень з'єднання даних: відповідає за інкапсуляцію даних у вигляді пакетів та їх інтерпретацію на фізичному рівні.

3. Мережевий рівень: відповідає за передачу пакетів від джерела A до призначення B.

4. Транспортний рівень: відповідає за доставку пакетів від джерела A до призначення B.

5. Сеансовий рівень: відповідає за управління доступом до мережі.

6. Рівень представлення: визначає формат передаваних даних для додатків, компресію/декомпресію даних, шифрування тощо.

7. Рівень додатків: містить додатки, які використовуються кінцевим користувачем, такі як Java, Word тощо.

Модель TCP/IP складається лише з 4 рівнів: додатковий, транспортний, інтернет та мережевий.

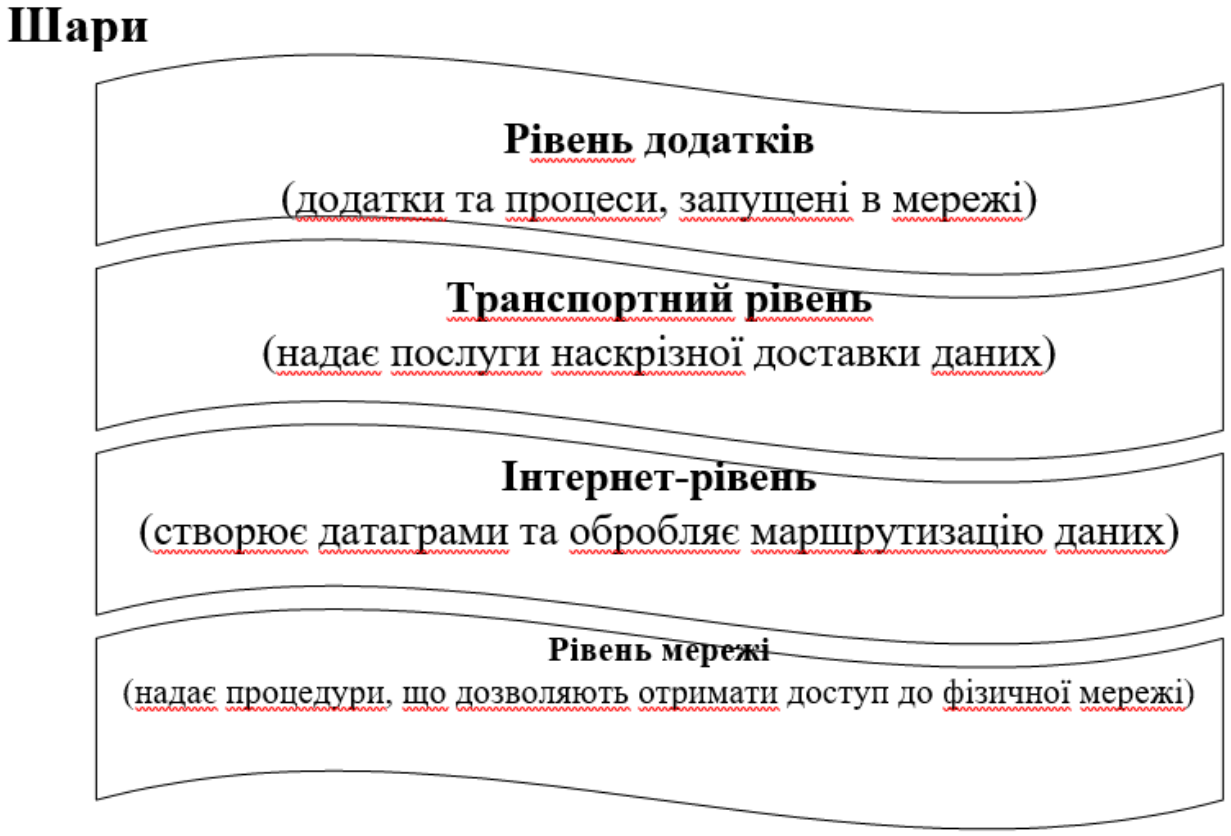

Рисунок 2.10.2 – OSI Шари,на основі TCP/IP

Рівень мережі

- Забезпечує ту ж функціональність, що й фізичний, рівень зв'язку даних та мережевий рівень в моделі OSI.
- Відповідальний за відображення між IP-адресами та мережевими фізичними адресами.
- Інкапсуляція IP-датаграм, наприклад, пакетів, у формат, зрозумілий для мережі.

Інтернет-рівень

1. Знаходиться у серці TCP/IP.

2. Оснований на Протоколі Інтернету (IP), який надає каркас для передачі даних з місця A в місце B.

Транспортний рівень

- Оснований на двох основних протоколах: TCP (Протокол керування передачею) та UDP (Протокол користувацьких датаграм).[13]

Рівень додатків

- Об'єднує функції додатку, представлення та сесійного рівня OSI.
- Протоколи, що використовуються на цьому рівні: HTTP, FTP, SMTP і т. д.

2.11 Протокол Інтернету (IP)

- Протокол IP надає дві основні функціональності:
  - Розкладання початкового потоку інформації на пакети стандартизованого розміру та їх збирання на призначенні.
  - Маршрутизація пакета через послідовні мережі, від джерела до призначення, ідентифікованого його IP-адресою.
- Передавані пакети не гарантується, що вони будуть доставлені (протокол датаграми).
- Протокол IP не запитує підключення (беззв'язковий) перед відправленням даних і не виконує жодної перевірки на помилки.

Функції:

- Розкладання початкових даних (що відправляються) на датаграми.
- Кожна датаграма буде мати заголовок, включаючи IP-адресу та номер порту призначення.
- Датаграми потім відправляються до вибраних шлюзів, наприклад, IP-маршрутизаторів, які підключені одночасно до локальної мережі та мережі постачальника IP-послуг.

● Датаграми передаються від шлюза до шлюза до того моменту, поки вони не прибудуть на кінцевий пункт призначення.

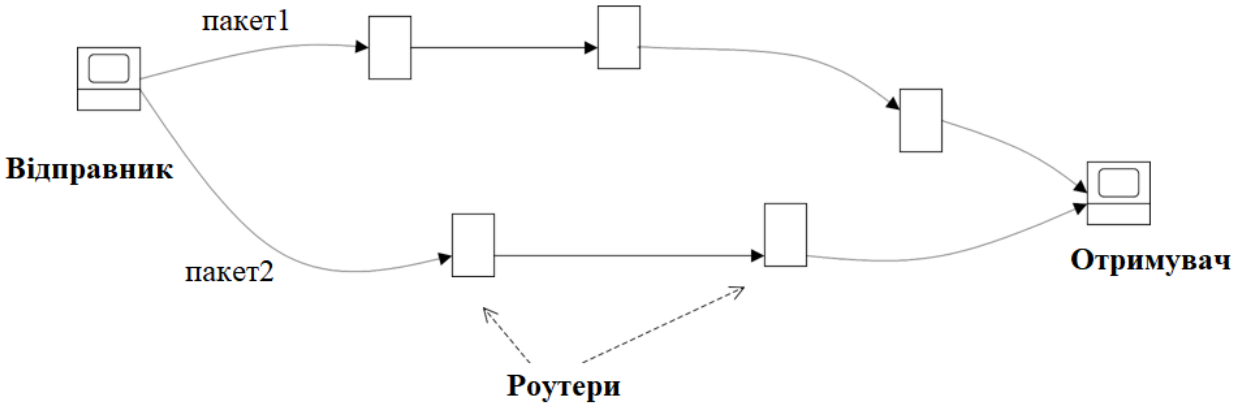

Рисунок 2.11.1 – Відправка пакетів,між відправником,та отримувачем,через роутери

Структура IP-пакету

● Поля на початку пакету, які називаються заголовком кадру, визначають функціональність та обмеження протоколу IP.

● Для кодування адреси джерела та призначення виділяється 32 біти (32 біти для кожного з цих полів адрес).

● Решта заголовка (16 біт) кодує різну інформацію, таку як загальна довжина пакету в байтах.

● Отже, IP-пакет може бути максимум довжиною 64 Кб.[13]

Структура IP-пакету

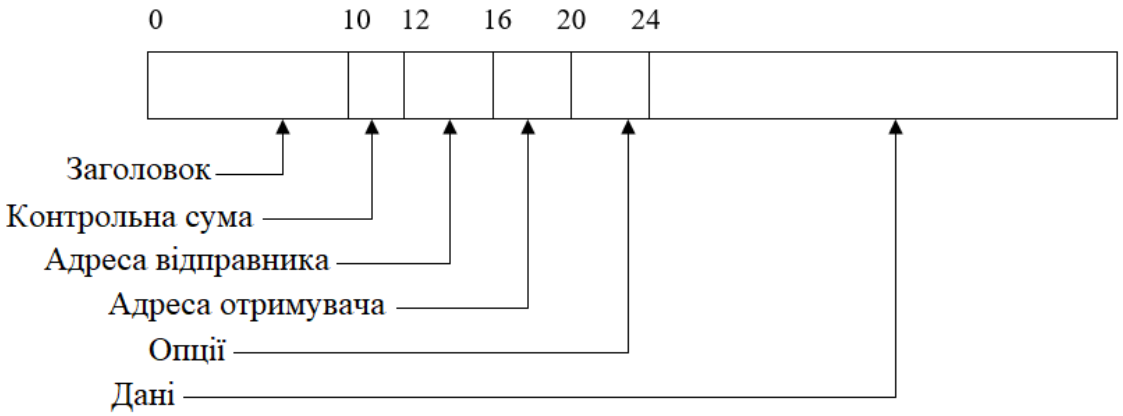

Рисунок 2.11.2 – Структура IP - пакету

2.12 Протокол управління передачею (TCP)

- TCP забезпечує базовий сервіс за допомогою пакетів IP, який гарантує безпечну доставку:
  - виявлення помилок
  - безпечна передача даних
  - гарантія того, що дані отримані в правильному порядку
- Перед відправленням даних TCP вимагає, щоб комп'ютери, що спілкуються, встановили з'єднання (протокол орієнтований на з'єднання).

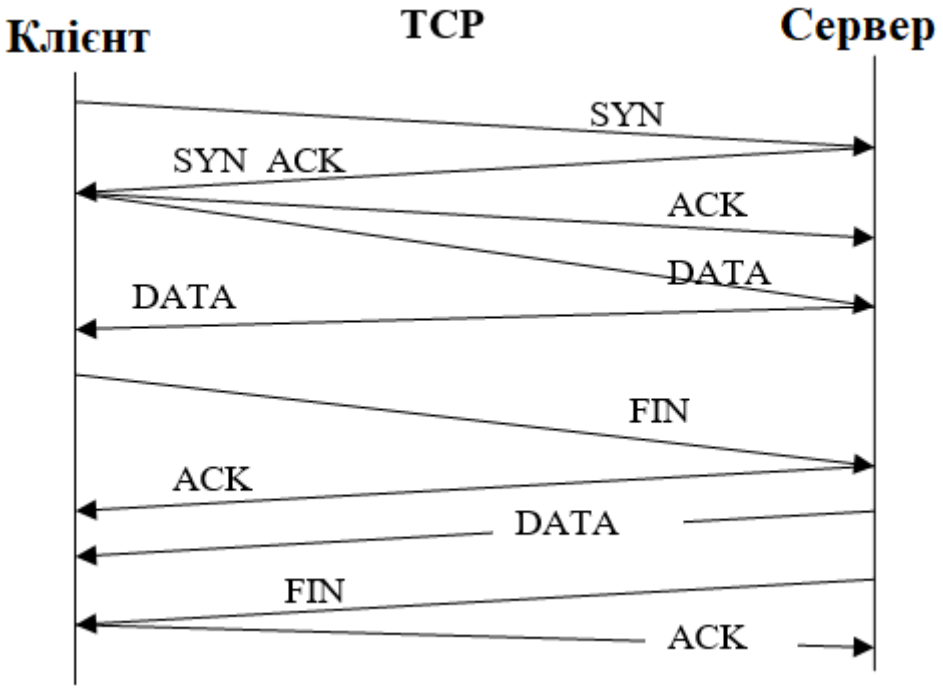

Рисунок 2.12.1 – Взаємодія клієнта,та сервера,через TCP

- TCP забезпечує підтримку для надсилання та отримання довільних обсягів даних як один великий потік байтів (IP обмежений до 64 Кб).
- TCP робить це, розбиваючи потік даних на окремі пакети IP.
- Пакети нумеруються, а потім збираються при отриманні, використовуючи номери послідовності та номери підтвердження послідовності.
- TCP також покращує можливості IP, вказуючи порти.
  - Існує 65 536 різних TCP-портів (роз'ємів), через які кожна машина TCP/IP може спілкуватися.[13]

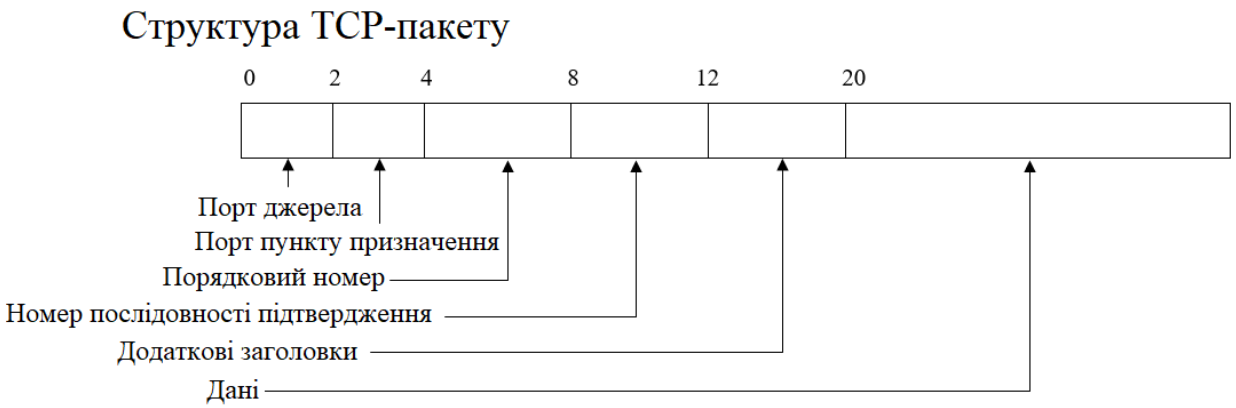

Рисунок 2.12.2 – Структура TCP - пакету

2.13 Протокол користувацьких датаграм (UDP)

- Датаграмний протокол також побудований на основі IP.
- Має те саме обмеження розміру пакета (64 Кб), що й IP, але дозволяє вказати номер порту.
- Надає також 65 536 різних портів.
- Отже, кожна машина має два набори з 65 536 портів: один для TCP, а інший для UDP.
- Безз'єднаність протоколу, без будь-яких можливостей виявлення помилок.
- Надає лише підтримку передачі даних з одного кінця до іншого без будь-яких подальших перевірок.
- Основний інтерес UDP полягає в тому, що, оскільки він не проводить подальшу перевірку, він дуже швидкий.
- Корисний для надсилання даних малого розміру у повторному режимі, наприклад, інформації про час.[13]

2.14 Протоколи інтернет-додатків

Над TCP/IP розроблено кілька служб для однорідності застосунків однакового характеру:

- FTP (Протокол передачі файлів) дозволяє передавати колекції файлів між двома машинами, підключеними до Інтернету.
- Telnet (Термінальний протокол) дозволяє користувачеві підключатися до віддаленого хоста в режимі терміналу.
- NNTP (Протокол передачі мережевих новин) дозволяє створювати групи спілкування (новинні групи), організовані за певними темами.
- SMTP (Простий протокол передачі пошти) визначає основну службу для електронної пошти.
- SNMP (Простий протокол управління мережею) дозволяє керувати мережею.[13]

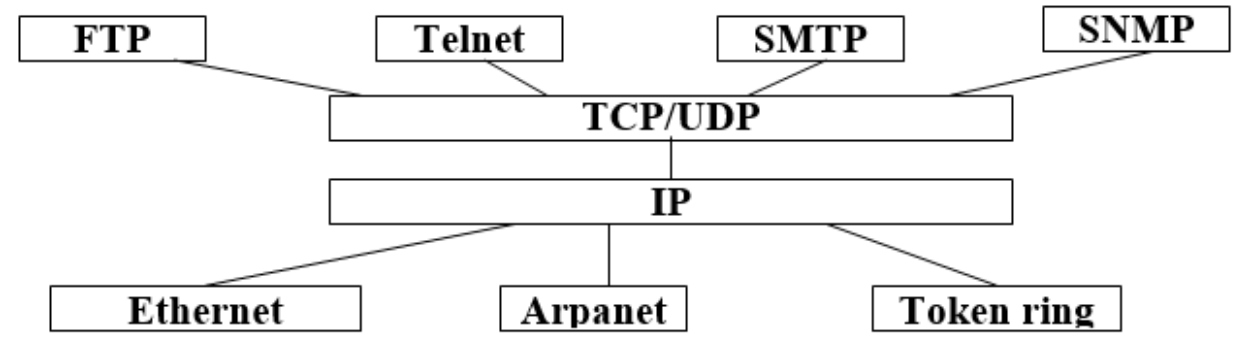

Рисунок 2.14.1 – Протоколи інтернет - додатків

2.15 VPN

VPN - це віртуальна приватна мережа (скорочення від virtual private network). Це зашифрований тунель між двома пристроями, що дозволяє отримати доступ до будь-якого веб-сайту та онлайн-сервісу конфіденційно та безпечно.[23]

VPN-тунель - це пряме з'єднання, встановлене між двома пристроями - зазвичай, між VPN-сервером і вашим пристроєм. Тунелювання дозволяє вкласти трафік у стандартні пакети TCP/IP і безпечно передавати його через Інтернет. Оскільки ваші дані зашифровані, то ні хакери, ні уряд, ні навіть

інтернет-провайдери не матимуть доступу до них, поки ви підключені до VPN-сервера.

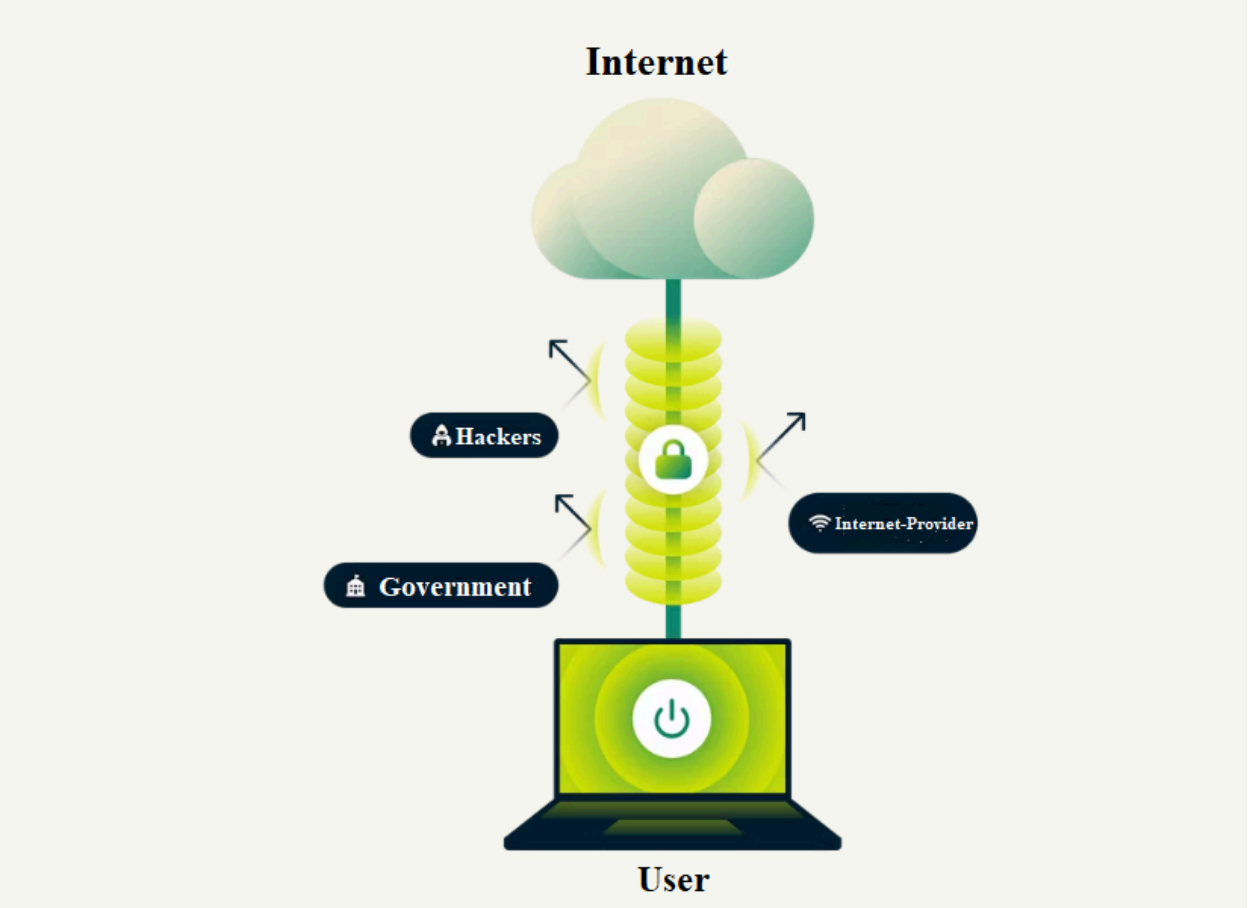

Рисунок 2.15.1 – Демонстрація роботи VPN

Коли ви підключаєтесь до проксі-сервера, він стає свого роду посередником між вашим пристроєм і Інтернетом. Увесь ваш інтернет-трафік проходить через проксі-сервер і, відповідно, отримує його IP-адресу.

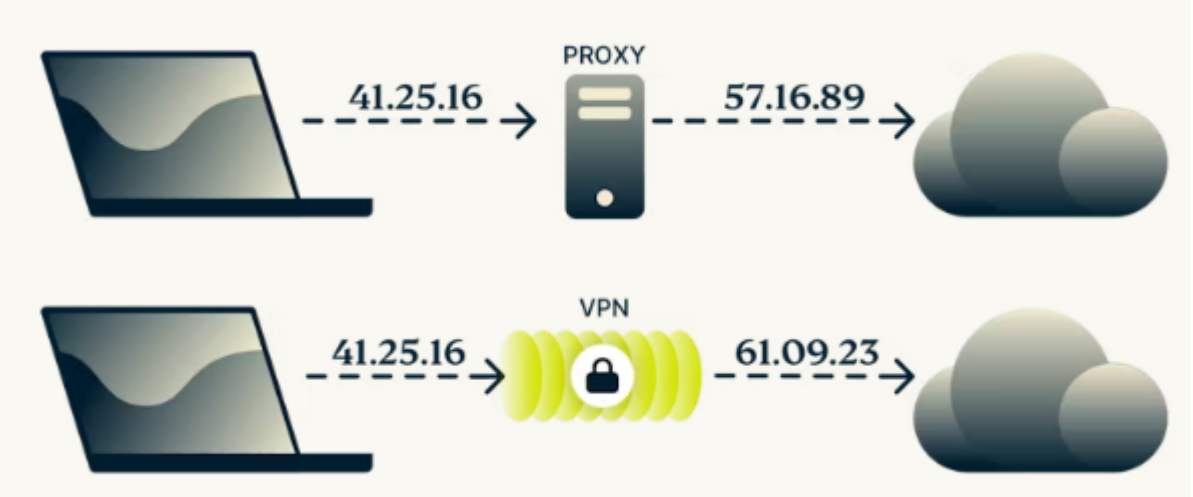

Рисунок 2.15.2 – Відмінності,між Proxy – сервером,та VPN

Підключення до проксі-сервера приховує ваш вихідний IP-адрес і дозволяє отримати доступ до заблокованого контенту. Однак проксі-сервери не шифрують трафік, тому всі дані, якими ви обмінюєтеся через встановлене з'єднання, можуть бути перехоплені сторонніми особами, підключеними до того ж серверу - наприклад, хакерами.

VPN-сервіс забезпечить вам всі переваги проксі-сервера, а також захистить і зашифрує трафік, яким ваш пристрій обмінюється з інтернет-серверами. Це дозволить вам працювати в Інтернеті, не боячись крадіжки або перехоплення ваших даних.

Файрвол - це спеціальний бар'єр, який аналізує пакети даних від інтернет-серверів, що намагаються отримати доступ до вашого комп'ютера. Файрвол пропускає лише ті пакети, які відповідають певним правилам.

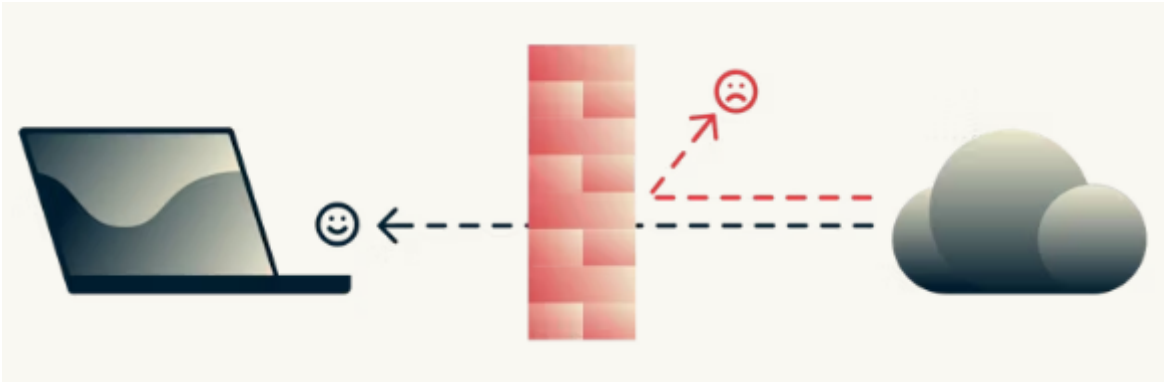

Рисунок 2.15.3 – Демонстрація,роботи файрвола

Використання файрволу буде відмінним способом захисту вашого пристрою від вірусних атак і комп'ютерних червей. Проте файрвол може захистити вас лише від небезпечного вхідного трафіку. Щоб захистити ваш вихідний мережевий трафік, скористайтеся VPN-сервісом. З свого боку, файрвол стане відмінним доповненням до VPN: використовуючи обидва ці рішення, ви отримаєте максимум захисту.[24]

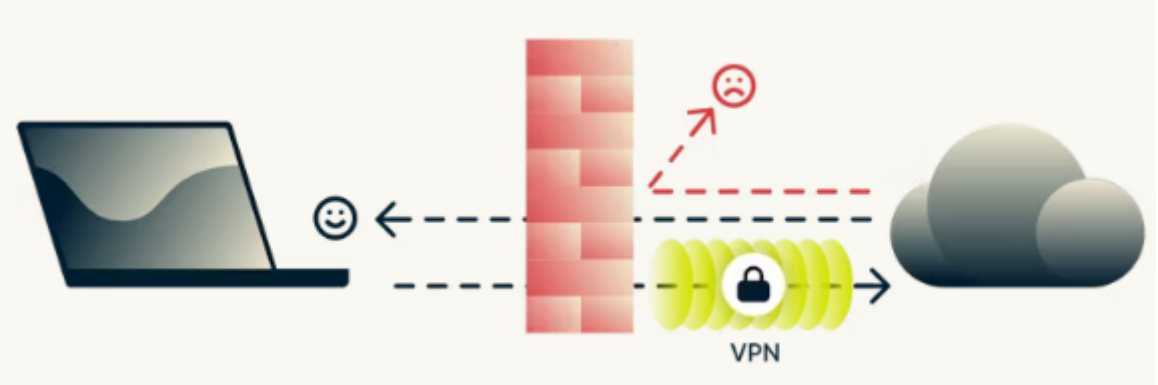

Рисунок 2.15.4 – Обхід файрвола,за допомогою VPN

3.ПРОЕКТНІ РІШЕННЯ

3.1 Використані програми

У тестовій реалізації будуть використані,такі програми:

1. Visual Studio Code — текстовий редактор, розроблений Microsoft для Windows, Linux та macOS. Позиціонується як "легкий" редактор коду для кросплатформенної розробки веб- та хмарних додатків. Включає в себе відлагоджувач, інструменти для роботи з Git, підсвічування синтаксису, IntelliSense та засоби для рефакторингу. Має широкі можливості для кастомізації: користувацькі теми, комбінації клавіш та файли конфігурації. Розповсюджується безкоштовно, розробляється як програмне забезпечення з відкритим вихідним кодом, але готові збірки розповсюджуються під пропрієтарною ліцензією.[25]

2. Node або Node.js — програмна платформа, заснована на двигуні V8 (що компілює JavaScript у машинний код), яка перетворює JavaScript з вузькоспеціалізованої мови на мову загального призначення. Node.js додає можливість JavaScript взаємодіяти з пристроями вводу-виводу через свій API, написаний на C++, підключати інші зовнішні бібліотеки, написані на різних мовах, забезпечуючи виклики до них з JavaScript-коду. Node.js використовується переважно на сервері, виконуючи роль веб-сервера, але є можливість розробляти на Node.js і десктопні віконні додатки (за допомогою NW.js, AppJS або Electron для Linux, Windows та macOS) і навіть програмувати мікроконтролери (наприклад, tessel, low.js і Espruino). В основі Node.js лежить подійно-орієнтоване та асинхронне (або реактивне) програмування з неблокуючим введенням/виведенням.[26]

3. LogMeIn Hamachi - це програма VPN, розроблена та випущена у 2004 році Алексом Панкратовим. Вона здатна встановлювати прямі зв'язки між комп'ютерами, які знаходяться за брандмауерами мережі з трансляцією адрес (NAT), без необхідності переконфігурування (коли комп'ютер користувача може бути доступний безпосередньо без ретрансляції з Інтернету/з боку WAN). Подібно до інших VPN, вона встановлює з'єднання

через Інтернет, що емулює з'єднання, яке існувало б, якщо комп'ютери були б підключені через локальну мережу (LAN).

LogMeIn Hamachi,може вважатися міжпрограмним забезпеченням, оскільки вона служить посередником для забезпечення безпечного з'єднання між різними комп'ютерами через мережу.

Hamachi дозволяє створювати локальну мережу поверх Інтернету. Найчастіше мережі Hamachi використовуються для з'єднання серверів з сірим IP і клієнтських комп'ютерів. Саме такий метод значно ускладнює дешифрування клієнтського трафіку.

Будь-які програми, які працюють через локальну мережу, можуть працювати через мережі Hamachi, при цьому передані дані будуть захищені, і обмін між ними здійснюється в стилі peer-to-peer.

Hamachi — система організації віртуальних захищених мереж на основі протоколу UDP. У такій мережі вузли для встановлення з'єднання між собою використовують третій вузол, який допомагає їм лише виявити один одного, а передача інформації здійснюється безпосередньо між вузлами. При цьому взаємодіючі вузли можуть знаходитися за NAT або файрволом.[27]

3.2 Застосування програм

Перш,створюємо спільну папку,для комп’ютерів,в якому створимо спільний сервер,та файли,з паралельними обчисленнями.

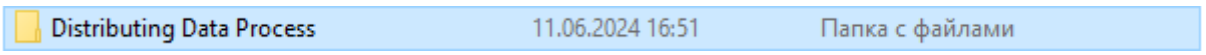

Рисунок 3.2.1 – Створення спільної папки

Далі,натискаємо правою кнопкою миші,на папку,та вибираємо “Властивості”,потім “Доступ”,а потім “Спільний доступ…”.

Далі,надаємо усім користувачам,доступ,а саме “Читання та запис”,після чого натискаємо “Поділитися”.

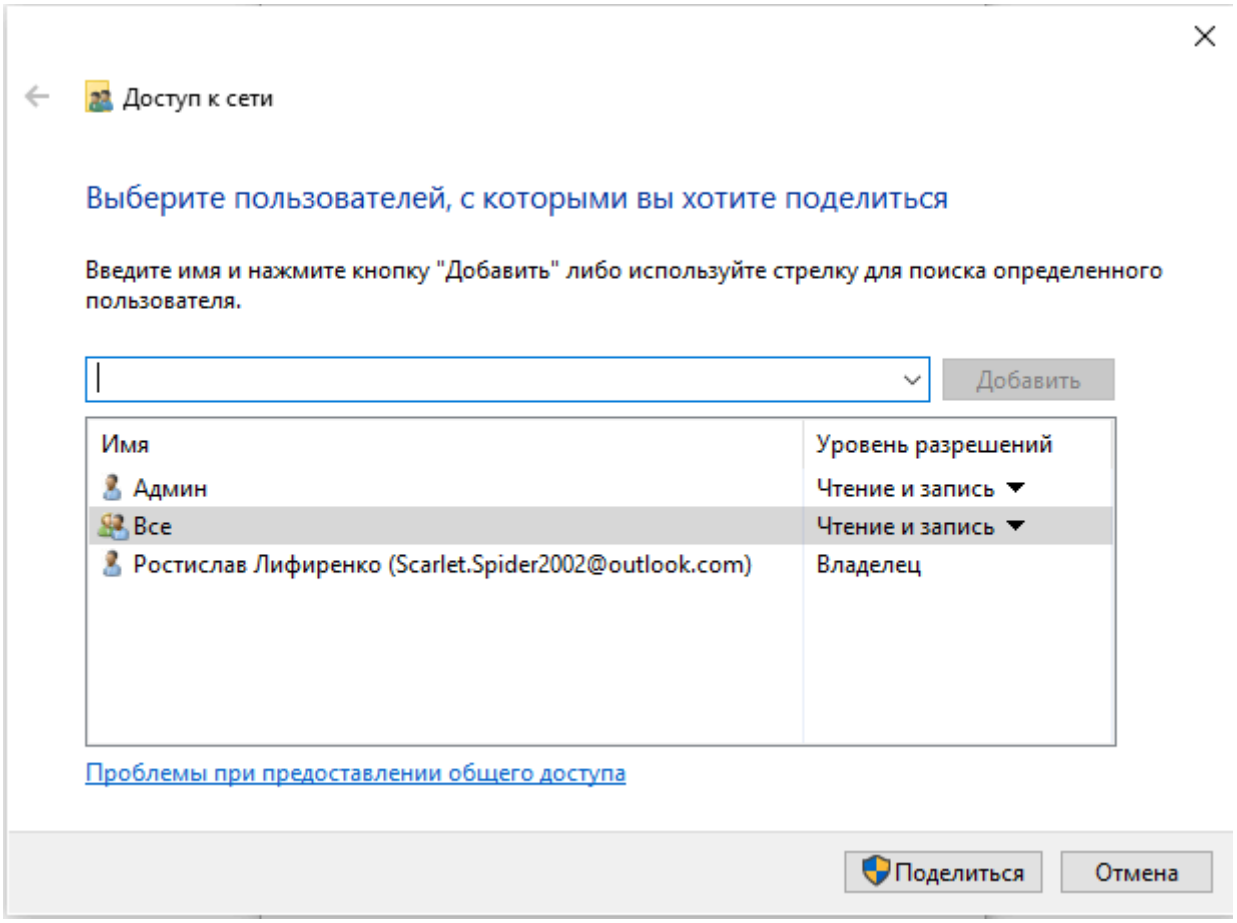

Рисунок 3.2.2 – Дозвіл,усім користувачам

У,доступі,треба вибрати “Розширене налаштування”,потім обрати “Дозволи”.

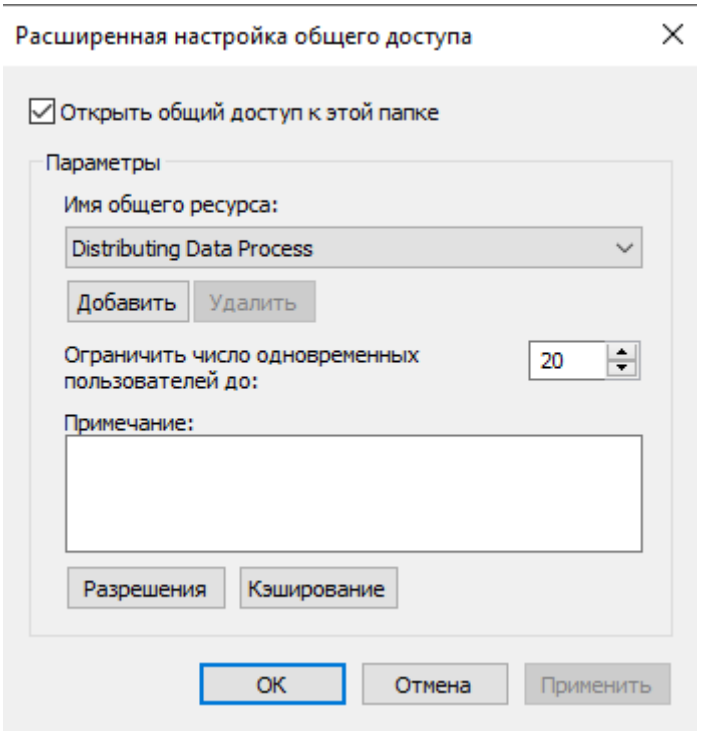

Рисунок 3.2.3 – Розширені налаштування спільного доступу

Тепер,надаємо повноцінний доступ,з редагуваннями,та додаванням нових файлів,та натискаємо “OK”.

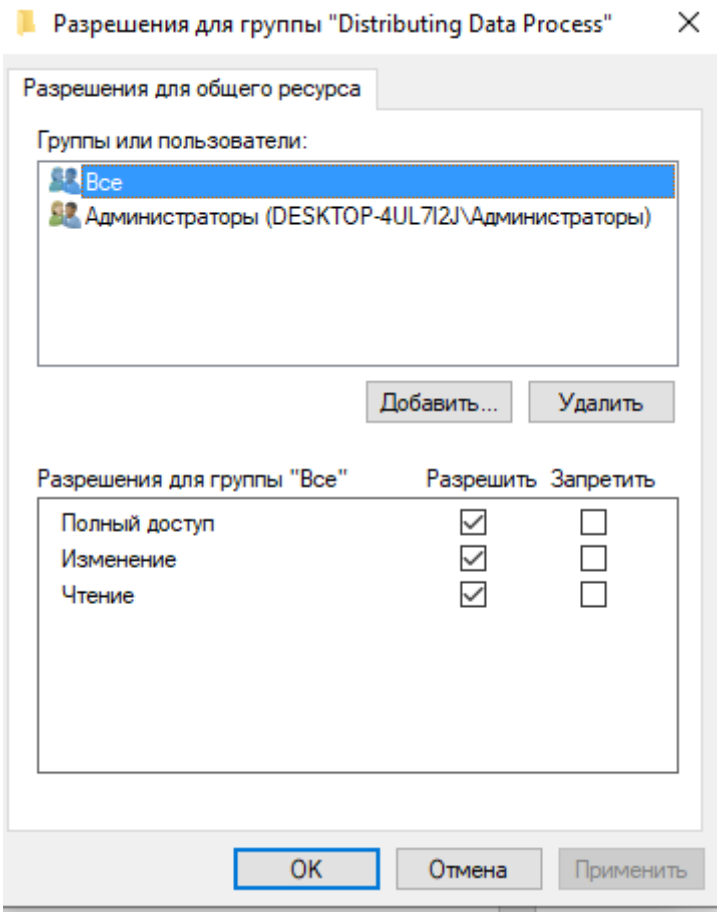

Рисунок 3.2.4 – Дозвіл,усім користувачам,на редагування файлів

Спільна папка,для комп’ютерів,була створена,тепер підключені комп’ютері,теж можуть приймати участь,у розподіленій обробці даних.

Тепер,створимо:шість HTML-файлів,які будуть,у розподіленій обробці,реалізовувати паралельні обчислення(у,якості прикладу);один HTML-файл,буде надавати інформацію про клієнтів;один JS-файл,який буде реалізовувати клієнт-серверний веб-зв’язок,у якому,буде демонструватись робота клієнтів/комп’ютерів.

Примітка:Створення,кодів програм,буде здійснено Visual Studio Code.

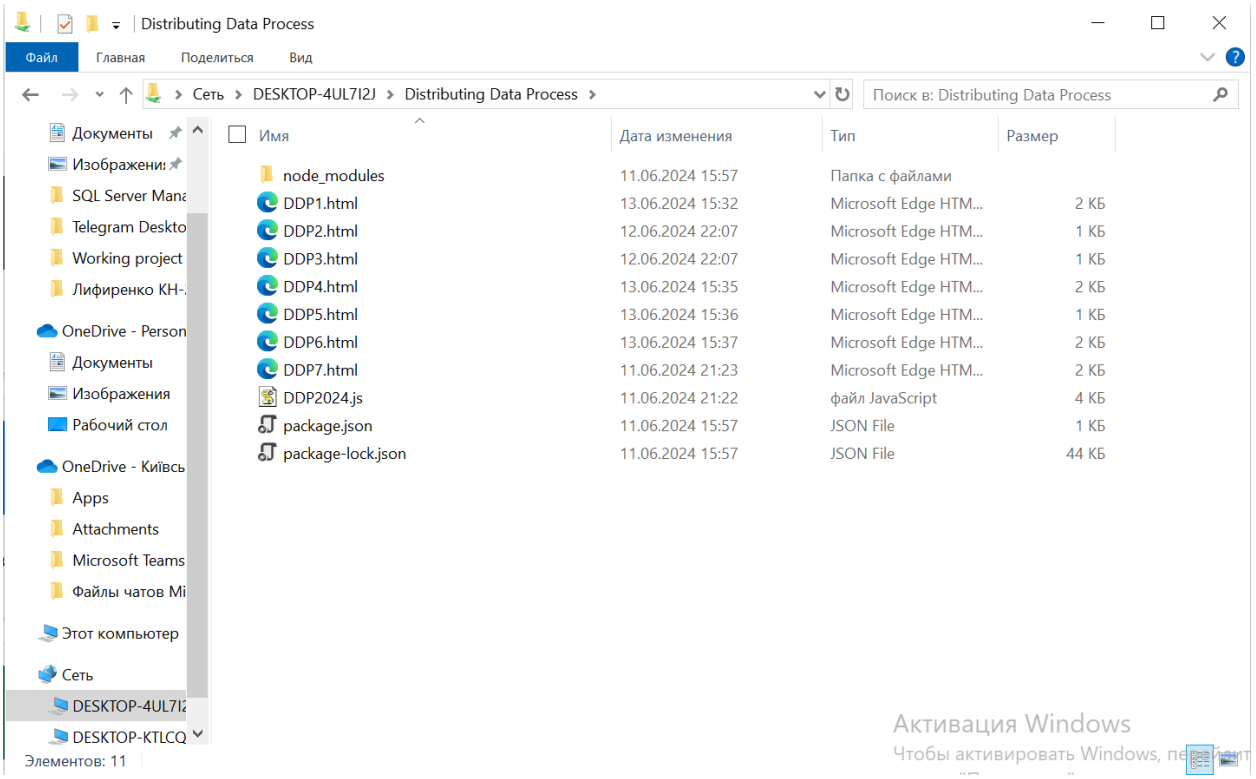

Рисунок 3.2.5 – Спільна папка,для клієнтів

3.2.1 Поділ задачі на підзадачі

Опис:

Поділ задачі на підзадачі передбачає розділення основного завдання на менші, більш керовані частини, які можна виконувати незалежно або з мінімальною взаємодією між ними. Спочатку потрібно чітко визначити та описати кожну підзадачу, враховуючи логічну послідовність їх виконання, щоб забезпечити правильний кінцевий результат. Далі, на основі доступних ресурсів (процесорів або ядер), кожну підзадачу призначають конкретним виконавчим одиницям, оптимізуючи завантаження системи. Важливо також передбачити механізми синхронізації та обміну даними між підзадачами, якщо це необхідно для коректного функціонування всієї задачі.[29]

Програмна реалізація:

Програма,буде виконувати дві складні обчислювальні задачі паралельно: task1 обчислює суму Math.sin(i) * Math.cos(i) для і від 0 до 9999999, а task2 обчислює суму Math.sqrt(i) * Math.log(i) для і від 1 до 9999999. Вони

використовують setTimeout(..., 0), щоб не блокувати основний потік і виводять результати відповідно до result1 і result2.

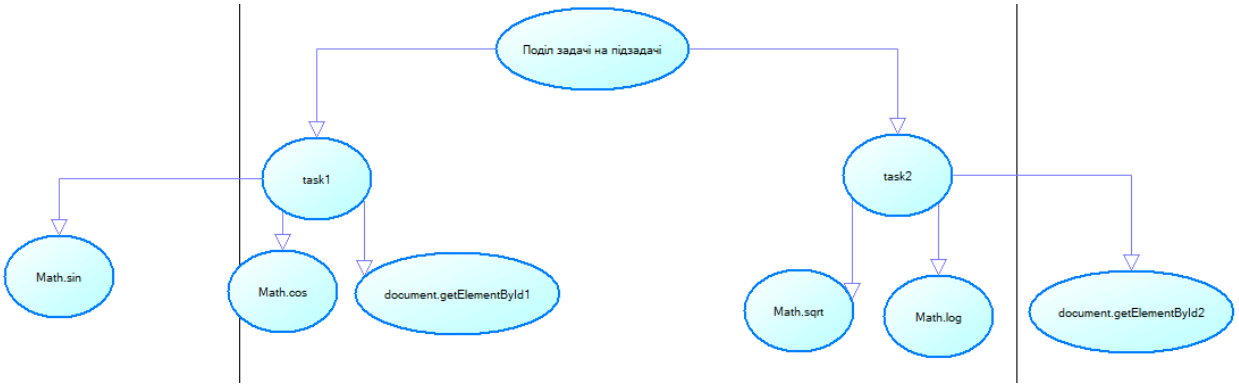

Рисунок 3.2.1.1 - Дерево функцій поділу задачі на підзадачі

3.2.2 Поділ даних

Опис:

Поділ даних (Data Parallelism) полягає у розбитті вхідних даних на окремі частини для паралельної обробки. Спочатку, дані розділяються на блоки, які потім розподіляються між доступними процесорами або ядрами. Кожен процесор або ядро виконує однакові операції над своєю частиною даних одночасно з іншими. Це дозволяє значно пришвидшити обробку великих обсягів даних. Після завершення паралельної обробки, результати з окремих частин об'єднуються для отримання кінцевого результату. Такий підхід особливо ефективний при роботі з великими масивами або матрицями, де однакові операції можуть бути застосовані до різних частин даних одночасно.[30]

Програмна реалізація:

Програма масиву data,буде виконувати складну операцію — піднесення кожного елементу до куба за допомогою вбудованої функції Math.pow(). Ця операція відбувається у циклі for, який ітерується по всім елементам масиву. Після обробки дані змінюються у самому масиві data, який потім повертається як результат функції. Використання join(', ') при виведенні результату дозволяє отримати рядок, де кожен оброблений елемент розділений комою та пробілом. Програма взагалі є простою в обробці, але ефективною, заснованою на зміні

масиву у місці, що може бути важливо для оптимізації пам'яті та швидкодії, особливо при великих обсягах даних.

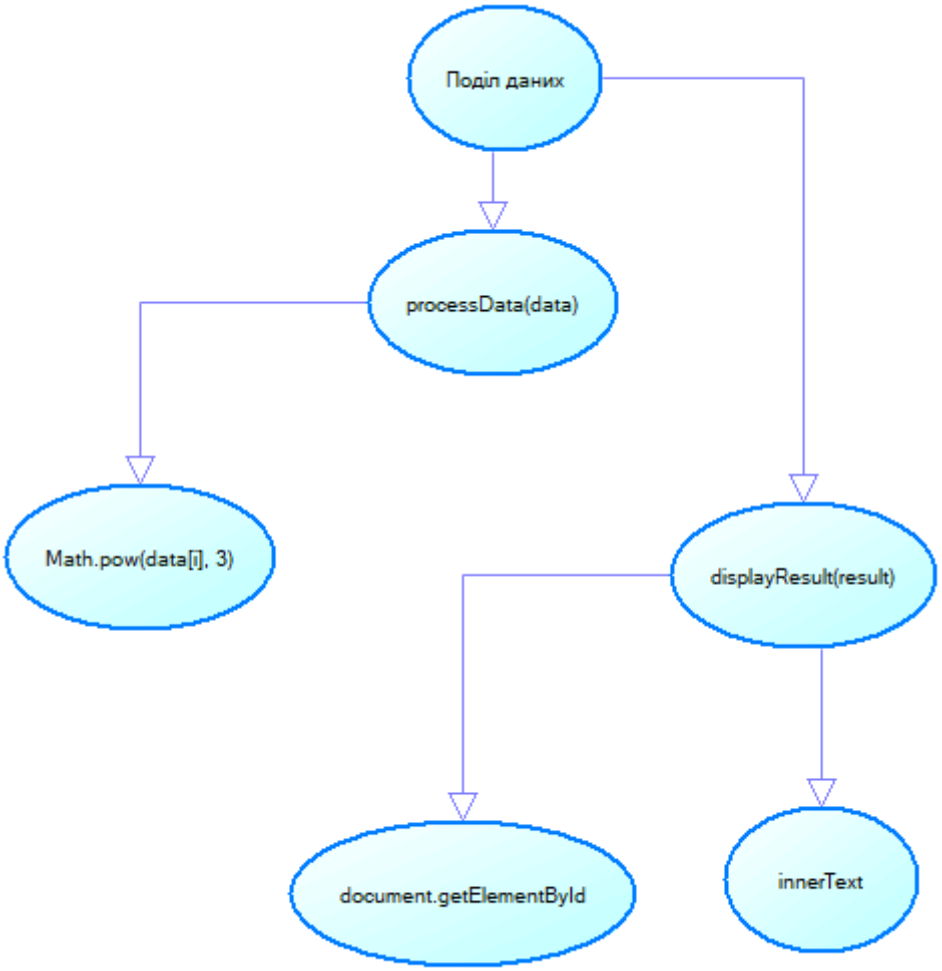

Рисунок 3.2.2.1 - Дерево функцій поділу даних

3.2.3 Паралельні алгоритми

Опис:

Паралельні алгоритми — це методи обчислень, які дозволяють одночасно виконувати декілька операцій за допомогою розподілу задач між кількома процесорами або ядрами. У контексті паралельного сортування, наприклад, елементи, що підлягають сортуванню, розподіляються між доступними процесорами. Кожен процесор використовує паралельні алгоритми сортування, такі як паралельний quicksort або паралельний merge sort, для обробки своєї частини даних. Потім відсортовані частини об'єднуються для отримання

кінцевого відсортованого масиву. У випадку паралельного пошуку і сортування графів, граф розбивається на частини, які розподіляються між процесорами. Кожен процесор паралельно виконує пошук або сортування для своєї частини графу, а результати потім об'єднуються для отримання повної картини. Такий підхід дозволяє значно прискорити обчислення, особливо при роботі з великими обсягами даних або складними структурами, такими як графи.[31]

Програмна реалізація:

Програма,буде виконувати сортування масиву чисел за спаданням, використовуючи вбудовану функцію JavaScript .sort(). У функції parallelSort, яка визивається з масивом data як параметром, застосовується порівняння b - a, що призводить до сортування елементів у порядку зменшення значень. Особливість програми полягає в тому, що сортування відбувається у вбудованій методі масивів JavaScript, що забезпечує високу ефективність і простоту реалізації. Після виконання сортування результат виводиться на сторінку у вказаному DOM-елементі з id result4 у вигляді розділених комами значень відсортованого масиву.

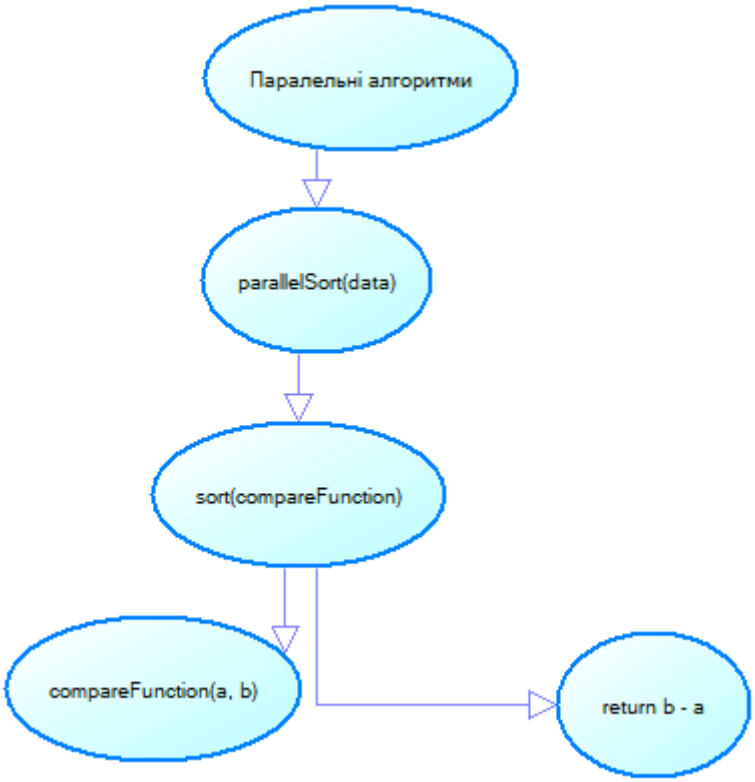

Рисунок 3.2.3.1 - Дерево функцій паралельних алгоритмів

3.2.4 Системи паралельних черг

Опис:

Системи паралельних черг (Pipeline Parallelism) організовують обробку задач шляхом їх розділення на послідовні етапи, де кожен етап виконується окремим процесором або потоком паралельно. Завдання розбивається на кілька кроків (етапів), і кожен етап працює над окремою частиною даних, передаючи результати на наступний етап. Це створює конвеєр обробки, де одночасно обробляються різні частини задачі на різних етапах. Така структура дозволяє ефективно використовувати ресурси та підвищити продуктивність системи, забезпечуючи при цьому правильний потік даних між етапами для узгодженості результатів.[32]

Програмна реалізація:

Програма,буде виконувати послідовну обробку даних через три функції: stage1, stage2 і stage3, які множать вхідне значення на 2, додають 5 і ділять на 3 відповідно. Початкове значення 10 піддається цим етапам обробки, результат виводиться на екран.

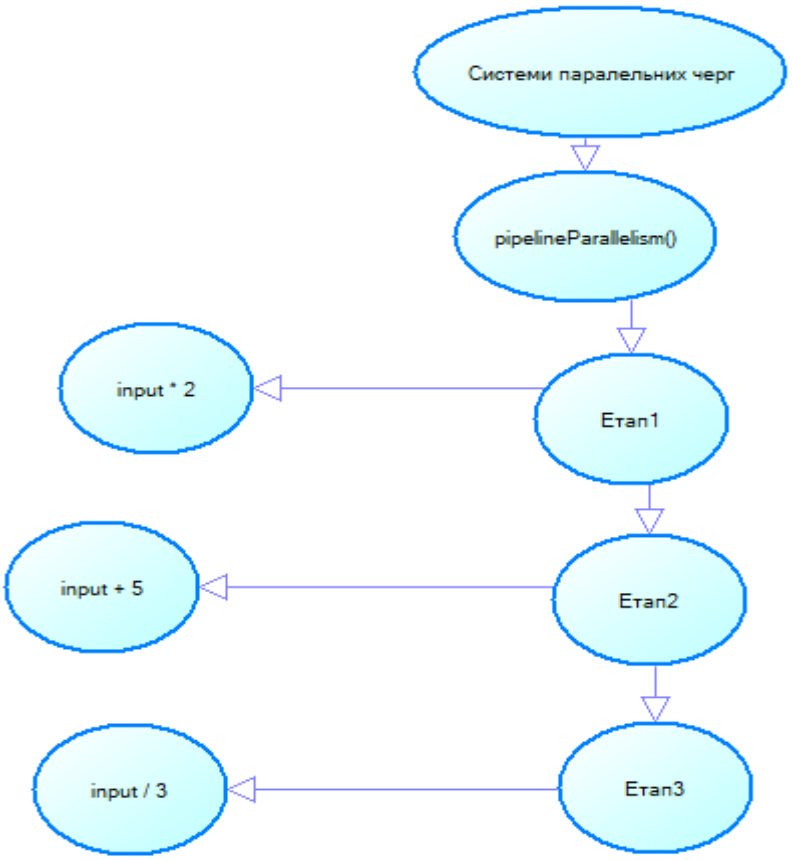

Рисунок 3.2.4.1 - Дерево функцій системи паралельних черг

3.2.5 Векторні обчислення

Опис:

Векторні обчислення (Vector Processing) використовують підхід SIMD при якому одна інструкція застосовується до багатьох елементів даних одночасно. Це дозволяє виконувати операції паралельно над векторами даних, що суттєво прискорює обчислення порівняно з традиційним покроковим підходом. Це включає завантаження даних у векторні регістри, використання відповідних векторних інструкцій для обробки даних та оптимізацію коду для зменшення затримок доступу до пам'яті. Такий підхід широко використовується в обробці зображень, наукових обчисленнях, машинному навчанні та інших областях, де потрібна висока продуктивність обробки великих обсягів даних.[33]

Програмна реалізація:

Програма,буде виконувати векторну операцію множення двох векторів vectorA і vectorB. Вектори задані як масиви чисел [1, 2, 3, 4, 5] і [6, 7, 8, 9, 10].

Під час виконання програми створюється новий масив result, куди зберігається результат елементних множень відповідних елементів з vectorA і vectorB. Наприклад, перший елемент result буде дорівнювати 1 * 6 = 6, другий 2 * 7 = 14, і так далі. Після цього результат виводиться на веб-сторінку в елемент з id result6 у вигляді рядка, де кожен елемент масиву result перетворюється в рядок. Програма використовує простий цикл for для ітерації по елементах масивів і метод push() для додавання нових значень до масиву result.

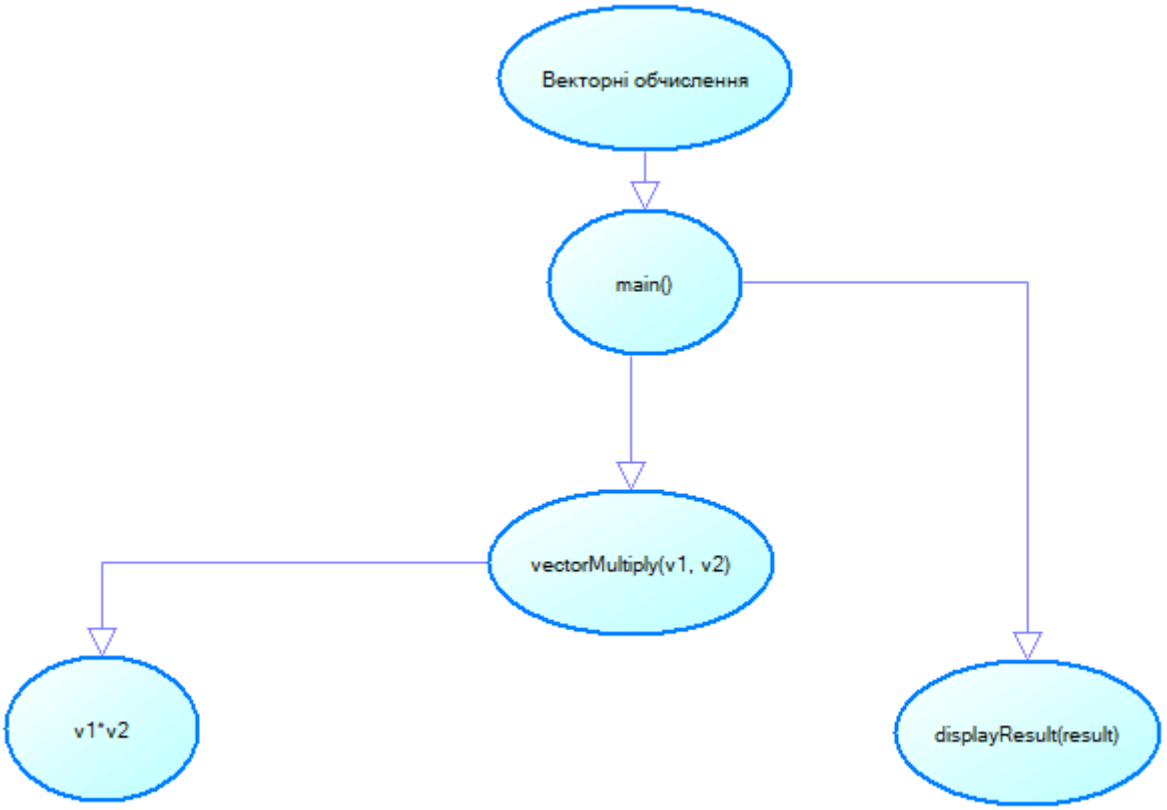

Рисунок 3.2.5.1 - Дерево функцій векторних обчислень

3.2.6 Синхронізація та узгодження

Опис:

Синхронізація та узгодження забезпечують правильну взаємодію між потоками або процесами при доступі до спільних ресурсів, запобігаючи конфліктам і забезпечуючи узгодженість даних. Для цього використовують блокування (locks), які гарантують, що тільки один потік чи процес може одночасно працювати з конкретним ресурсом, забезпечуючи таким чином

правильну послідовність операцій. Семафори ж контролюють доступ до обмежених ресурсів, дозволяючи визначеній кількості потоків або процесів одночасно використовувати ресурс, що сприяє більш ефективному розподілу ресурсів і синхронізації роботи між потоками чи процесами. Таким чином, блокування і семафори є основними інструментами для реалізації синхронізації та узгодження в багатопотокових і багатопроцесорних середовищах.[34]

Програмна реалізація:

У,цій програмі counter є глобальною змінною, яка починається зі значенням 0. Функція incrementCounter призначена для інкрементування цієї змінної на 1000000 одиниць через цикл.

При запуску програми два виклики setTimeout(incrementCounter, 0); створюють два асинхронних потоки для виконання incrementCounter. Оскільки JavaScript в однопотоковому середовищі, ці дві функції виконуються практично паралельно.

Після запуску обох функцій, через 1 секунду запускається третій setTimeout, який встановлює текст для елементу з id result7 в HTML-документі, показуючи значення counter.

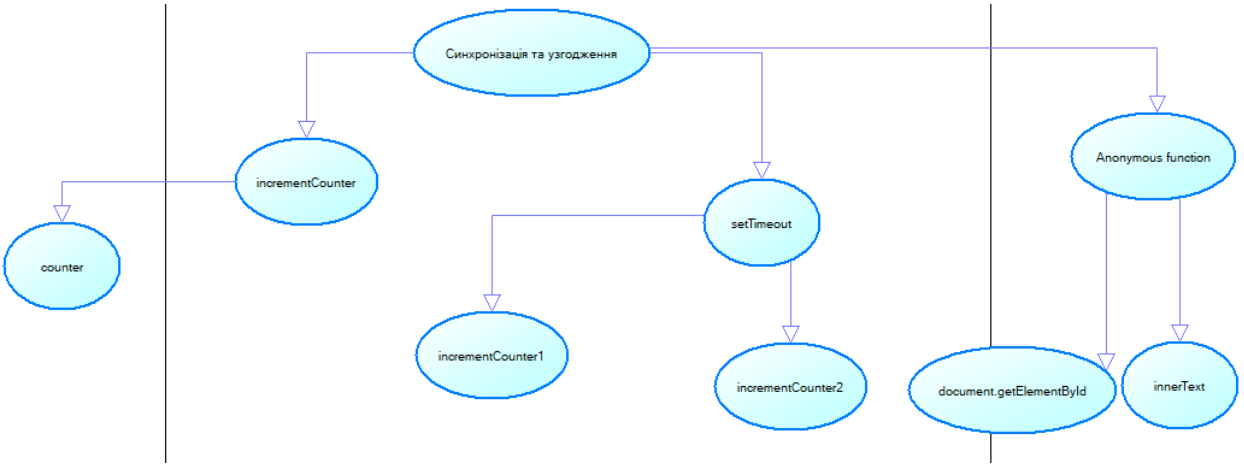

Рисунок 3.2.6.1 - Дерево функцій синхронізації та узгоджень

3.2.7 Інформація про клієнтів

Опис:

JS Heap (JavaScript Heap) — це частина пам'яті, яку використовує JavaScript-двигун для зберігання об'єктів та даних, що використовуються у програмі. JavaScript Heap є динамічною областю пам'яті, де зберігаються всі об'єкти, функції та інші дані, створені під час виконання JavaScript-коду. Ключові характеристики та особливості JS Heap:

- Купа (Heap) vs. Стек (Stack):

Heap використовується для динамічного розподілу пам'яті, тобто для об'єктів, створених в процесі виконання програми.

Stack використовується для зберігання примітивних типів даних та викликів функцій.

- Управління пам'яттю:

JavaScript автоматично керує пам'яттю за допомогою збирача сміття (Garbage Collector). Він автоматично видаляє об'єкти, які більше не використовуються, звільняючи пам'ять для нових об'єктів.

- Робота з об'єктами:

Всі складні типи даних (масиви, об'єкти, функції тощо) зберігаються в Heap. Це дозволяє програмі динамічно змінювати структуру та об'єм даних.

- Збирання сміття (Garbage Collection):

Збирання сміття працює за принципом пошуку та звільнення об'єктів, які більше не мають посилань з інших об'єктів або змінних.

- Проблеми з пам'яттю:

Пам'яткові витоки (Memory Leaks) виникають, коли об'єкти, які більше не потрібні, не звільняються збирачем сміття, що може призвести до збільшення використання пам'яті та уповільнення роботи програми.

Сокети — назва програмного інтерфейсу для забезпечення обміну даними між процесами. Процеси при такому обміні можуть виконуватися як на одній ЕОМ, так і на різних ЕОМ, пов'язаних між собою мережею. Сокет - абстрактний об'єкт, що представляє кінцеву точку з'єднання.

Слід розрізняти клієнтські і серверні сокети. Клієнтські сокети грубо можна порівняти з кінцевими апаратами телефонної мережі, а серверні - з

комутаторами. Клієнтський додаток (наприклад, браузер) використовує лише клієнтські сокети, а серверний (наприклад, вебсервер, якому браузер посилає запити) - як клієнтські, так і серверні сокети.

Інформація клієнта,які будуть описані:

1. IP адреса: Унікальний числовий ідентифікатор комп'ютера в мережі, який використовується для комунікації з іншими комп'ютерами або серверами.

2. Операційна система: Програмне забезпечення, що керує ресурсами комп'ютера і надає користувачу інтерфейс для взаємодії з комп'ютером та програмами.

3. Кількість процесорів: Кількість фізичних або логічних процесорів (ядер), доступних на комп'ютері для виконання обчислювальних завдань паралельно.

4. Загальна пам'ять: Обсяг оперативної пам'яті, доступний для програм та операційної системи для забезпечення швидкодії і продуктивності.

5. Використана пам'ять JS Heap: Кількість оперативної пам'яті, яку використовує область пам'яті JS Heap для зберігання об'єктів і данних під час виконання JavaScript програм.

6. Загальна пам'ять JS Heap: Максимальний обсяг пам'яті, який може бути використаний для JS Heap, встановлений для виконання JavaScript коду.

7. Ліміт пам'яті JS Heap: Максимальний обсяг пам'яті, який може бути виділений для JS Heap під час виконання JavaScript програми, що обмежує обсяг доступної пам'яті для JavaScript.

Програмна реалізація:

Програма,представляє собою клієнтську частину програми, яка використовує бібліотеку socket.io для взаємодії з сервером за допомогою WebSocket протоколу. Давайте розберемо деталі і особливості:

1. Підключення до сервера:

Створюється з'єднання з сервером, яке встановлюється через WebSocket на адресу http://25.28.183.168:3000. Цей адрес може бути змінений на IP-адресу вашого сервера, який працює через Hamachi або іншу VPN мережу.

2. Отримання інформації про клієнта:

Формується об'єкт clientInfo, який містить інформацію про клієнта. Зокрема, використовуються властивості navigator.platform, navigator.deviceMemory і navigator.hardwareConcurrency для отримання типу ОС, загальної пам'яті пристрою та кількості процесорних ядер. Якщо інформація про пам'ять JavaScript (використовуючи performance.memory) доступна, вона також додається до об'єкту.

3. Надсилання інформації на сервер:

Після підключення до сервера ('connect' event), відбувається відправка об'єкта clientInfo на сервер за допомогою події 'clientInfo'. Таким чином, сервер може отримати інформацію про клієнта та використовувати її за потреби.

4. Обробка відключення від сервера:

Якщо відбувається відключення від сервера ('disconnect' event), в консоль виводиться повідомлення про це.

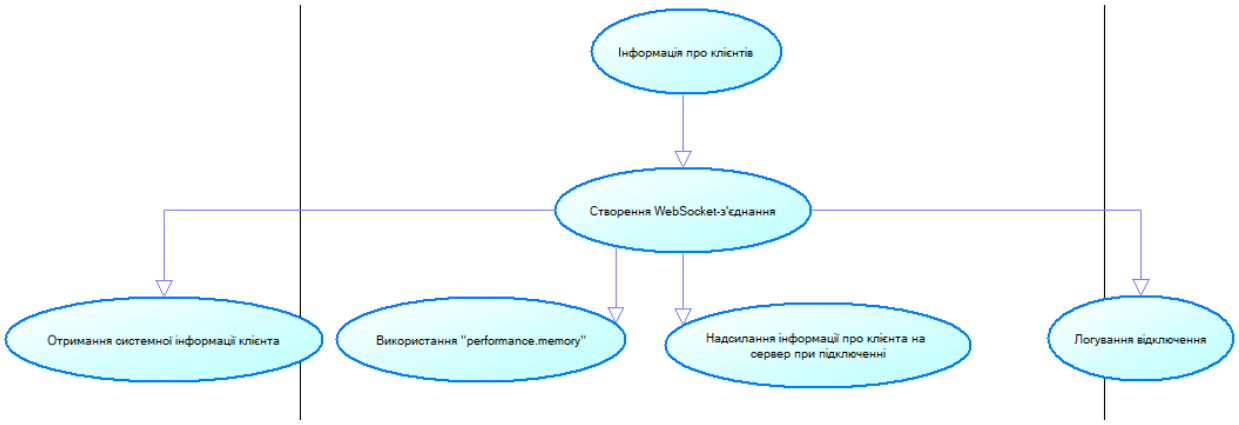

Рисунок 3.2.7.1 - Дерево функцій інформації про клієнтів

3.2.8 Клієнт-серверний веб-зв’язок

Опис:

Програма,буде,використовуватися для створення простого серверного додатку з можливістю взаємодії в реальному часі з клієнтами через веб-сокети,

що дозволяє реалізувати різноманітні застосунки, такі як чати, ігри або спільна робота з даними.

Програмна реалізація:

Програма є серверною аплікацією на Node.js, яка використовує Express для створення веб-сервера і Socket.IO для реалізації багатоканального обміну даними в реальному часі з клієнтами. Основні компоненти та їх функції такі:

1. Express: Це фреймворк для обробки HTTP-запитів. В даному випадку, app є екземпляром Express.
2. http.createServer: Створює HTTP-сервер, який передається екземпляру Express (app).
3. Socket.IO: Бібліотека для побудови додатків, які підтримують багатоканальний обмін даними в реальному часі. Вона працює поверх HTTP і дозволяє бідірекціональну комунікацію між веб-сервером і клієнтами через веб-сокети.
4. Робота з файлами і каталогами:
   - fs.readdir: Читає зміст каталогу з HTML-файлами.
   - fs.readFileSync: Синхронно читає вміст кожного HTML-файлу.
   - Файли зчитуються з вказаної директорії (calculationsDir) і додаються до відповіді сервера.
5. Express маршрутизація:
   - Обробник для шляху '/', який відповідає на HTTP GET-запити.
   - Відповідь сервера формується з HTML-контенту, який складається з вмісту прочитаних HTML-файлів і списку підключених клієнтів.
6. Socket.IO події:
   - 'connection': Спрацьовує при підключенні нового клієнта.

- 'clientInfo': Очікує інформацію про клієнта від клієнта, яку він передає при підключенні. Ця інформація зберігається в connectedClients разом із socket.id.
- 'disconnect': Спрацьовує при відключенні клієнта, видаляє інформацію про клієнта з connectedClients.

7. Серверний запуск:
   - Визначені константи PORT і HOST, які вказують порт і IP-адресу, на яких запускається сервер. У цьому прикладі HOST використовує IP-адресу з Hamachi, що дозволяє доступ до сервера з зовнішньої мережі.

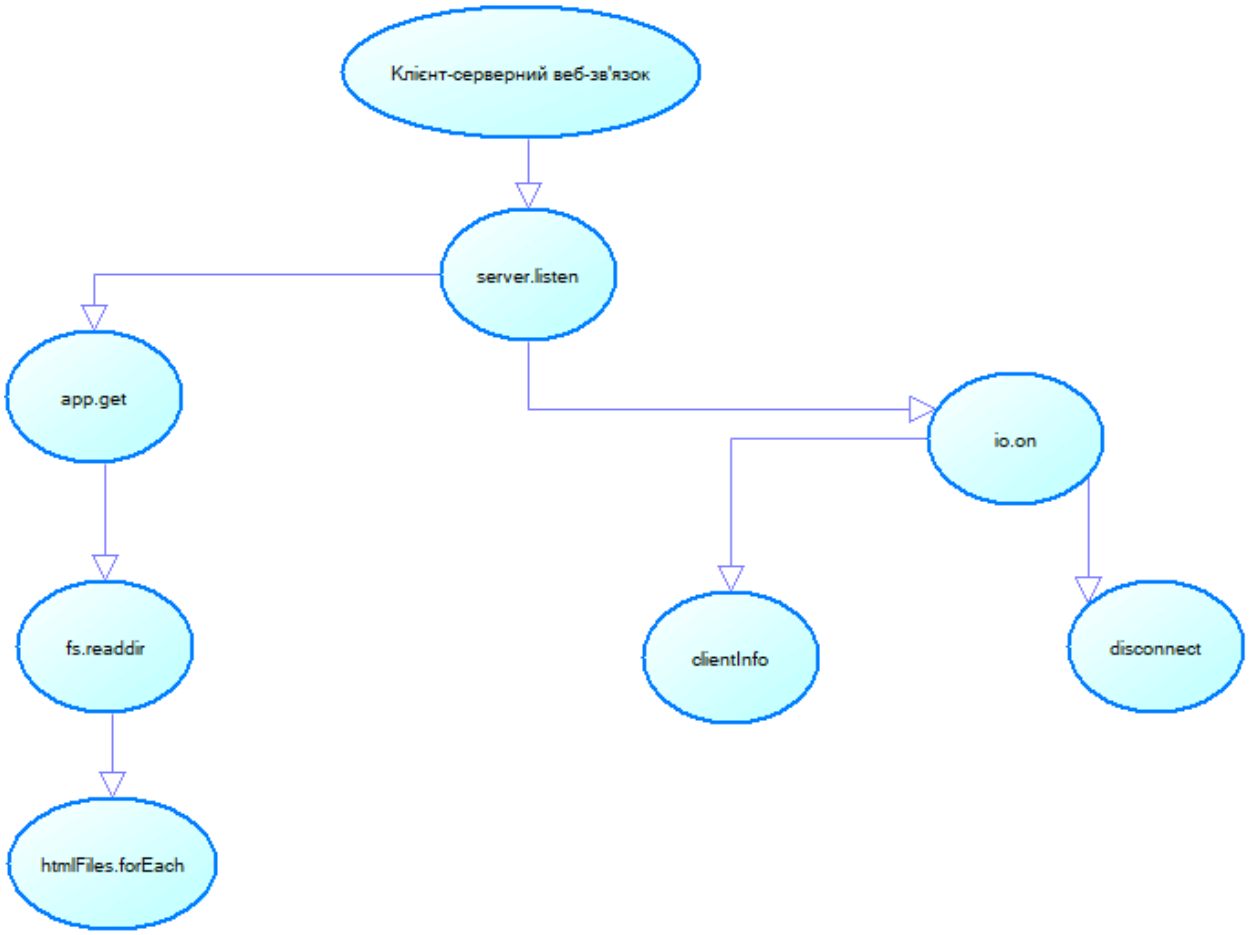

Рисунок 3.2.8.1 - Дерево функцій клієнт – серверного веб – зв’язку

3.2.9 Підключення клієнтів

Після,встановлення програми Hamachi,треба створити обліковий запис,використовуючи електронну пошту,та пароль,після цього,з’явиться головне вікно програми(так,само,для інших клієнтів).

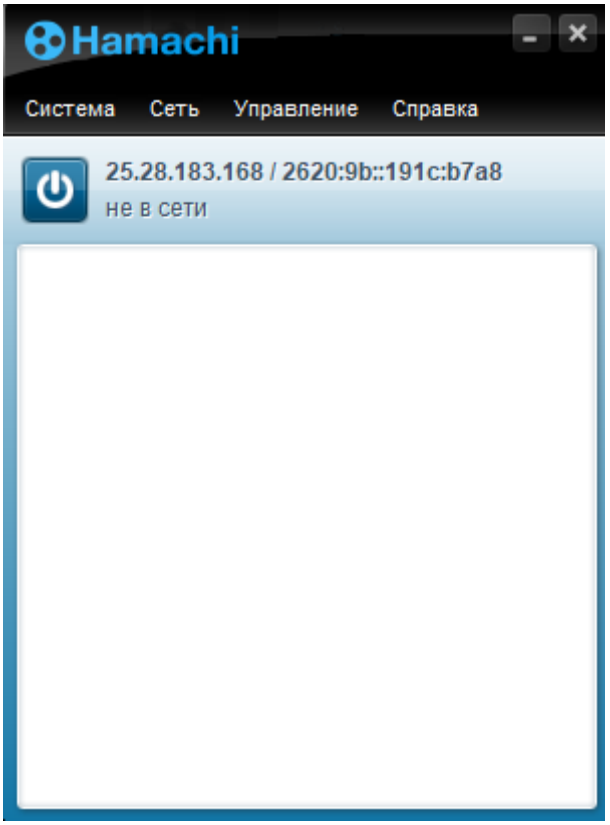

Рисунок 3.2.9.1 – Головне вікно Hamachi

У,головному вікні,треба обрати вкладку “Мережа\Створити нову мережу…”.

Після,чого створюємо ідентифікатор мережі,та його пароль.

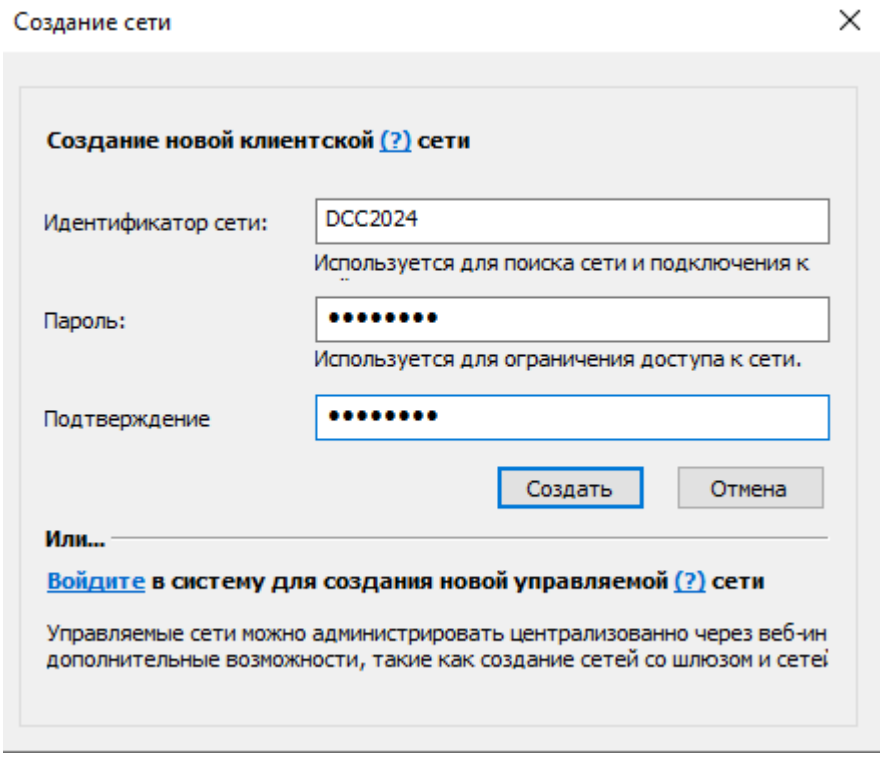

Рисунок 3.2.9.2 – Створення локальної мережі

Клієнти,які завантажили,та створили обліковий запис Hamachi,повинні,у головному вікні, “Мережа\Підключитись до існуючої мережі…”,вказавши ідентифікатор,та пароль існуючої мережі.

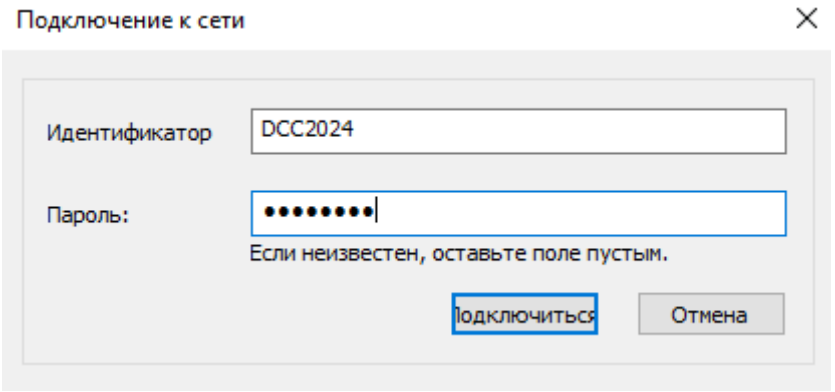

Рисунок 3.2.9.3 – Підключення,до локальної мережі

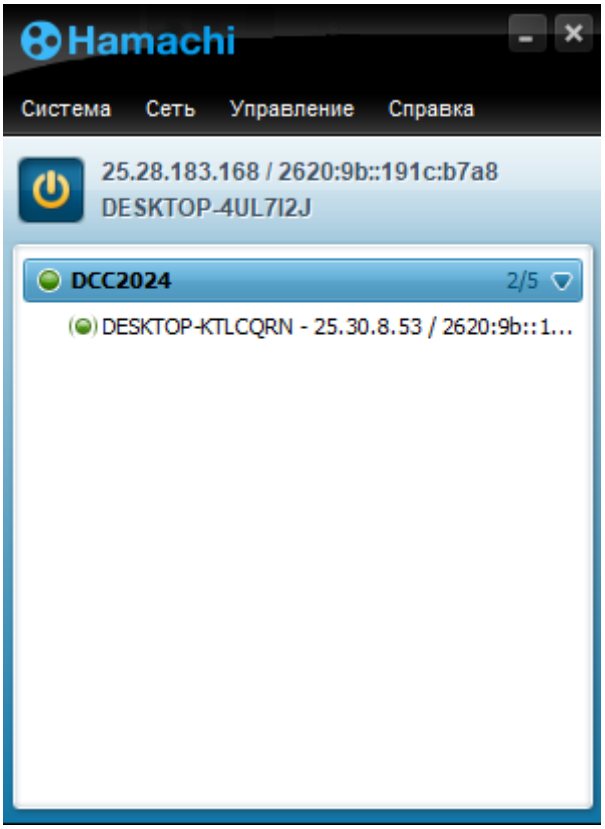

Рисунок 3.2.9.4 – Дві комп’ютери підключені,через Hamachi

Перед,наступними діями хост,та клієнти повинні,увійти,до “Панель керування”,а потім зайди,до “Система та безпека\Брандмауер Захисника Windows\Додаткові параметри\Правила для вхідних підключень”,треба дозволити програмам Hamachi,та Node.js,дозвіл,до вхідних підключень,бо наступні дії,буде блокувати файрвол.

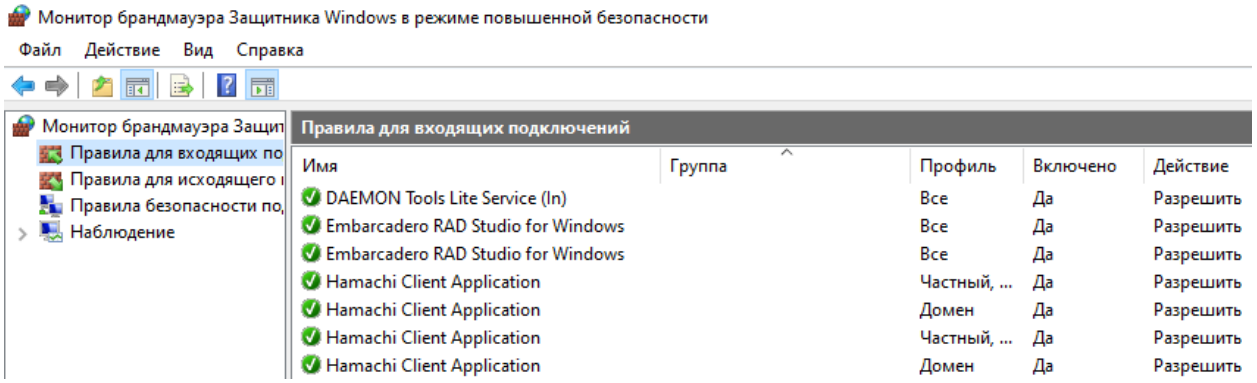

Рисунок 3.2.9.5 – Дозвіл брандмауера,на Hamachi

Рисунок 3.2.9.6 – Дозвіл брандмауера,на Node.js

У,Hamachi,може зробити,тест обміну пакетами:

Обмін пакетами з IP-адресою 25.30.8.53 за допомогою команди ping включає відправлені пакети розміром 32 байти кожен. Ось підсумок отриманих відповідей:

- Середній час відгуку: Більшість відповідей мають час відгуку 2 мс.
- Час до життя (TTL): Всі відповіді мають значення TTL, рівне 128, що вказує на відстань до цільового вузла в мережі.
- Варіація часу: Час відгуку в основному стабільний (2 мс).

Результати обміну пакетами:

1. Середній час відгуку (RTT):

- Найнижчий час: 2 мс
- Найвищий час: 6 мс

2. Рівень стабільності:

- Більшість відповідей мають однаковий час відгуку (2 мс), що свідчить про стабільність мережевого з'єднання.

Висновки:

- Загалом: З'єднання виглядає стабільним з невеликими та рідкими збоями.
- Мережеве середовище: Можливо, мережа знаходиться в хорошому стані, але час від часу можуть виникати незначні коливання затримок.

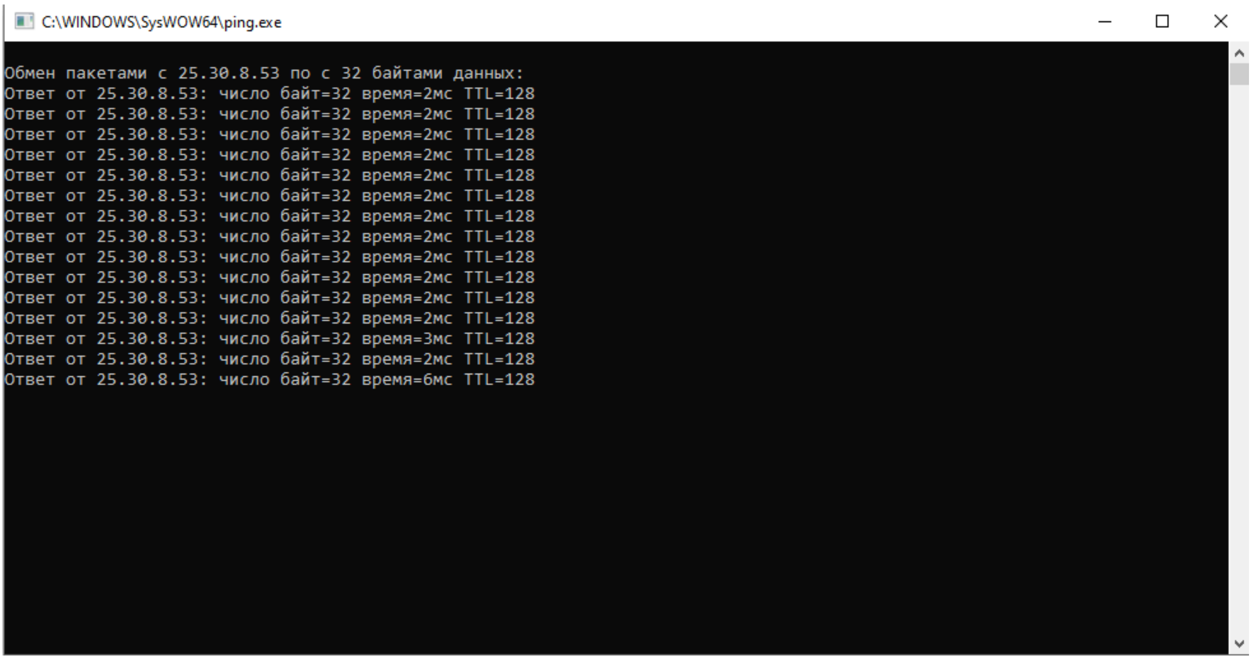

Рисунок 3.2.9.7 –Обмін пакетами даних,між двома комп’ютерами,через Hamachi

Після,усіх підготовок,останнє,що залишається запусти клієнт-серверний веб-зв’язок,використовуючи Node.js,у терміналі Visual Studio Code.

Примітка:перед,запуском програми,треба,у терміналі,ввести команди:

- "npm install ejs";
- "npm install express"
- "npm install express socket.io",щоб наша програма працювала.
- Тепер,запускаємо програму,у терміналі,вводячи команду "node DDP2024.js"

Коли,програма запустилося,заходимо,у браузер,вводячи URL-адресу:" http://25.28.183.168:3000",створиться сторінка,з введеною назвою URL-адреси,у сторінці,відкриті,уся інформація HTML-файлів,про паралельні обчислення,та інформація,про клієнтів.

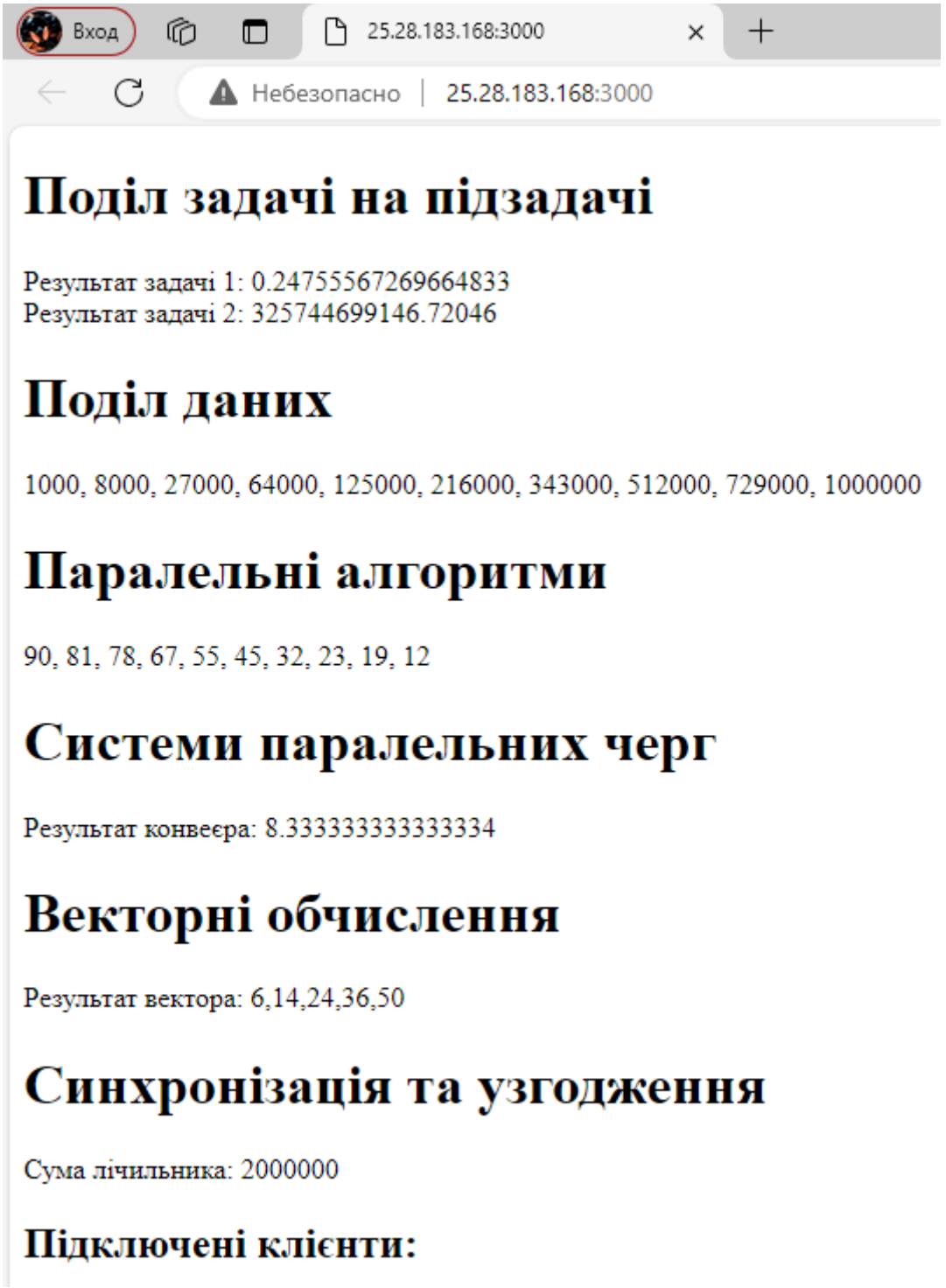

Рисунок 3.2.9.8 – Результат,зчитування HTML - файлів

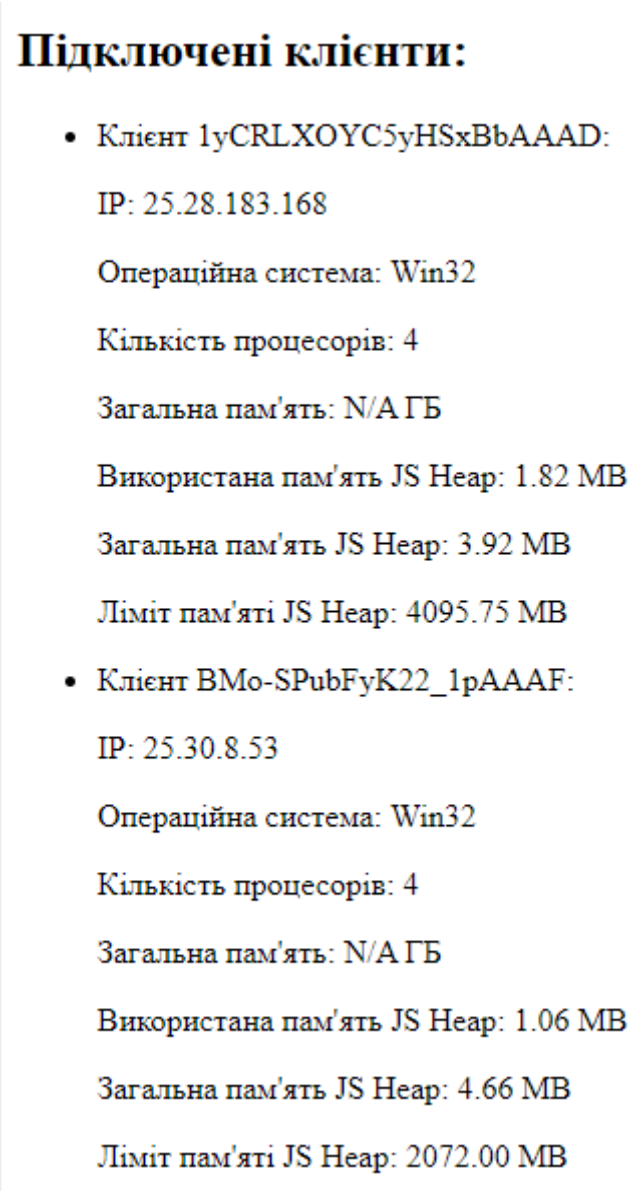

Рисунок 3.2.9.9 – Інформація,про клієнтів,у браузері

Також,інформація,про клієнтів,відображається,у терміналі.

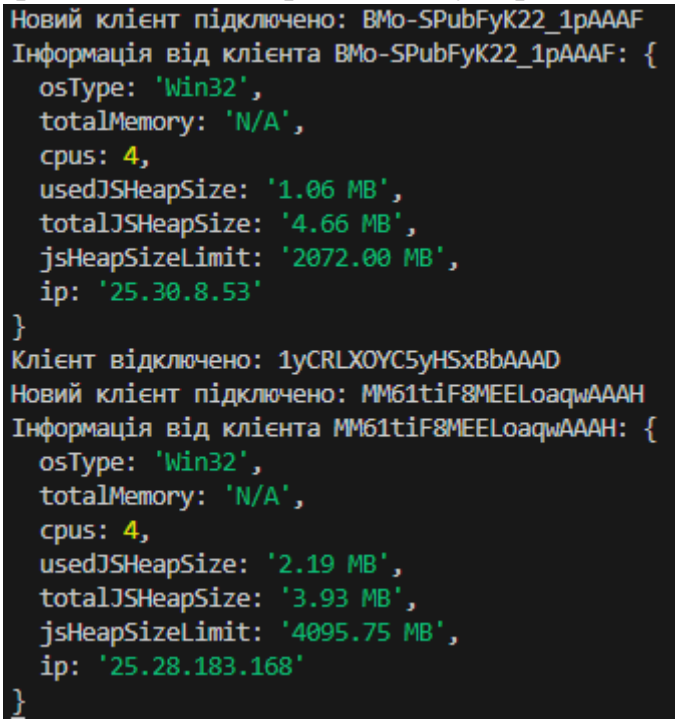

Рисунок 3.2.9.10 – Інформація,про кллієнтів,у терміналі

Якщо,натиснути,у терміналі комбінацію клавіш "Ctrl+C",тоді сервер припинить роботу.

Рисунок 3.2.9.11 – Припинення,роботи програми

Висновки

Тестова реалізація програми,створює сервер на базі Node.js, використовуючи Express для обслуговування HTML-сторінок і Socket.IO для реального часу комунікації з клієнтами. Сервер читає HTML-файли з вказаної директорії та відправляє їх вміст при запиті до кореневого URL, а також збирає і зберігає інформацію про підключених клієнтів, таку як IP-адреса, операційна система, кількість процесорів, пам'ять і використання пам'яті JavaScript heap. Ця інформація відображається на веб-сторінці, яка автоматично оновлюється при підключенні або відключенні клієнтів.

У контексті розподіленої обробки даних сервер виконує роль централізованого моніторингового вузла, що збирає і відображає метрики з різних клієнтів, які можуть виконувати розподілені обчислення. Це дозволяє адміністраторам і розробникам моніторити стан і продуктивність клієнтів, приймати рішення щодо перерозподілу завдань на основі поточного стану клієнтів і збирати дані для подальшого аналізу з метою оптимізації використання ресурсів і алгоритмів розподілених обчислень. Використання Hamachi для VPN-з'єднань забезпечує безпечний зв'язок між клієнтами і сервером, що є важливим для забезпечення надійного обміну даними в розподілених системах.

Список використаних джерел

1. Технології оброблення великих даних URL: https://ela.kpi.ua/bitstream/123456789/42206/1/%D0%9AonspLekts_Tekhnolohii-obroblennia-velykykh danykh_%D0%9Eleshchenko.pdf[29.02.2024]
2. Обробка даних URL: https://uk.wikipedia.org/wiki/%D0%A0%D0%BE%D0%B7%D0%BF%D0%BE%D0%B4%D1%96%D0%BB%D0%B5%D0%BD%D1%96_%D0%BE%D0%B1%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F[29.02.2024]
3. Розподілені обчислення URL: https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%B3%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85[29.02.2024]
4. Distributed computing URL: https://en.wikipedia.org/wiki/Distributed_computing[29.02.2024]
5. Folding@home URL: https://en.wikipedia.org/wiki/Folding@home[29.02.2024]
6. SLinCA@Home URL: https://uk.wikipedia.org/wiki/SLinCA@Home[29.02.2024]
7. PrimeGrid URL: https://uk.wikipedia.org/wiki/PrimeGrid[29.02.2024]
8. Apache Hadoop URL: https://en.wikipedia.org/wiki/Apache_Hadoop[29.02.2024]
9. Apache Spark URL: https://en.wikipedia.org/wiki/Apache_Spark[29.02.2024]
10. Amazon EMR URL: https://aws.amazon.com/emr/?nc1=h_ls[29.02.2024]
11. Google Cloud DataflowURL: https://cloud.google.com/dataflow[29.02.2024]

12. Apache Flink URL: https://en.wikipedia.org/wiki/Apache_Flink[29.02.2024]

13. Introducing of network URL: https://www.ece.uvic.ca/~itraore/elec567-13/notes/dist-03-4.pdf[29.02.2024]

14. GRID URL: https://uk.wikipedia.org/wiki/%D2%90%D1%80%D1%96%D0%B4[29.02.2024]

15. Introduction to GRID computing by Stefano Cozzini

16. Проміжне Програмне Забезпечення URL: https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BC%D1%96%D0%B6%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F[29.02.2024]

17. Обчислювальний кластер URL: https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B8%D0%B9_%D0%BA%D0%BB%D0%B0%D1%81%D1%82%D0%B5%D1%80[29.02.2024]

18. Cluster Computing High-Performance High-Availability and High-Throughput Processing on a Network of Computers URL: https://www.researchgate.net/publication/226533607_Cluster_Computing_High-Performance_High-Availability_and_High-Throughput_Processing_on_a_Network_of_Computers[29.02.2024]

19. URL: https://uk.wikipedia.org/wiki/%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D0%B0[29.02.2024]

20. Bus network URL: https://en.wikipedia.org/wiki/Bus_network[29.02.2024]

21. Ring network URL: https://en.wikipedia.org/wiki/Ring_network[29.02.2024]

22. Star network URL: https://en.wikipedia.org/wiki/Star_network[29.02.2024]

23. Mesh networking URL: https://en.wikipedia.org/wiki/Mesh_networking[29.02.2024]

24. VPN URL: https://uk.wikipedia.org/wiki/VPN[29.02.2024]

25. What is VPN URL: https://www.expressvpn.com/ru/what-is-vpn/vpn-for-dummies[29.02.2024]

26. Visual Studio Code URL: https://en.wikipedia.org/wiki/Visual_Studio_Code[29.02.2024]

27. Node.js URL: https://en.wikipedia.org/wiki/Node.js[29.02.2024]

28. LogMeIn Hamachi URL: https://en.wikipedia.org/wiki/LogMeIn_Hamachi[29.02.2024]

29. Task parallelism URL: https://en.wikipedia.org/wiki/Task_parallelism[29.02.2024]

30. Data parallelism URL: https://en.wikipedia.org/wiki/Data_parallelism[29.02.2024]

31. Parallel algorithm URL: https://en.wikipedia.org/wiki/Parallel_algorithm[29.02.2024]

32. Pipeline (computing) URL: https://en.wikipedia.org/wiki/Pipeline_(computing) [29.02.2024]

33. Vector (mathematics and physics) URL: https://en.wikipedia.org/wiki/Vector_(mathematics_and_physics) [29.02.2024]

34. Synchronization (computer science) URL: https://en.wikipedia.org/wiki/Synchronization_(computer_science)[29.02.2024]