Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Проміжне програмне забзпечення мультикомп'ютерної системи з топологією "кільце" однонаправленого зв'язку
Назва теми

КвРКІ 101051.21.01.01 ПЗ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Виконав: студент III курсу, група КІ2с-21-1 _____ Горобчишен Д.А.
Підпис Ініціали, прізвище

Керівник _____ Медзатий Д.М.
Підпис, дата Ініціали, прізвище

Нормоконтролер _____ Засорнова І.О.
Підпис, дата Ініціали, прізвище

До захисту допускаю:
Зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Т.О. Говорущенко
Підпис Ініціали, прізвище

«12» червня 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Освітній рівень бакалавр

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма «Комп'ютерна інженерія та програмування»

ЗАТВЕРДЖУЮ

Зав. кафедри Т.О.Говорущенко

" 10 " 01 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Горобчишену Денису Анатолійовичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Проміжне програмне забзпечення мультикомп'ютерної системи з топологією "кільце" однонаправленого зв'язку

Керівник проекту (роботи) Медзатий Д.М., к.т.н., доц.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 15.02.2024 р. № 8

2. Строк подання студентом проекту (роботи) на кафедру 01.06.2024 р.

3. Вихідні дані до проекту (роботи) Завдання на кваліфікаційну роботу

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

Теоритичні основи досліджуваної проблеми

Проєктування та вибір архітектури програмно-апаратного комплексу

Проєктування програмно-технічного засобу

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

UML-діаграма та діаграма залежностей

Схеми будов архітекстур мультикомпю'терних систем

Сценарії використання

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Засорнова І.О., доцент кафедри КІС		
Антиплагіат	Нічепорук А.О., доцент кафедри КІС		

7. Дата видачі завдання «_10_»_____01_____2024 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2024	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2024	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	01.03.2024	виконано
4	Робота над розділом 2 – – вибір компонентів для проєктування мультикомп'ютерної системи системи	01.04.2024	виконано
5	Робота над розділом 3 – проєктування мультикомп'ютерної системи типу кільце	29.04.2024	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2024	виконано
7	Попередній захист ВКР	30.05.2024	виконано
8	Захист ВКР на засіданні ЕК	Червень 2024 року	

Студент _____ Підпис _____ Д.А. Горобчишен Ініціали, прізвище

Керівник роботи _____ Підпис _____ Д.М. Медзатий Ініціали, прізвище

роботи)

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 101051.21.01.01 ПЗ	Пояснювальна записка	72		
			Графічні матеріали			
2		КвРКІ 101051.21.01.01 Е8	UML-діаграма та діаграма	1		
			залежностей			
3		КвРКІ 101051.21.01.01 Е8	Схеми будов архітектур	1		
			мультикомпю'терних			
			систем			
4		КвРКІ 101051.21.01.01 Е8	Сценарії використання	1		

КвРКІ 101051.21.01.01 ВП

Зм	Арк	№ докум	Підпис	Дата
Розробив		Горобчишен		
Перевір.		Медзатий		12.06
Н. контр.		Засорнова		
Затв.		Говорущенко		12.06

Відомість проекту

Літера	Аркуш	Арк
У	1	

ХНУ, КІ2с-21-1

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Проміжне програмне забзпечення мультикомп’ютерної системи з топологією “кільце” однонаправленого зв’язку».

Автор роботи: Горобчишен Денис Анатолійович.

Керівник роботи: Медзатий Дмитро Миколайович.

Пояснювальна записка: 72 с., 17 рис., 3 дод., 50 джерел.

Графічна частина: 3 креслення.

ТОПОЛОГІЯ “КІЛЬЦЕ”, МЕРЕЖЕВІ ТЕХНОЛОГІЇ, МУЛЬТИКОМП’ЮТЕРНА СИСТЕМА, БЕЗПЕКА МЕРЕЖІ , ПРОМІЖНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Метою дипломної роботи є створення проміжного програмного забезпечення для мультикомп'ютерної системи з топологією "кільце" та однонаправленим зв'язком .

Об’єктом дослідження є проміжне програмне забезпечення мультикомп'ютерної системи з топологією "кільце" та однонаправленим зв'язком..

Предметом дослідження є оцінка умов та особливостей застосування обладнання та проміжного програмного забезпечення у кільцевій мультикомп'ютерній системі з однонаправленим зв'язком.

Під час проведення цього дослідження було використано метод систематичного аналізу літератури для дослідження тематичної області з джерел текстової інформації.

Підпис студента

10.06.2024

Дата

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ЕОМ - електронно-обчислювальна машина

SIMD - Single Instruction, Multiple Data одиночний потік команд, множинний потік даних

MIMD - Multiple Instruction, Multiple Data множинний потік команд, множинний потік даних

MSIMD - Multiple Single Instruction, Multiple Data множинний одиночний потік команд, множинний потік даних

SISD - Single Instruction, Single Data одиночний потік команд, одиночний потік даних

SIMT - Single Instruction, Multiple Threads одиночний потік команд, множинний потік потоків

SPMD - Single Program, Multiple Data одиночна програма, множинний потік даних

ASIP - Application-Specific Instruction-set Processor процесор зі специфічним для програми набором команд

VLIW - Very Long Instruction Word дуже довге інструкційне слово

EPIC - Explicitly Parallel Instruction Computing явно паралельна обчислювальна архітектура

LAN - Local Area Network локальна мережа

USB - Universal Serial Bus універсальна послідовна шина

HDMI - High-Definition Multimedia Interface мультимедійний інтерфейс високої чіткості

Wi-Fi - Wireless Fidelity бездротова точність

WLAN - Wireless Local Area Network бездротова локальна мережа

VM - Virtual Machine віртуальна машина

ОС - Operating System операційна система

МКС - мультикомп'ютерна система

ВСТУП

Значущість дослідження полягає в тому, що мультикомп'ютерні системи з топологією "кільце" та однонаправленим зв'язком відіграють важливу роль у сучасних інформаційних технологіях. Вони застосовуються в різних сферах, включаючи мережі передачі даних, виробничі системи та телекомунікаційні інфраструктури.

Метою дипломної роботи є створення проміжного програмного забезпечення для мультикомп'ютерної системи з такою топологією, а також вивчення умов та особливостей його застосування. Крім того, метою є оцінка механізмів обробки інформації у такій системі з метою забезпечення ефективного функціонування.

Об’єктом дослідження є функціонування проміжного програмного забезпечення у мультикомп'ютерній системі з топологією "кільце" та однонаправленим зв'язком.

Предметом дослідження є оцінка різних режимів застосування проміжного програмного забезпечення в такій системі з урахуванням конкретних вимог та умов топології "кільце" та однонаправленого зв'язку.

Завдяки своїм унікальним властивостям, мультикомп'ютерні системи з топологією "кільце" та однонаправленим зв'язком забезпечують високу швидкість та надійність передачі даних. Це досягається завдяки тому, що кожен вузол в системі передає дані лише у визначеному напрямку, що мінімізує можливість колізій та втрат інформації. Такі системи широко застосовуються у високонавантажених середовищах, де необхідна швидка та безперебійна комунікація між численними компонентами. Зокрема, вони знаходять своє застосування у великих дата-центрах, де обробляються великі обсяги інформації, а також у виробничих системах, де важлива синхронізована робота різних частин виробничого процесу.

Іншим важливим аспектом дослідження є розробка та оптимізація проміжного програмного забезпечення для таких систем. Проміжне ПЗ відіграє

ключову роль у забезпеченні ефективного обміну даними між вузлами системи, зменшуючи затримки та покращуючи загальну продуктивність. Вивчення різних режимів застосування цього програмного забезпечення дозволяє виявити оптимальні стратегії для різних сценаріїв використання, забезпечуючи гнучкість та адаптивність системи до змінних умов. Крім того, аналіз механізмів обробки інформації сприяє створенню більш надійних та ефективних алгоритмів, що можуть бути використані у різних галузях, від телекомунікацій до промислової автоматизації.

1 ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖУВАНОЇ ПРОБЛЕМИ

1.1 Аналіз предметної області і виявлення наявних проблем і завдань

Максимальна продуктивність обчислювальної системи забезпечується використанням високошвидкісних елементів та одночасним виконанням багатьох операцій [1].

За класифікацією Флінна, розрізняють два типи: SIMD (рисунок 1.1) (Single Instruction Multiple Data): один потік команд для множини потоків даних. MIMD (рисунок 1.2) (Multiple Instruction Multiple Data): множинний потік команд та даних.

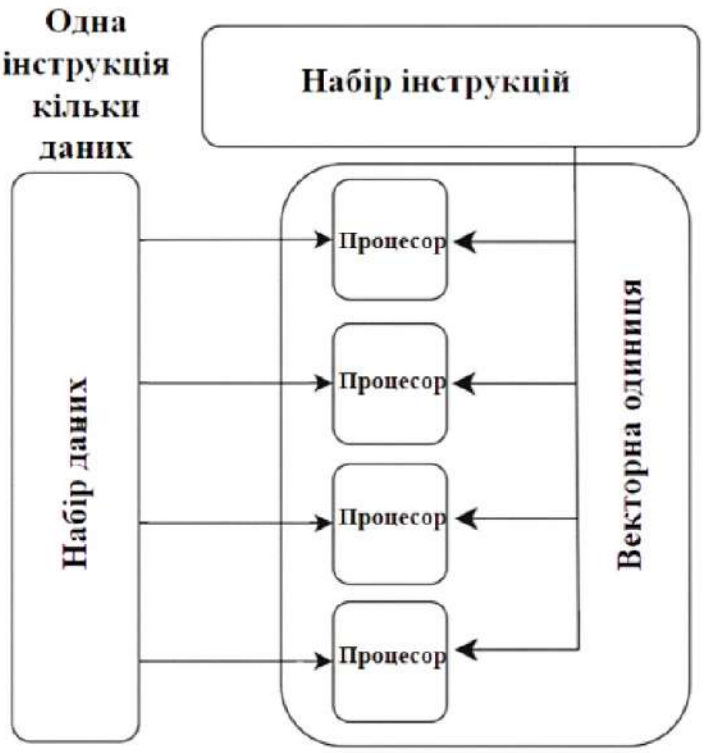


Рисунок 1.1 – SIMD архітектура

Існують такі основні пити архітектури:

– конвеєрна та векторна обробка: роздільне виконання операції в кілька етапів з передачею даних;

– машини типу SIMD: складаються з великої кількості ідентичних процесорних елементів з власною пам'яттю. Всі процесори виконують одну й ту ж програму;

– машини типу MIMD: кожен процесор виконує свою незалежну програму. Існують два типи: мультипроцесори з загальною пам'яттю (сильно пов'язані) доступ до пам'яті даних та команд для всіх процесорів. Другий тип це мультипроцесори з розподіленою пам'яттю (слабко пов'язані): кожен процесор має власну пам'ять;

– багатопроцесорні машини з SIMD процесорами: супер-ЕОМ з векторними процесорами (MSIMD), можливість використовувати одночасно векторні операції та гнучкість MIMD архітектури.

Деякі додаткові класифікації:

– SISD (Single Instruction Single Data): це традиційний однопроцесорний підхід, де лише одна команда виконується над одними даними в один момент часу;

– SIMT (Single Instruction Multiple Threads): цей підхід схожий на SIMD, але з деякими відмінностями. Кілька потоків виконують одну і ту ж команду, але кожен потік може мати свій власний набір даних;

– SPMD (Single Program Multiple Data): в цьому випадку багато потоків виконують одну програму, але кожен потік може виконувати різні частини програми або мати свої власні дані;

– ASIP (Application-Specific Instruction-set Processor): ця класифікація вказує на процесори, які спеціалізовані для конкретних завдань або додаткових інструкцій, що підтримують конкретний тип обчислень;

– VLIW (Very Long Instruction Word): це архітектурний підхід, де кожна інструкція містить кілька операцій, які можуть виконуватися паралельно, але відмінно від SIMD, тут це може відбуватися на рівні інструкцій, а не на рівні даних;

– EPIC (Explicitly Parallel Instruction Computing): схожий на VLIW, але відзначається використанням компіляторів та архітектурних засобів для ефективного виявлення та використання паралелізму.

Мультикомп'ютерна система - це складна структура, що включає в себе різноманітні компоненти для ефективної обробки і обміну даними.

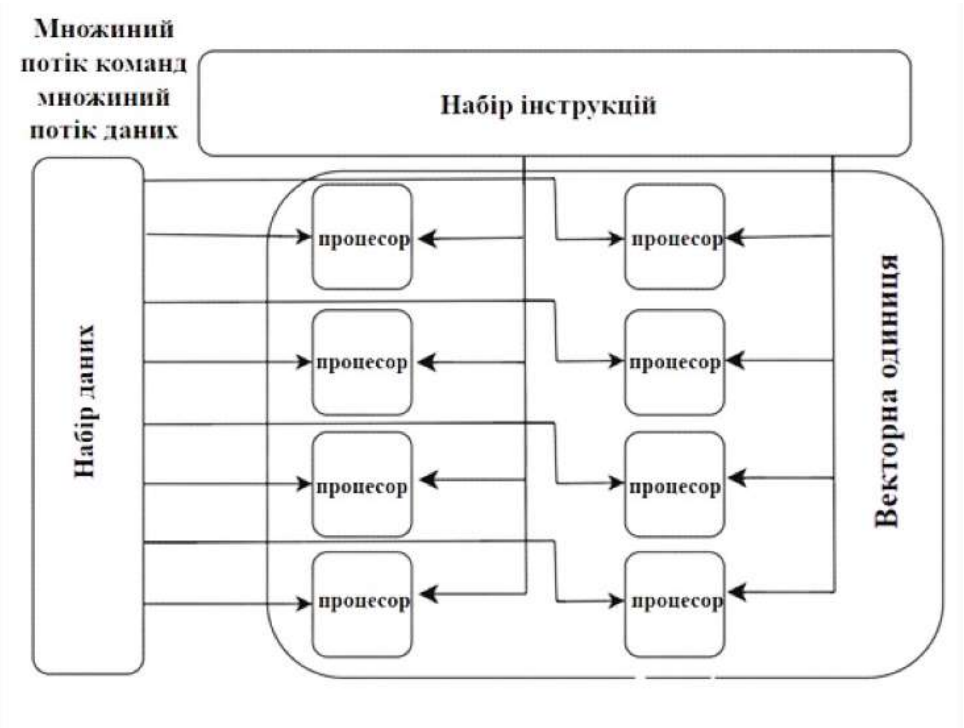


Рисунок 1.2 – MIMD архітектура

Основні складові цієї системи можна розділити на кілька ключових категорій:

– персональні комп'ютери (ПК). Це комп'ютери, які зазвичай використовуються однією особою для особистих або робочих завдань. Вони можуть бути розпізнані за своєю компактністю та призначенням для індивідуального використання. Персональні комп'ютери включають настільні комп'ютери, ноутбуки, планшети та інші подібні пристрої;

– спеціалізовані пристрої. Ця категорія включає в себе різноманітні комп'ютерні системи, призначені для конкретних завдань або областей використання;

– сервери: спеціальні комп'ютери, які забезпечують ресурси та послуги іншим комп'ютерам, зазвичай через мережу;

– робочі станції. Потужні комп'ютери, спроектовані для високоефективних обчислювальних завдань, таких як графічний дизайн, інженерія, обробка відео тощо;

– кластери. Групи пов'язаних між собою комп'ютерів, які працюють разом для розв'язання складних завдань;

– вбудовані системи. Комп'ютерні системи, вбудовані безпосередньо в інші пристрої або системи, такі як мікроконтролери в автомобілях, холодильниках, телевізорах тощо.

Мережеве з'єднання:

– ethernet (LAN). Використовується для провідного з'єднання в локальних мережах (LAN). Кабелі Ethernet можуть бути використані для підключення комп'ютерів, принтерів, маршрутизаторів та інших пристроїв;

– USB (Universal Serial Bus). Широко використовується для підключення різноманітних пристроїв, включаючи принтери, камери, зовнішні накопичувачі, клавіатури та миші;

– HDMI (High-Definition Multimedia Interface). Використовується для передачі відео та аудіо сигналів між пристроями, такими як комп'ютери, телевізори, монітори та аудіосистеми [2];

– Wi-Fi. Бездротовий стандарт для локального з'єднання в мережах (WLAN). Використовується для підключення комп'ютерів, смартфонів, планшетів та інших пристроїв до мережі без потреби в фізичних кабелях;

– bluetooth. Бездротовий протокол короткого з'єднання, який використовується для підключення різних пристроїв, таких як гарнітури, клавіатури, миші та інші периферійні пристрої;

– cellular (мобільний). Забезпечує з'єднання через мобільні мережі, такі як 4G або 5G. Використовується для забезпечення Інтернет-з'єднання на мобільних пристроях, таких як смартфони та планшети [3].

Програмне забезпечення:

– операційна система (ОС). Це базовий шар програмного забезпечення, який забезпечує основні служби та ресурси для інших програм і користувачів. Операційна система відповідає за управління процесами, пам'яттю, введенням-виведенням, файловою системою та іншими базовими аспектами комп'ютерної системи;

– диспетчер завдань. Це програмне забезпечення, яке відповідає за розподіл та управління ресурсами комп'ютерної системи між різними процесами та задачами. Диспетчер завдань визначає порядок виконання процесів і контролює доступ до ресурсів [4];

– системи управління ресурсами. Це програми, які моніторять та управляють використанням ресурсів, таких як центральний процесор, пам'ять, мережа та інші, для забезпечення ефективності та оптимальної роботи системи;

– мережеве управління. Для систем, які взаємодіють в мережі, існують системи управління мережею, які дозволяють моніторити та керувати ресурсами, а також забезпечують безпеку та ефективність мережевих операцій;

– системи моніторингу та журналювання: для виявлення та вирішення проблем в системі використовуються системи моніторингу, які збирають дані про продуктивність, стан ресурсів та інші параметри. Журналювання дозволяє фіксувати події та взаємодії для подальшого аналізу;

– віртуальна машина (VM). Це програмне забезпечення, яке імітує апаратне забезпечення та дозволяє виконувати програми в ізольованому середовищі. VM може бути використана для імітації апаратного забезпечення різних архітектур, що дозволяє виконувати програми, написані для однієї архітектури, на різних платформах;

– контейнеризація. Однією з форм віртуальних машин є контейнери, такі як Docker. Контейнери дозволяють запускати програми в ізольованому середовищі, але вони використовують спільний ядро операційної системи [5].

Операційна система:

– розподілена операційна система (distributed operating system). Це операційна система, яка керує розподіленими ресурсами та взаємодіє з різними вузлами в мережі. Вона дозволяє координувати та керувати роботою розподілених програм та сервісів;

– менеджер ресурсів. Операційна система в розподіленій системі відповідає за управління ресурсами, такими як процесорний час, пам'ять, мережа, для оптимальної роботи системи;

– механізми комунікації та синхронізації. Операційна система визначає правила взаємодії між вузлами, забезпечуючи ефективну обмін даними та взаємодію розподілених компонентів.

Крім того, важливо враховувати аспекти безпеки, моніторингу та системи резервного копіювання як додаткові елементи для забезпечення надійності та безпеки мультикомп'ютерної системи. Взаємодія всіх цих складових елементів допомагає створити ефективну та функціональну мультикомп'ютерну інфраструктуру.

В мультикомп'ютерних систем, є важливим також розгляд аспектів паралельного програмування та взаємодії між процесами. У мультикомп'ютерних системах, де використовуються різні комп'ютери чи обчислювальні вузли, ефективне використання ресурсів вимагає розподілення завдань та обміну даними між цими вузлами.

Паралельне програмування використовується для розділення завдань на підзадачі, які виконуються паралельно на різних обчислювальних ресурсах. Це може бути особливо важливим у мультикомп'ютерних системах, де різні вузли можуть працювати над різними частинами великого обчислення [6].

Важливо також зазначити питання безпеки в мультикомп'ютерних системах, оскільки вони зазвичай взаємодіють через мережі. Заходи безпеки, такі як шифрування даних, аутентифікація та авторизація, стають ключовими для захисту важливої інформації під час обміну між різними частинами системи.

Також важливо враховувати тенденції у розвитку мультикомп'ютерних систем, такі як використання хмарних технологій, що дозволяють розширювати обчислювальні ресурси за потребою, а також впровадження нових технологій, таких як квантові обчислення, які можуть змінити парадигму обчислень в майбутньому [7].

Сфери застосування мультикомп'ютерних систем:

– наукові дослідження. Мультикомп'ютерні системи використовуються для складних наукових досліджень, таких як моделювання клімату, геноміка та астрофізика;

– інженерія. Мультикомп'ютерні системи використовуються для комп'ютерного проектування, аналізу та моделювання в таких галузях, як авіація, автомобілебудування та машинобудування;

– фінанси. Мультикомп'ютерні системи використовуються для аналізу фінансових ринків, моделювання ризиків та прогнозування цін;

– медицина. Мультикомп'ютерні системи використовуються для обробки медичних зображень, аналізу даних пацієнтів та розробки нових ліків.

Тенденції розвитку:

– хмарні технології. Мультикомп'ютерні системи все частіше використовуються в хмарних середовищах, що дозволяє користувачам отримувати доступ до обчислювальних ресурсів за потребою;

– квантові обчислення. Квантові обчислення мають потенціал значно збільшити продуктивність мультикомп'ютерних систем;

– штучний інтелект. Мультикомп'ютерні системи використовуються для розробки та використання систем штучного інтелекту.

1.2 Порівняльний аналіз переваг та недоліків існуючих рішень

Мультикомп'ютерна система – це не просто сукупність комп'ютерів [8]. Це оркестр, де кожен інструмент – це потужний процесор, а диригент – це мережа, що

об'єднує їх у єдиний організм. Цей організм володіє значними перевагами, але й має свої нюанси, які потребують уваги.

Переваги:

– підвищення продуктивності. МКС може значно пришвидшити обробку даних, розбиваючи завдання на частини та розподіляючи їх між комп'ютерами;

– підвищення доступності. Якщо один комп'ютер виходить з ладу, інші продовжують роботу, забезпечуючи безперебійну роботу системи;

– резервне копіювання даних. Кожен комп'ютер МКС може мати свою копію даних, що робить їх більш стійкими до втрати;

– масштабованість. МКС можна легко розширювати, додаючи нові комп'ютери для збільшення потужності.

Недоліки:

– висока вартість. Створення та підтримка МКС потребує значних витрат на обладнання, програмне забезпечення та адміністрування;

– складність. Управління МКС може бути складним завданням, адже потрібно координувати роботу багатьох комп'ютерів;

– проблеми з безпекою. МКС може бути більш вразливою до кібератак, адже кожен комп'ютер – це потенційна точка входу для зловмисників;

– синхронізація даних. Забезпечення узгодженості даних між комп'ютерами МКС може бути складним завданням;

– сумісність програмного забезпечення. Різні комп'ютери МКС можуть мати різні операційні системи та програмне забезпечення, що може призвести до проблем з сумісністю.

Додаткові переваги та недоліки використання МКС:

– енергоефективність. МКС може споживати менше електроенергії в порівнянні з одним потужним комп'ютером, оскільки завдання розподіляються між багатьма пристроями [9];

– резервне відновлення. В разі виходу з ладу одного з комп'ютерів, його завдання може бути автоматично перенаправлене на інші пристрої, забезпечуючи безперебійну роботу системи;

– інновації. Можливість використання різних комп'ютерів у складі МКС дозволяє впроваджувати нові технології та оновлювати обладнання без зупинки всієї системи;

– обмежена взаємодія. Різні комп'ютери можуть використовувати різні архітектури та технології, що може ускладнити обмін даними та взаємодію між ними;

– залежність від мережі. МКС вимагає надійної мережної інфраструктури. При проблемах з мережею може виникнути велика затримка у виконанні завдань;

– обмеження масштабування. При досягненні певної точки розміру МКС, може виникнути складність у подальшому розширенні без значних змін в інфраструктурі;

– неоднорідність ресурсів. Різні комп'ютери можуть мати різну продуктивність, що ускладнює розподіл завдань та використання ресурсів ефективно.

Важливо ретельно зважити всі плюси та мінуси МКС перед її впровадженням. Її доцільно використовувати для задач, які потребують значної потужності та безперебійної роботи.

Для ефективної роботи мультикомп'ютерної системи важливо також враховувати такі аспекти:

– паралельність та розподілені алгоритми. Ідеальною ситуацією для МКС є можливість виконувати завдання паралельно та використовувати розподілені алгоритми, які розподіляють обчислення між різними компонентами системи;

– балансування навантаження. Система повинна ефективно розподіляти завдання між комп'ютерами, уникати перевантаження деяких процесорів та забезпечувати рівномірне використання ресурсів;

– механізми обміну даними. Система повинна мати ефективні механізми обміну даними між компонентами, такі як швидкі мережеві з'єднання та оптимізовані протоколи обміну інформацією;

– автоматизоване моніторинг та управління. Забезпечення постійного моніторингу стану системи, виявлення проблем та автоматизоване управління ресурсами можуть покращити ефективність та надійність.

– технології віртуалізації. Використання технологій віртуалізації може полегшити управління різними операційними системами та додатками, що працюють на комп'ютерах МКС;

– стандартизація та сумісність. Важливо встановлювати стандарти для апаратного забезпечення, мережевих протоколів та програмного забезпечення, щоб уникнути проблем зі сумісністю та спростити інтеграцію нових компонентів;

– системи безпеки та аутентифікації. Забезпечення високого рівня безпеки є критичним, оскільки МКС може бути предметом кібератак. Ефективна система аутентифікації та захисту даних є важливою складовою.

1.3 Підходи до вирішення задачі за темою дослідження

Мультикомп'ютерні системи в сучасному інформаційному суспільстві є невід'ємною частиною, сприяючи обробці та обміну великими обсягами даних між різними комп'ютерними системами. Однак перед тим, як зануритися у подробиці архітектури та топології мультикомп'ютерних систем, важливо розглянути ключові складові цього складного технологічного ландшафту [10].

Комп'ютери, що становлять основну частину мультикомп'ютерних систем, можуть приймати різні форми: персональні комп'ютери, спеціалізовані пристрої чи інші обчислювальні пристрої здатні обробляти дані. Їх робота стає можливою завдяки мережевому з'єднанню, яке визначає засоби зв'язку між цими комп'ютерами. Це може бути провідне чи бездротове з'єднання, залежно від конкретної реалізації системи.

Невід'ємною частиною мультикомп'ютерних систем є програмне забезпечення, яке відповідає за управління системою та забезпечує обмін даними між комп'ютерними системами. Це може бути програмне забезпечення розподіленої обробки даних, віртуальна машина, операційна система та інше.

Протоколи комунікації визначають правила передачі даних між комп'ютерами у мультикомп'ютерній системі. Популярні протоколи включають TCP/IP, UDP, HTTP, FTP та інші, забезпечуючи стандартизацію обміну інформацією.

Система керування ресурсами є ключовим елементом для ефективного функціонування мультикомп'ютерної системи. Вона відповідає за ефективне розподілення та управління ресурсами, такими як пам'ять, процесорний час та мережеві ресурси [11].

Застосунки, які використовують програми, є важливою частиною мультикомп'ютерних систем, забезпечуючи виконання різноманітних завдань відповідно до потреб користувачів.

Архітектура мультикомп'ютерної системи може включати системи з розподіленою пам'яттю, де кожен комп'ютер має свою локальну оперативну пам'ять, а доступ до неї відсутній для інших процесорів. Це вимагає створення копій вихідних даних на кожному процесорі.

Топологія мережі визначає спосіб взаємодії пристроїв та каналів у комп'ютерній мережі. Зіркова, лінійна, кільцева, деревоподібна та меш топології можуть бути використані в залежності від потреб та вимог конкретної системи.

Кільцева топологія, яка є однією з варіантів, включає підключення пристроїв в коло. Це забезпечує надійність та просту структуру, але може бути повільною, оскільки передача даних відбувається послідовно через кожен пристрій на кільці. Також слід зазначити, що ця топологія може бути дещо обмеженою у швидкості порівняно з іншими типами [12].

Отже, в контексті мультикомп'ютерних систем важливо розглядати комп'ютери, мережеве з'єднання, програмне забезпечення, протоколи комунікації,

систему керування ресурсами, застосунки та архітектуру разом з вибором оптимальної топології мережі, щоб забезпечити ефективне та надійне функціонування системи.

Важливим елементом в мультикомп'ютерних системах є також управління енергоспоживанням та оптимізація ресурсів. Механізми для автоматичного вимкнення частини системи у періоди неактивності або регулювання енергетичних режимів можуть допомогти знизити витрати та сприяти сталому розвитку.

Додатково, важливо враховувати принципи гнучкості та масштабованості мультикомп'ютерних систем. Гнучкість дозволяє системі адаптуватися до змін у вимогах та конфігураціях, а масштабованість забезпечує легкість розширення або зменшення ресурсів системи в залежності від потреб користувачів чи завдань.

З точки зору управління енергоспоживанням та оптимізації ресурсів, механізми для автоматичного вимкнення частини системи у періоди неактивності або регулювання енергетичних режимів можуть значно знизити витрати та сприяти сталому розвитку. Це особливо важливо в сучасному інформаційному суспільстві, де зменшення впливу на навколишнє середовище є однією з ключових задач.

Також важливо враховувати принципи гнучкості та масштабованості мультикомп'ютерних систем. Гнучкість дозволяє системі адаптуватися до змін у вимогах та конфігураціях, а масштабованість забезпечує легкість розширення або зменшення ресурсів системи в залежності від потреб користувачів чи завдань.

Ці аспекти разом із вибором оптимальної топології мережі та іншими ключовими складовими забезпечують ефективне та надійне функціонування мультикомп'ютерних систем у сучасному інформаційному середовищі.

1.4 Постановка задачі

Метою цієї роботи є створення саме мультикомп’ютерної системи загального призначення на основі топології «кільце» однонаправленого типу.

У процесі виконання данної роботи планується докладний аналіз та вивчення різноманітних архітектур, які можуть бути ефективно використані в системах з загальною пам'яттю.

У другому та третьому розділах роботи буде надано вичерпний аналіз проектування та втілення системи, що супроводжується докладним обґрунтуванням вибору та використання апаратно-програмної бази.

Під час процесу створення мультикомп'ютерної системи на основі топології "кільце" однонаправленого типу, велика увага повинна бути приділена вибору та оптимізації компонентів системи. Ефективна архітектура системи з загальною пам'яттю вимагає глибокого розуміння властивостей та характеристик обраного топологічного підходу.

Один із ключових аспектів при створенні такої системи - це ретельний відбір комп'ютерів, плат, шини та пам'яті. Вибір повинен базуватися на ретельному аналізі та обґрунтуванні, щоб досягти максимальної сумісності та продуктивності всіх елементів системи.

Оптимальне співвідношення ціни та продуктивності грає важливу роль у процесі вибору компонентів. Важливо забезпечити, щоб обрані компоненти відповідали технічним вимогам проекту, а також відповідали економічним обмеженням.

На рівні програмного забезпечення важливо здійснити розумний вибір для управління системою. Відповідне програмне забезпечення дозволить забезпечити ефективне управління ресурсами, взаємодію компонентів та забезпечить зручність використання системи. Крім того, правильно підібране програмне забезпечення дозволить максимально використовувати потенціал апаратної частини системи та надасть можливість налагодження мультипроцесорної системи.

Усі ці аспекти взаємодії апаратних та програмних компонентів системи мають ключове значення для досягнення максимальної продуктивності та ефективності у роботі мультикомп'ютерної системи на основі топології "кільце".

У підсумку, успішна реалізація мультикомп'ютерної системи на основі топології "кільце" вимагає комплексного підходу до вибору та оптимізації як апаратної, так і програмної складових. Глибокий аналіз різноманітних архітектур, ретельний відбір компонентів, та ефективне програмне забезпечення стають ключовими етапами цього процесу.

Дотримання технічних вимог проекту та економічних обмежень у поєднанні з розумним вибором компонентів і програмного забезпечення є важливими факторами у досягненні оптимального співвідношення ціни та продуктивності. Такий підхід дозволяє не лише забезпечити стабільну роботу системи, але і оптимізувати витрати.

Окрім того, важливо підкреслити, що ретельний вибір та правильна настройка програмного забезпечення сприяють не лише ефективному управлінню ресурсами, але й максимальному використанню потенціалу апаратної частини системи. Це дозволяє створити оптимальне середовище для розвитку та використання мультикомп'ютерної системи на основі топології "кільце" однонаправленого типу[13].

Загалом, зазначені аспекти взаємодії апаратних та програмних компонентів є ключовими для досягнення максимальної продуктивності та ефективності у роботі такої системи. Правильне планування, аналіз та вибір компонентів, а також належна настройка програмного забезпечення формують основу успішної реалізації мультикомп'ютерної системи на основі топології "кільце".

1.5 Висновки

У першому розділі проведено аналіз різноманітних типів мультикомп'ютерних систем, зосереджуючись на їх перевагах і недоліках. Ретельно вивчалися та порівнювалися різні архітектури систем, включаючи розподілені.

Переваги різних типів систем були визначені з урахуванням їх потенційних застосувань та характеристик, таких як масштабованість, надійність та швидкодія.

Наприклад, розподілені системи можуть забезпечувати велику масштабованість завдяки розподіленому зберіганню даних, тоді як клієнт-серверні системи можуть бути більш простими в управлінні та обслуговуванні.

Однак недоліки кожного типу системи також були визначені. Наприклад, розподілені системи можуть стикатися з проблемами синхронізації даних та складнощами у виявленні та усуненні несправностей через їх розподілений характер. З іншого боку, у клієнт-серверних систем може виникнути проблема однопунктової недоступності, коли відмова сервера призводить до недоступності всієї системи.

Залежно від конкретних вимог і умов застосування, вибір оптимальної архітектури мультикомп'ютерної системи може виявитися складним завданням. Наприклад, для задач, які вимагають великої масштабованості та високої доступності, розподілені системи можуть бути найкращим вибором. Вони здатні забезпечити паралельну обробку даних та високу стійкість до відмов, але водночас вимагають складних механізмів синхронізації та управління.

У той же час, для менших за обсягом завдань клієнт-серверні системи можуть бути більш простими та ефективними в управлінні рішеннями. Ці системи дозволяють централізовано керувати доступом до ресурсів та забезпечують простіше виявлення та усунення несправностей. Однак їхні можливості можуть обмежуватися однопунктовою недоступністю, що робить їх менш підходящими для завдань, які вимагають безперебійної роботи системи.

У кінцевому підсумку, вибір між різними типами мультикомп'ютерних систем повинен базуватися на специфікаціях завдань, потребах у масштабованості та надійності, а також на доступних ресурсах для управління системою та її обслуговування. Все це дозволить забезпечити максимальну користь від обраної архітектури і знизити ризики її використання.

В результаті цього аналізу було зроблено висновок про те, який тип мультикомп'ютерної системи може бути найбільш підходящим для конкретних вимог та умов застосування. Такий підхід дозволив виявити найкращі практики та

врахувати особливості кожного типу системи для максимізації їхнього корисного застосування. Крім основних типів мультикомп'ютерних систем, таких як розподілені та клієнт-серверні, важливо також розглянути інші архітектури, що можуть бути ефективними у певних сценаріях. Наприклад, системи на основі обчислювальних кластерів забезпечують високу продуктивність завдяки поєднанню обчислювальних ресурсів кількох вузлів, що працюють разом як єдина система. Такі системи широко використовуються у наукових дослідженнях та інженерних задачах, де потрібні великі обчислювальні потужності для обробки великих обсягів даних. Основною перевагою обчислювальних кластерів є можливість досягнення високої продуктивності при відносно низьких витратах порівняно з суперкомп'ютерами, однак вони можуть вимагати складного налаштування і управління.

2 ПРОЄКТУВАННЯ ТА ВИБІР АРХІТЕКТУРИ ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ

2.1 Вибір топології “кільце” однонаправленого зв’язку

Системи з розподіленою пам'яттю, такі як мультикомп'ютери, включають кілька комп'ютерів, кожен з власною оперативною пам'яттю. Процесори на кожному комп'ютері не мають прямого доступу до пам'яті інших комп'ютерів. Щоб працювати з даними на такій системі, потрібно мати копії даних на кожному комп'ютері. Кожен вузол має свої процесори і власну пам'ять, і кожен процесор має доступ лише до локальної пам'яті свого вузла.

Топологія мережі - це метод, яким взаємодіють пристрої та мережні канали в комп'ютерній системі. Це може включати фізичну топологію, що описує реальне розміщення пристроїв та кабелів у мережі, або логічну топологію, яка представляє шляхи передачі даних у мережі. Фізична топологія визначає, як пристрої з'єднані між собою та зв'язані кабелями, тоді як логічна топологія описує шляхи, по яких дані подаються у мережі [14].

Топологія “кільце” однонаправленого типу (рисунок 2.2), це є підвид топології типу кільце (рисунок 2.1).

Топологія кільця застосовується у різних областях, включаючи комп'ютерні мережі, де вона дозволяє ефективно організовувати передачу даних між пристроями. Вона використовує технологію "токену", що сприяє раціональному використанню мережевих ресурсів і забезпечує стабільну роботу мережі навіть у разі відмов окремих вузлів.

Кільцева топологія у комп'ютерних мережах означає, що всі пристрої підключені до одного логічного кільця. Кожен пристрій з'єднаний з двома сусідніми, утворюючи замкнений кільцевий шлях для передачі даних. Цю топологію можна реалізувати як за допомогою фізичного з'єднання, наприклад, коаксіального кабелю або оптичного волокна, так і за допомогою програмного забезпечення, що створює логічну мережу [15].

У кільцевій топології передачі даних використовується технологія "токену", яка циркулює вздовж кільця від одного пристрою до наступного. Коли пристрій бажає передати дані, він чекає, доки токен досягне його, після чого додає свої дані до токену і передає його далі по кільцю. Коли дані досягають призначеного пристрою, вони вилучаються з токену, і токен продовжує свій шлях по кільцю. Перший вузол з'єднаний з останнім вузлом, щоб утворити цикл. Така організація забезпечує ефективну передачу даних у кільцевій мережі, дозволяючи пристроям передавати і отримувати інформацію у визначеному порядку [16].

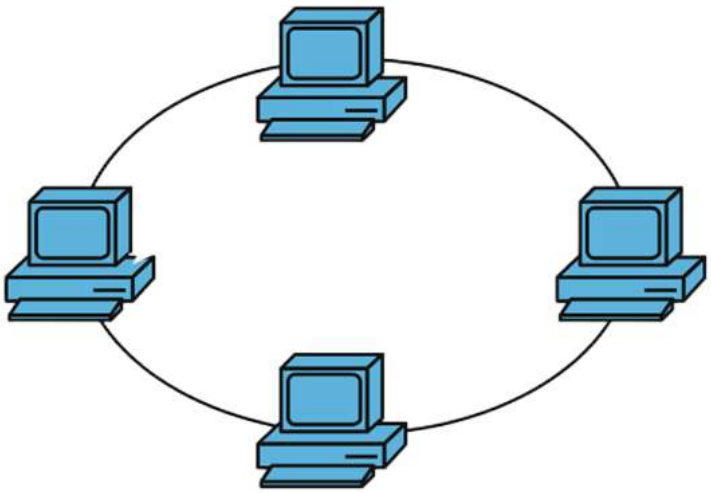


Рисунок 2.1 – Топологія кільце

Кільцева топологія є однією з основних структур для побудови комп'ютерних мереж. Вона часто використовується в ситуаціях, де потрібно забезпечити однаковий доступ до мережевих ресурсів для всіх пристроїв. Однією з переваг такої топології є простота та надійність передачі даних, а також можливість легкої розширення мережі шляхом додавання нових пристроїв до кільця. Однак важливо пам'ятати про обмеження, зокрема, вразливість мережі до відмови в разі відмови одного з пристроїв або порушення кільцевої структури. Також необхідно враховувати, що кільцева топологія може бути менш ефективною в порівнянні з іншими типами топологій у великих мережах з великою кількістю пристроїв.

Для управління мережею та іншими пристроями в межах даної топології обирається один вузол. Кільцеві топології можуть мати напівдуплексну передачу даних, але також можуть бути зроблені дуплексними. Щоб зробити кільцеву топологію повнодуплексною, потрібно мати два з'єднання між мережевими вузлами, створивши тим самим подвійну кільцеву топологію.

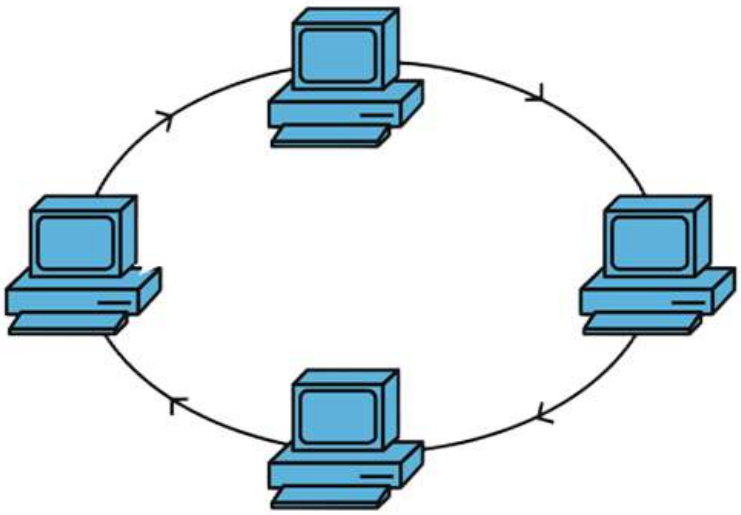


Рисунок 2.2 – Топологія кільце однонаправленого типу

Проведемо характеристики топології “кільце” однонаправленого зв’язку:

– Діаметр - залежить від кількості вузлів у мережі та довжини лінії зв'язку між вузлами. Оскільки в однонаправленому кільці кожен вузол має одне з'єднання, то діаметр становить довжину між вузлами, які мають максимальну відстань між собою в кільці:

$$D = 2 * n * d, \quad (2.1),$$

де D – діаметр;

n – кількість пристроїв в мережі;

d – відстань між двома сусідніми пристроями.

– Зв’язність:

$$C = (n - 1)/2, \quad (2.2),$$

де C – зв'язність;

n – кількість пристроїв в мережі.

– топологія "кільце" однонаправленого зв'язку має високий рівень зв'язності, що означає кожен пристрій в мережі може досягти будь-якого іншого пристрою. Дані передаються в одному напрямку по кільцю, від одного пристрою до наступного, поки не дійдуть до призначення [17];

– формула для розрахунку часу надсилання даних в топології "кільце" однонаправленого зв'язку:

$$T_s = (n * L)/S, \quad (2.3),$$

де T_s – час надсилання даних;

n – кількість пристроїв в мережі;

L – довжина кабелю;

S – швидкість передачі даних;

$n * L$ – цей множник дає нам загальну відстань, яку пакет даних повинен пройти, щоб обійти все кільце;

S – цей дільник дає нам швидкість, з якою пакет даних може передаватися по кабелю;

T_s – час надсилання даних - це час, необхідний для того, щоб пакет даних пройшов загальну відстань (n * L) з заданою швидкістю (S).

– ефективність:

$$E = C/S, \quad (2.4),$$

де E – ефективність;

C – пропускна здатність;

S – швидкість передачі даних.

У топології "кільце" однонаправленого зв’язку, пропускна здатність (C) визначається швидкістю передачі даних (S) кожним вузлом у кільці. Кожен вузол може передавати дані тільки в одному напрямку через кільце, і ця швидкість передачі даних є обмеженням для всього кільця.

Ефективність (E) у такій топології визначається співвідношенням пропускної здатності (C) до швидкості передачі даних (S). Оскільки дані у кільці рухаються лише в одному напрямку, є можливість виникнення заторів або затримок у випадку, якщо швидкість передачі даних перевищує пропускну здатність кільця. У таких ситуаціях ефективність кільця буде меншою за 1, оскільки частка використання доступної пропускної здатності буде меншою за 100%.

– час затримки:

$$T_d = n * L/S, \quad (2.5),$$

де T_d – час затримки;

n – кількість вузлів у кільці;

L – довжина кільця;

S – швидкість передачі даних.

Обчислення загального часу затримки у топології "кільце" однонаправленого зв’язку. Затримка виникає через час, який потрібен даним для проходження від одного вузла до іншого через всі інтерфейси та кабелі у кільці.

2.2 Види архітектур мультикомп’ютерної системи

2.2.1 Кластерна архітектура

Кластерна архітектура (рисунок 2.3) в сучасних обчислювальних системах відіграє важливу роль у забезпеченні високої продуктивності, масштабованості та надійності обчислювальних середовищ.

Кластерна архітектура - це метод організації обчислювальних систем, який використовує групу взаємопов'язаних комп'ютерів (вузлів), що працюють разом як єдиний об'єкт для вирішення обчислювальних задач. Основними компонентами кластера є вузли, мережа зв'язку та програмне забезпечення для управління та координації роботи вузлів.

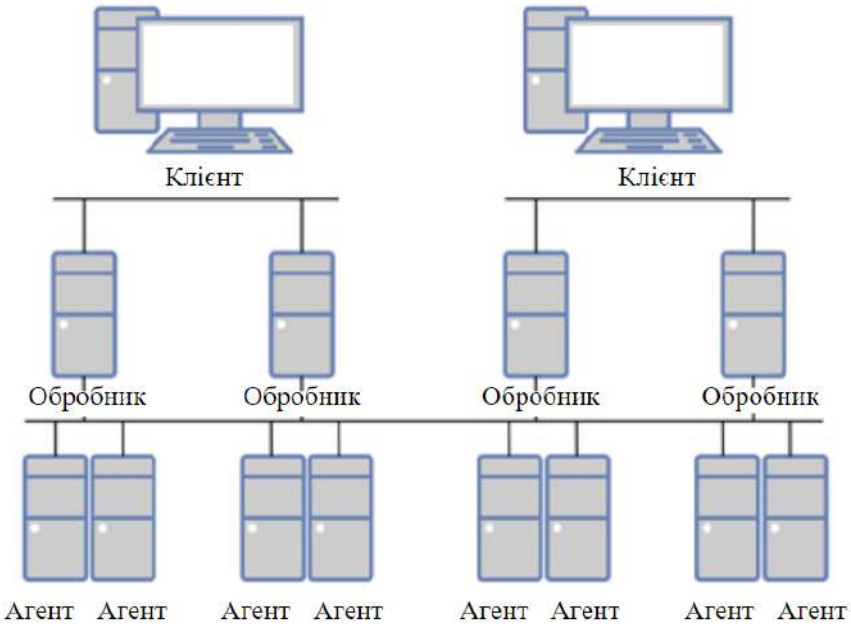


Рисунок 2.3 – Кластерна архітектура

Існують різні технології кластерної архітектури, включаючи відкриті та пропрієтарні рішення. До відомих систем кластеризації належать Apache Hadoop, Kubernetes, OpenMPI тощо. Одним з основних завдань є розробка методів балансування навантаження між вузлами кластера для оптимізації використання ресурсів [18].

Топологія кластера визначає спосіб підключення вузлів та організацію мережевого зв'язку. Розподілена топологія передбачає розподілення обчислювальних ресурсів між вузлами, в той час як централізована топологія передбачає наявність центрального вузла, який керує роботою всього кластера.

Реалізація кластерних систем вимагає як апаратного, так і програмного забезпечення. Для забезпечення високої продуктивності та надійності необхідно використовувати високоякісне обладнання та ефективне програмне забезпечення для управління кластером [19].

Кластерні системи дозволяють досягти високої продуктивності та масштабованості, а також забезпечують високу надійність завдяки резервному копіюванню та розподіленому зберіганню даних. Проте, недоліками можуть бути складність управління та високі витрати на обслуговування.

Кластерні системи застосовуються у різних галузях, включаючи наукові дослідження, фінансові установи, хмарні сервіси та інше. Вони використовуються для обробки великих обсягів даних, моделювання складних систем, а також для забезпечення високої доступності веб-сервісів та додатків.

Кластерна архітектура є важливою складовою сучасних обчислювальних систем, особливо в умовах, коли вимагається висока продуктивність та масштабованість. Вона дозволяє об'єднати значну кількість обчислювальних ресурсів у єдине ціле, що сприяє оптимізації використання обладнання та програмного забезпечення. Одним з ключових аспектів кластерної архітектури є можливість балансування навантаження між вузлами для забезпечення максимальної продуктивності [20].

Успішна реалізація кластерних систем вимагає не лише потужного апаратного забезпечення, але й ефективного програмного забезпечення для управління кластером. Програмне забезпечення для керування вузлами кластера забезпечує координацію та управління роботою вузлів, оптимізуючи ресурси і виконання завдань [21].

Кластерні системи часто використовуються для обробки великих обсягів даних, наприклад, у сферах наукових досліджень і фінансових установ, де необхідно проводити складні обчислення та аналізи. Вони також широко використовуються у хмарних сервісах для забезпечення високої доступності та продуктивності сервісів та додатків.

Недоліками кластерних систем можуть бути складність у налаштуванні та управлінні, а також високі витрати на обслуговування та підтримку. Однак при правильному проектуванні і налагодженні вони здатні значно підвищити ефективність обчислень та забезпечити стабільну роботу великих систем [22].

2.2.2 Сіткова архітектура

Сіткова архітектура (рисунок 2.4) - це концепція будівництва комп'ютерних систем, що базується на ідеї розподіленого оброблення даних та завдань. Цей підхід знаходить широке застосування в різних сферах, включаючи хмарні обчислення, Інтернет речей (IoT), оброблення великих обсягів даних та інші.

Основні принципи сіткової архітектури:

– розподілені ресурси. Основна ідея сіткової архітектури полягає в тому, щоб розподілити завдання та ресурси між різними вузлами або вузловими групами у мережі. Це дозволяє оптимізувати використання обчислювальних ресурсів та забезпечує більшу масштабованість системи;

– масштабованість. Сіткова архітектура забезпечує можливість легкої масштабованості. Нові вузли можуть бути додані до мережі для забезпечення

оброблення додаткових завдань або збільшення продуктивності без великих змін у загальній структурі системи;

– відмовостійкість. Оскільки сіткова архітектура базується на розподілених ресурсах, вона зазвичай володіє високою відмовостійкістю. Відмова одного вузла не призводить до відмови всієї системи, оскільки завдання можуть бути перерозподілені на інші вузли;

– паралельне оброблення. Сіткова архітектура сприяє паралельному обробленню завдань, оскільки різні вузли можуть одночасно виконувати різні частини роботи. Це дозволяє підвищити продуктивність та зменшити час оброблення даних.

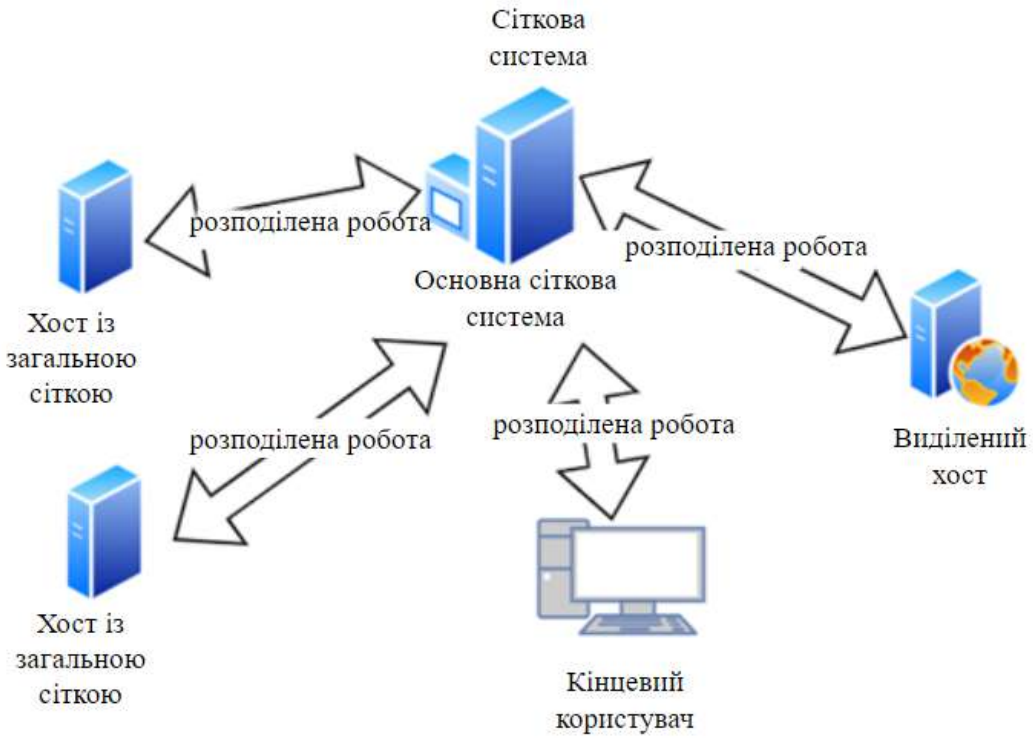


Рисунок 2.4 – Сіткова архітекрура

Переваги сіткової архітектури:

– висока масштабованість. Завдяки розподіленню завдань та ресурсів, сіткова архітектура дозволяє системам ефективно масштабуватись, щоб відповідати зростаючим вимогам обробки даних та завдань;

– відмовостійкість. Системи, побудовані на основі сіткової архітектури, володіють високою відмовостійкістю, оскільки вони можуть продовжувати працювати навіть після відмови окремих компонентів;

– ефективне використання ресурсів. Сіткова архітектура дозволяє ефективно використовувати обчислювальні ресурси, розподіляючи завдання на вузли в мережі та використовуючи їх повністю.

Застосування сіткової архітектури:

– хмарні обчислення. Багато хмарних обчислювальних платформ базуються на сітковій архітектурі, дозволяючи користувачам легко масштабувати свої ресурси та обробляти великі обсяги даних;

– інтернет речей (IoT). Системи IoT часто використовують сіткову архітектуру для забезпечення сприятливої обробки даних з великої кількості пристроїв, розташованих у різних місцях;

– оброблення великих обсягів даних. У сферах, таких як аналітика даних, машинне навчання та штучний інтелект, сіткова архітектура дозволяє розподілити завдання обробки даних між багатьма вузлами чи серверами. Це дозволяє швидше обробляти великі обсяги даних та виконувати складні аналізи без перевантаження одного сервера.

Однією з ключових переваг сіткової архітектури є її здатність до гнучкого розширення. Нові вузли можуть бути додані або вилучені з мережі в залежності від потреб системи, що дозволяє ефективно адаптувати її до змінюючихся обставин. Це особливо важливо у сучасному світі, де вимоги до обробки даних можуть раптово змінюватися.

Крім того, сіткова архітектура підтримує використання відкритих стандартів та протоколів, що сприяє інтеграції різних систем та пристроїв. Це дозволяє

розробникам створювати складні розподілені системи, які можуть легко взаємодіяти між собою, що є важливим для розвитку сучасних технологій.

Отже, сіткова архітектура не лише сприяє оптимізації використання обчислювальних ресурсів та забезпечує високу відмовостійкість, але й є ключовою складовою для побудови сучасних розподілених систем у різних галузях, що забезпечує їм необхідну гнучкість та масштабованість для вирішення складних завдань обробки даних.

Ця гнучкість і масштабованість, що забезпечується сітковою архітектурою, особливо важлива в умовах швидкого темпу зростання обсягів даних та потреби у їх обробці. Розподілені системи, побудовані на основі цієї архітектури, можуть легко адаптуватися до змінних умов і вимог, що дозволяє їм ефективно функціонувати в різних галузях, від фінансів та телекомунікацій до медицини та науки. Крім того, завдяки використанню відкритих стандартів, ці системи можуть швидко інтегруватися з існуючими технологіями та рішеннями, що сприяє зниженню витрат на розробку та впровадження нових продуктів і сервісів. Такий підхід до побудови розподілених систем відкриває шлях до нових можливостей у сфері інновацій та розвитку технологій, дозволяючи компаніям швидко реагувати на зміни в ринкових умовах та вимоги споживачів.

2.2.3 Масивно-паралельна архітектура

Масивно-паралельна архітектура (рисунок 2.5) є ключовим елементом в розвитку сучасних систем обчислень. Вона дозволяє ефективно виконувати великі обчислювальні завдання шляхом розділення їх на багато менших фрагментів, які обробляються паралельно.

Масивно-паралельна архітектура - це підхід до проектування систем обчислень, де багато обчислювальних одиниць працюють паралельно для вирішення обчислювальних задач. Ця архітектура використовується для

розподілення завдань між багатьма процесорами або ядрами, щоб забезпечити високу продуктивність та швидкодію [23].

Масивно-паралельна архітектура знаходить широке застосування в різних галузях, де потрібні високі обчислювальні потужності. Наприклад, у наукових дослідженнях, таких як моделювання кліматичних змін, біоінформатика та аналіз великих даних, масивно-паралельні системи дозволяють виконувати складні розрахунки набагато швидше, ніж традиційні послідовні системи. Крім того, в індустрії розваг, такі архітектури використовуються для рендерингу високоякісної графіки в реальному часі, що є критичним для створення сучасних відеоігор та анімаційних фільмів. Завдяки можливості обробляти великі обсяги даних одночасно, масивно-паралельні системи значно підвищують ефективність та продуктивність цих процесів.

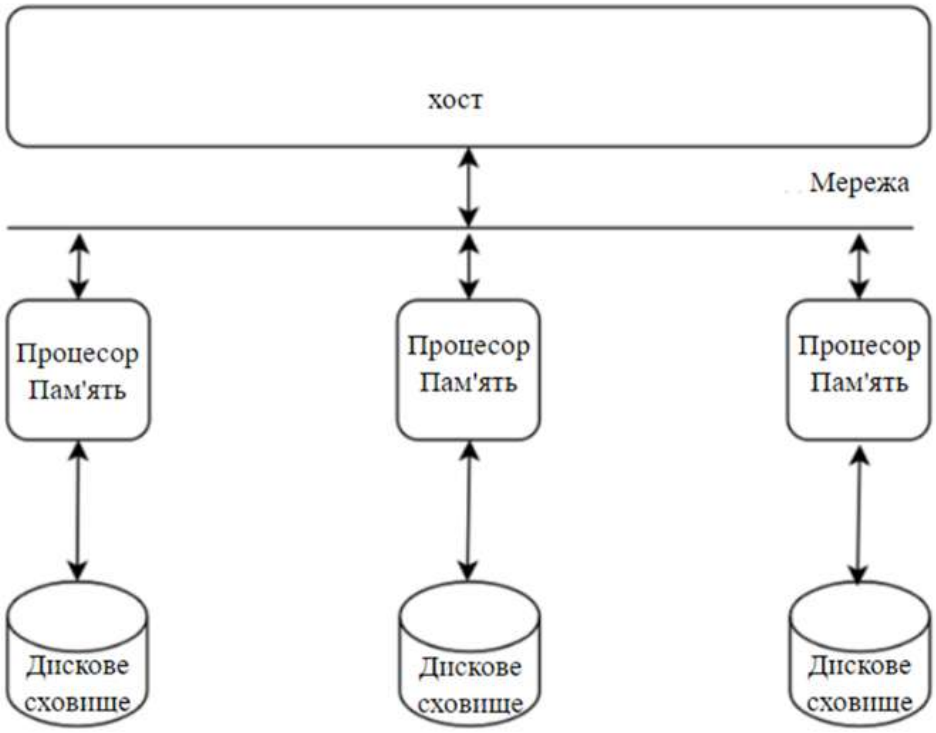


Рисунок. 2.5 – Масивно-паралельна архітектура

Основні принципи масивно-паралельної архітектури:

– розділення завдань. Обчислювальні завдання розбиваються на багато менших фрагментів, які можуть бути виконані паралельно;

– комунікація. Паралельні обчислювальні одиниці повинні здійснювати обмін даними та результатами своєї роботи для координації та завершення завдань;

– синхронізація. Деякі обчислювальні задачі можуть вимагати синхронізації між паралельними обчислювальними одиницями для забезпечення правильності результатів;

– масштабованість. Масивно-паралельні архітектури повинні бути здатні масштабуватися від невеликих конфігурацій до дуже великих систем з тисячами обчислювальних одиниць.

Переваги масивно-паралельної архітектури:

– висока продуктивність. Завдяки паралельному виконанню обчислень можна досягти значно вищої продуктивності порівняно з традиційними однопроцесорними системами;

– швидкодія. Розподілення завдань між багатьма обчислювальними одиницями дозволяє вирішувати складні обчислювальні задачі швидше;

– масштабованість. Масивно-паралельні системи можуть бути легко масштабовані для вирішення зростаючих обчислювальних вимог;

– резервне копіювання. При використанні масивно-паралельної архітектури можна використовувати резервні копії обчислювальних одиниць для забезпечення надійності системи.

Використання масивно-паралельної архітектури:

– наукові обчислення. Великі обчислювальні завдання в області науки, такі як моделювання клімату, обробка сигналів та геноміки, можуть бути вирішені за допомогою масивно-паралельних систем;

– ігрова індустрія. У виробництві комп'ютерних ігор використовуються масивно-паралельні системи для обробки складних графічних ефектів та штучного інтелекту;

– фінансові обчислення. У фінансовому секторі масивно-паралельні архітектури використовуються для швидкого аналізу великих обсягів фінансових даних та моделювання ринків;

– хмарні обчислення також використовують масивно-паралельні системи для забезпечення високої доступності та швидкості обробки даних. Це дозволяє користувачам отримувати доступ до різноманітних сервісів та додатків з будь-якого пристрою та в будь-який час.

Застосування масивно-паралельної архітектури також розширюється на сектори штучного інтелекту та машинного навчання, де великі обчислювальні завдання можуть бути розподілені між багатьма обчислювальними ресурсами для тренування складних моделей та аналізу великих наборів даних. Це дозволяє підвищити точність та швидкість аналізу даних, що є критичним для розробки нових технологій та додатків.

Однією з важливих переваг масивно-паралельної архітектури є її ефективність у використанні ресурсів. Замість використання одного потужного процесора, системи можуть об'єднувати потужності сотень або навіть тисяч обчислювальних одиниць для виконання завдань. Це дозволяє економно використовувати електроенергію та оптимізувати витрати на обслуговування обладнання.

У підсумку, масивно-паралельна архітектура стала основним інструментом у розвитку сучасних систем обчислень, що дозволяє вирішувати складні завдання швидко, ефективно та надійно. Її застосування в різних сферах забезпечує високу продуктивність та швидкість обробки даних, що робить її невід'ємною частиною сучасних технологій.

Крім того, ця архітектура дозволяє обробляти великі обсяги даних у режимі реального часу, що є критично важливим для таких галузей, як фінанси, медицина та наукові дослідження. Завдяки своїй масштабованості, масивно-паралельна архітектура легко адаптується до зростаючих вимог і може бути розширена для підтримки нових технологій та додатків.

2.2.4 Масштабована архітектура

Масштабована архітектура (рисунок 2.6) стала новинкою у світі сучасних технологій, яка дозволяє системам ефективно збільшувати свою потужність та продуктивність з ростом обсягів даних та навантаження.

Масштабована архітектура - це дизайн системи, яка може змінюватися за рахунок збільшення чи зменшення її ресурсів без значних змін у самому програмному коді чи архітектурі. Ця архітектура може бути застосована до різних типів систем, включаючи хмарні платформи, бази даних, мережеві інфраструктури та додатки [24].

Масштабована архітектура дозволяє легко додавати або зменшувати ресурси, такі як обчислювальні потужності, пам'ять та сховища даних, забезпечуючи таким чином стабільну роботу системи та високу продуктивність навіть при значних змінах у робочому навантаженні [25].

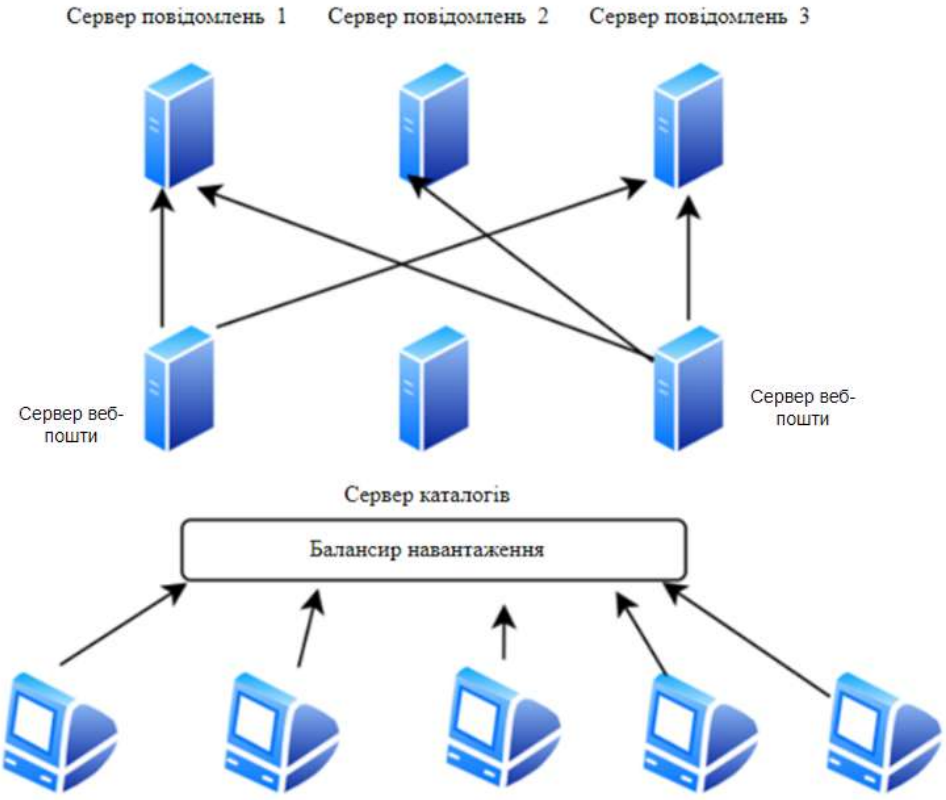


Рисунок.2.6 – Масштабована архітектура

Основні принципи масштабованої архітектури:

– горизонтальна та вертикальна масштабованість. Горизонтальна масштабованість (scale-out) полягає в додаванні нових екземплярів системи для збільшення потужності. Вертикальна масштабованість (scale-up), натомість, використовує збільшення ресурсів (наприклад, CPU, пам'ять) на існуючих серверах;

– відновлення під час збоїв. Масштабовані системи повинні мати механізми для автоматичного відновлення у випадку виникнення збоїв. Це може включати резервне копіювання даних та використання дублювання серверів [26];

– гнучкість та еластичність. Система повинна бути гнучкою та еластичною, щоб забезпечити зміну обсягів навантаження без значних перерв у роботі;

– моніторинг та аналіз. Ефективне використання масштабованих систем вимагає постійного моніторингу та аналізу їхньої продуктивності та ресурсів.

Переваги масштабованої архітектури:

– гнучкість. Масштабовані системи можуть швидко адаптуватися до зміни обсягів даних та навантаження без значних зусиль;

– швидкодія. Із збільшенням кількості ресурсів система може забезпечувати швидше виконання завдань та обробку даних;

– економічність. Масштабовані системи дозволяють оптимізувати використання ресурсів та знижувати витрати на обслуговування та розширення;

– надійність. Завдяки можливості автоматичного відновлення в разі збоїв, масштабовані системи є більш надійними та стійкими до випадкових перебоїв.

Використання масштабованої архітектури:

– хмарні платформи. Багато хмарних платформ надають можливості автоматичного масштабування ресурсів в залежності від потреб користувачів;

– веб-програмування. Сайти та веб-додатки можуть використовувати масштабовану архітектуру для забезпечення швидкого та надійного доступу до контенту;

– бази даних. Масштабовані бази даних дозволяють збільшувати обсяги даних та виконувати запити швидше за рахунок розподілення даних між багатьма серверами;

– аналіз даних. У сфері аналізу даних масштабовані архітектури використовуються для обробки великих обсягів даних та виконання складних аналітичних операцій. Це дозволяє підприємствам та дослідникам швидше виявляти тенденції, робити прогнози та виявляти важливі інсайти, що полегшує прийняття стратегічних рішень [27].

Забезпечення безперервності роботи. Масштабовані системи часто використовують різноманітні стратегії для забезпечення неперервності роботи, включаючи розміщення даних та додатків у різних географічних регіонах, використання резервних серверів та автоматичне перемикання на резервні ресурси у разі необхідності.

Безпека. Масштабовані системи також вимагають потужних заходів безпеки для захисту від вторгнень, втрати даних та інших загроз. Це може включати шифрування даних, аутентифікацію користувачів, контроль доступу та моніторинг потенційних загроз [28].

В цілому, масштабована архітектура стала важливою складовою сучасних технологій, дозволяючи створювати системи, які можуть ефективно масштабуватися та працювати в умовах зростаючих обсягів даних та навантаження. Це дозволяє організаціям забезпечувати високу доступність, продуктивність та безпеку своїх систем у сучасному цифровому середовищі.

2.2.5 Системи з розподіленою пам'яттю (NUMA)

Системи з розподіленою пам'яттю (рисунок 2.7) (NUMA) стали основою для сучасних високопродуктивних обчислювальних систем, зокрема в серверних та облачних середовищах.

Системи з розподіленою пам'яттю (NUMA) - це архітектурний підхід, у якому пам'ять комп'ютера фізично розташована на різних вузлах чи вузлах обробки, і доступ до неї може відрізнятися в залежності від того, на якому процесорі або ядрі виконується обчислення. Кожен процесор має свою власну локальну пам'ять, а також може мати доступ до пам'яті інших процесорів через шину.

Системи з розподіленою пам'яттю (NUMA) відіграють ключову роль у сучасних високопродуктивних обчислювальних системах, забезпечуючи ефективне управління ресурсами та забезпечуючи підтримку великих обсягів даних. Ця архітектура дозволяє збільшити масштабованість систем, дозволяючи додавати нові вузли обробки та пам'яті без значного впливу на продуктивність. Крім того, NUMA дозволяє зменшити час доступу до пам'яті шляхом забезпечення доступу до локальної пам'яті протягом коротших проміжків часу. Це робить NUMA особливо ефективною для великих масштабованих застосунків, таких як бази даних, веб-сервери та високопродуктивні обчислення в облачних середовищах [29].

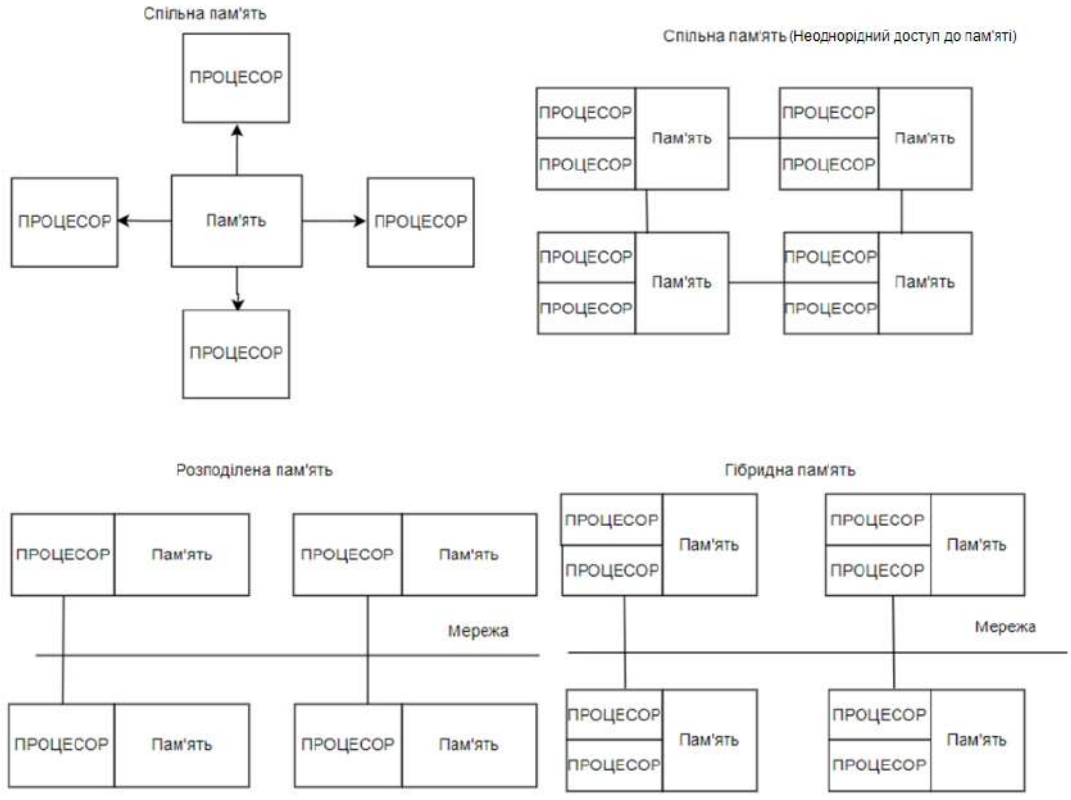


Рисунок 2.7 – Системи з розподіленою пам'яттю (NUMA)

Основні принципи NUMA:

– локальність доступу до пам'яті. Процесори мають прямий доступ до своєї локальної пам'яті, що забезпечує швидкий доступ до даних, які зберігаються на цьому процесорі;

– віддалений доступ до пам'яті. Процесори також можуть мати доступ до пам'яті, яка розташована на інших вузлах, але цей доступ може бути повільнішим через використання шини або мережі;

– оптимізація міграції задач. Операційні системи та обчислювальні середовища можуть використовувати механізми міграції задач для забезпечення того, щоб обчислювальні завдання виконувалися на процесорах, які мають доступ до необхідних даних;

– балансування навантаження. Із збільшенням кількості процесорів може виникати потреба в балансуванні навантаження між вузлами системи для оптимізації продуктивності.

Переваги систем з розподіленою пам'яттю (NUMA):

– висока продуктивність. NUMA дозволяє ефективно використовувати ресурси системи, забезпечуючи швидкий доступ до локальної пам'яті та оптимізований доступ до віддаленої пам'яті;

– масштабованість. NUMA може легко масштабуватися для використання в системах з великими обсягами даних та високими обчислювальними вимогами;

– ефективне використання ресурсів. Локальний доступ до пам'яті дозволяє уникнути зайвого навантаження на системні шини та мережу, що покращує загальну продуктивність системи;

– гнучкість конфігурації. NUMA системи дозволяють конфігурувати топологію вузлів та взаємозв'язки між ними для оптимізації використання ресурсів.

Використання систем з розподіленою пам'яттю (NUMA):

– серверні системи. NUMA широко використовується в серверних системах для обробки великих обсягів даних та виконання вимогливих обчислювальних завдань;

– бази даних. Системи NUMA дозволяють оптимізувати доступ до даних у великих базах даних та забезпечувати високу швидкодію обробки запитів.

2.2.6 Вибір архітектури мультикомп’ютерної системи

Серед різноманітних архітектур комп'ютерів виокремлюється MIMD (Multiple Instruction, Multiple Data), яка знаходить широке застосування в сучасних комп'ютерах.

MIMD-архітектура полягає у виконанні однієї або декількох інструкцій на різних наборах даних у різних обчислювальних блоках. Це дозволяє розділяти завдання на більші елементарні операції, що виконуються паралельно. Основні особливості MIMD-архітектури включають:

– паралельність інструкцій. Кожен обчислювальний блок виконує свої власні інструкції, що дозволяє одночасно обробляти різні набори даних;

– незалежність блоків. Різні обчислювальні блоки можуть працювати незалежно один від одного, що робить систему більш гнучкою та масштабованою.

MPP системи (рисунок 2.8) є найактуальнішим підкласом MIMD-архітектури, де велика кількість вузлів з'єднана мережею для спільної обробки даних. Основні характеристики MPP систем включають:

– масштабованість. Можливість додавати нові вузли для збільшення обчислювальних ресурсів системи;

– висока продуктивність. Завдяки паралельному виконанню завдань MPP системи здатні швидко обробляти великі обсяги даних.

Застосування в наукових дослідженнях та бізнесі: MPP системи використовуються для обробки великих обсягів даних у таких галузях, як наукові дослідження, фінанси, медицина тощо.

MPP системи також забезпечують високий рівень надійності та стійкості до відмов, оскільки кожен вузол працює незалежно, і вихід з ладу одного вузла не призводить до зупинки всієї системи. Це робить їх ідеальними для критично

важливих застосувань, де безперервність роботи є ключовою вимогою. Завдяки своїй здатності масштабуватися та забезпечувати високу продуктивність, MPP системи стали незамінним інструментом у багатьох галузях. Крім того, у сфері енергетики MPP системи допомагають в оптимізації виробництва та розподілу енергії, аналізуючи великі масиви даних для прогнозування попиту та управління ресурсами. Додатково, MPP системи забезпечують велику швидкість обробки даних завдяки паралельному виконанню операцій на кількох вузлах одночасно. Це дозволяє ефективно впоратися з великим обсягом інформації та складними обчисленнями у реальному часі.

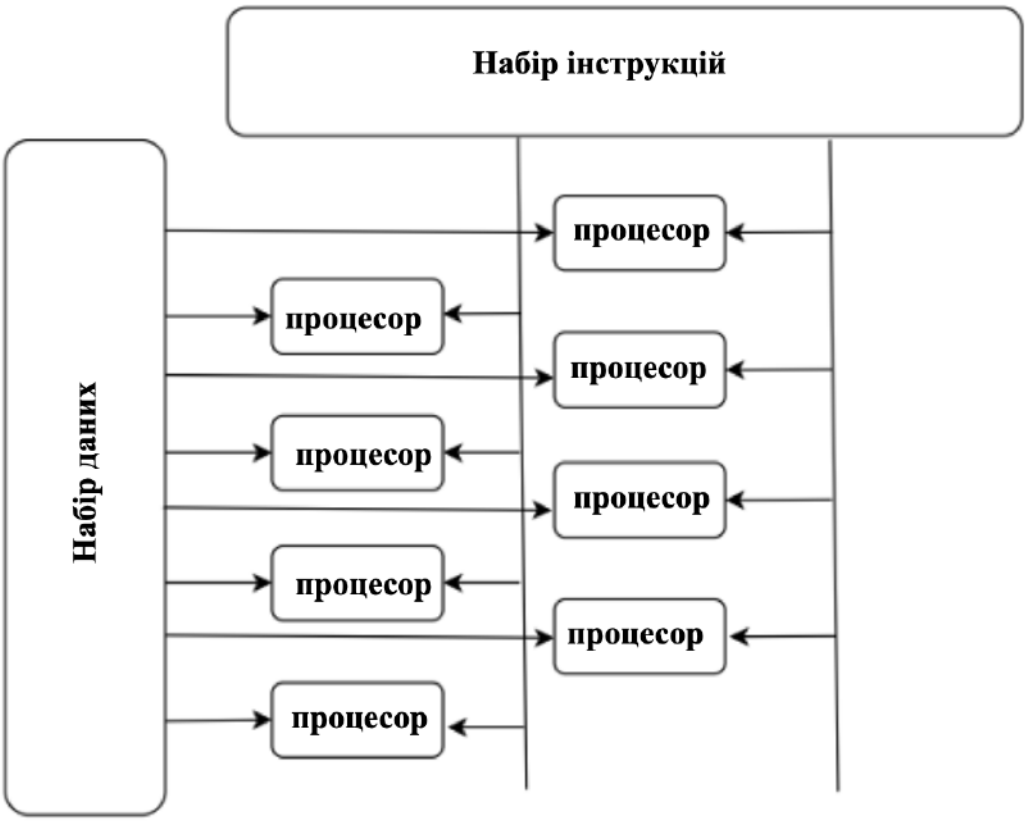


Рисунок 2.8 – MIMD-архітектури та MPP систем

Переваги MIMD-архітектури та MPP систем:

– ефективність паралелізму. MIMD-архітектура та MPP системи дозволяють виконувати багатозадачні обчислення паралельно, що значно прискорює час обробки даних;

– масштабованість. MPP системи можуть легко масштабуватися для відповіді на зростаючі обсяги даних або потреби в обчислювальних ресурсах;

– висока надійність. Завдяки розділенню завдань між різними вузлами, MPP системи є менш схильними до відмов та забезпечують надійність у виконанні завдань.

Одним з найвразливіших місць в MPP-системі завжди був і залишається центральний управляючий пристрій (ЦУП). Якщо він вийде з ладу, це може призвести до непрацездатності всієї системи. Для підвищення надійності ЦУП використовуються стратегії спрощення або навіть дублювання його апаратних компонентів. Сфера застосування операційних систем з масовим паралелізмом постійно розширюється, і різні системи цього класу успішно експлуатуються у провідних суперкомп'ютерних центрах по всьому світу.

D-архітектура, зокрема MPP системи, є одними з найактуальніших та ефективних рішень для обробки великих обсягів даних у сучасному світі. Їхні переваги у високій продуктивності, масштабованості та надійності роблять їх привабливими для застосування в різних областях, від наукових досліджень до бізнесу.

2.3 Вибір та опис програмного забезпечення

У мультикомп'ютерних системах з топологією "кільце" однонаправленого зв'язку ефективне проміжне програмне забезпечення є ключовим компонентом для забезпечення надійного та ефективного обміну даними між вузлами мережі.

Основні функції проміжного програмного забезпечення включають:

1) управління мережевим трафіком. Програмне забезпечення повинно здійснювати контроль та керування потоком даних по кільцевій топології, забезпечуючи правильну адресацію та маршрутизацію пакетів даних між вузлами;

2) обробка повідомлень. Проміжне програмне забезпечення повинно обробляти повідомлення, які надходять від вузлів мережі, враховуючи їхні призначення та забезпечуючи передачу до наступного вузла у кільці;

3) механізми виявлення та виправлення помилок. Враховуючи особливості однонаправленого зв'язку у топології "кільце", проміжне програмне забезпечення повинно вміти виявляти та виправляти помилки, такі як втрата пакетів даних чи затримки у передачі;

4) синхронізація та керування конфліктами. З урахуванням кільцевої топології, проміжне програмне забезпечення повинно забезпечувати синхронізацію взаємодії між вузлами та вирішення можливих конфліктів, що можуть виникнути у процесі передачі даних;

5) безпека та захист даних. Програмне забезпечення повинно забезпечувати механізми шифрування та аутентифікації для забезпечення конфіденційності та цілісності даних, що передаються по мережі;

6) моніторинг та аналіз мережі. Проміжне програмне забезпечення може включати засоби для моніторингу та аналізу стану мережі, що дозволяє виявляти проблеми та вдосконалювати її ефективність.

Проміжне програмне забезпечення в мультикомп'ютерних системах з топологією "кільце" однонаправленого зв'язку відіграє критичну роль у забезпеченні надійності та продуктивності мережі, тому вибір відповідного рішення вимагає уважного аналізу та врахування специфічних вимог системи.

Обрання мови програмування C# та використання сокетів для реалізації мультикомп'ютерної системи з топологією "кільце" однонаправленого зв'язку має кілька обґрунтованих причин:

Підтримка мережевого програмування в C#: Мова програмування C# має вбудовану підтримку мережевого програмування через класи System.Net.Sockets, що дозволяє легко створювати сокети для взаємодії між вузлами мережі.

Простота та зручність синтаксису C#: Синтаксис мови C# є інтуїтивно зрозумілим та легким для вивчення, що сприяє швидкій розробці програмного

забезпечення. Це особливо важливо у випадку розробки складних систем, таких як мережеві додатки.

Підтримка асинхронного програмування: C# має вбудовану підтримку асинхронного програмування за допомогою ключових слів async і await, що дозволяє створювати ефективні та реактивні додатки, які можуть працювати з паралельним виконанням операцій в мережі.

Широка спільнота розробників та ресурси для підтримки: C# має велику та активну спільноту розробників, що забезпечує доступ до безлічі ресурсів, бібліотек та форумів для підтримки. Це може виявитися важливим у випадку виникнення проблем або потреби у консультаціях під час розробки мережевого програмного забезпечення[30].

Обрання сокетів (в даному випадку TCP та UDP) для реалізації мережевого взаємодії також має свої переваги:

Простота і надійність передачі даних: Сокети TCP і UDP дозволяють ефективно передавати дані між вузлами мережі з різними гарантіями доставки (наприклад, з TCP - гарантією доставки та порядком пакетів, з UDP - без гарантії доставки).

Гнучкість у виборі протоколу: Залежно від потреб конкретного додатку, можна вибрати між протоколами TCP або UDP з урахуванням вимог до надійності та швидкодії передачі даних.

Підтримка різноманітних мережевих топологій: Сокети дозволяють забезпечувати взаємодію між вузлами мережі в різних топологіях, включаючи кільцеву топологію, яка в даному випадку використовується.

Отже, використання мови програмування C# та сокетів для реалізації мультикомп'ютерної системи з топологією "кільце" однонаправленого зв'язку є логічним вибором, оскільки ці технології забезпечують швидку розробку, ефективну мережеву взаємодію та надійність передачі даних.

2.4 Вибір та опис апаратного забезпечення

Для побудови проміжного програмного забезпечення мультикомп'ютерної системи використовується топологія "кільце". Ця топологія відображається у способі підключення пристроїв, де кожен ПК з'єднаний з двома сусідніми ПК у кільцевій структурі. Використане апаратне забезпечення:

– мережеві комутатори. Для забезпечення комутації мережевого трафіку використовуються мережеві комутатори, що підтримують технологію Ethernet. Комутатори забезпечують підключення кожного ПК до кільцевої мережі;

– мережеві кабелі. Використовуються мережеві кабелі категорії 5e або вище для з'єднання кожного ПК з мережевим комутатором. Ці кабелі забезпечують надійну передачу даних з високою швидкістю;

– мережеві адаптери для ПК. Кожен ПК обладнаний мережевим адаптером, що дозволяє підключатися до мережі. Це може бути вбудований адаптер на материнській платі або зовнішній адаптер, підключений через USB або PCI-роз'єми [31];

– додаткові матеріали. Для забезпечення організованості мережевого обладнання можуть використовуватися кабельні тримачі або інші засоби кріплення, які допомагають утримувати кабелі в порядку та запобігають заплутанню.

2.5 Висновки

У рамках проєктування проміжного програмного забезпечення для мультикомп'ютерної системи з топологією "кільце" однонаправленого зв'язку було розглянуто важливі аспекти апаратного та програмного забезпечення.

Перш за все, було визначено, що кільцева топологія забезпечує однонаправлене з'єднання між кожним з комп'ютерів, що дозволяє ефективно передавати дані в одному напрямку. Для цього було обрано відповідне апаратне

забезпечення, таке як мережеві комутатори, кабелі та мережеві адаптери, щоб забезпечити надійність та продуктивність мережі.

У контексті програмного забезпечення було обрано мову програмування C#, яка має численні переваги, включаючи підтримку .NET Framework/.NET Core, можливості мультиплатформенності та велику спільноту розробників. Крім того, використання сокетів (UDP або TCP) дозволить ефективно організувати комунікацію між комп'ютерами у кільцевій мережі.

Для досягнення найкращої продуктивності та надійності системи було використано розподілене програмне забезпечення, що дозволило кожному комп'ютеру виконувати свої завдання без зайвих навантажень на центральний сервер. Кожен вузол мережі може обмінюватися даними з сусідніми вузлами безпосередньо через мережеві комутатори, що значно полегшує процес передачі інформації.

У результаті проєктування програмно-технічного засобу було забезпечено оптимальну апаратну та програмну інфраструктуру для ефективної роботи мультикомп'ютерної системи з топологією "кільце". Це дозволить досягти надійного та продуктивного зв'язку між комп'ютерами та забезпечити стабільну роботу програмного засобу.

3 ПРОЄКТУВАННЯ ПРОГРАМНО-ТЕХНІЧНОГО ЗАСОБУ

3.1 Алгоритм під’єднання складових мультикомп’ютерної системи

Алгоритм підключення складових мультикомп'ютерної системи з топологією "кільце" однонаправленого зв'язку може виглядати наступним чином:

– підготовка обладнання. Підключення мережевих адаптерів до кожного комп'ютера. Підключення кабелів для створення кільцевої топології зв'язку між комп'ютерами;

– підключення комутаторів. Комутатори в такій топології можуть не використовуватися, оскільки мережа буде зв'язана утвореним кільцем прямими з'єднаннями між комп'ютерами [32];

– підключення маршрутизаторів (якщо потрібно). Якщо в мережі потрібен маршрутизатор для доступу до інших мереж або Інтернету, він може бути підключений до будь-якого з комп'ютерів у кільці;

– налаштування програмного забезпечення. Налаштування IP-адрес для мережевих адаптерів комп'ютерів відповідно до потреб мережі;

– встановлення програмного забезпечення для керування мережевими пристроями, якщо воно необхідне.

Перевірка з'єднання. Після завершення підключення перевірте з'єднання між комп'ютерами за допомогою інструментів для мережевої діагностики.

У цьому випадку ми створили кільцеву топологію (рисунок 3.1) зв'язку між комп'ютерами, де кожен комп'ютер безпосередньо з'єднаний з двома сусідніми комп'ютерами. Це створює однонаправлений обхід даних по кільцю і забезпечує певний рівень надійності та стійкості мережі [33].

Однією з ключових переваг такої топології є забезпечення безперервності зв'язку, оскільки мережа може продовжувати функціонувати навіть у випадку виходу з ладу одного з комп'ютерів. Дані можуть обійти пошкоджену ділянку, що мінімізує вплив на загальну роботу системи. Крім того, кільцева структура сприяє

низькій затримці передачі даних, оскільки дані передаються послідовно від одного вузла до іншого, що робить затримку передбачуваною і мінімальною. Це є важливою характеристикою для додатків, чутливих до часу.

Простота розширення є ще однією перевагою кільцевої топології. Додавання нового комп'ютера в таку мережу є відносно простим процесом, оскільки новий комп'ютер підключається між двома існуючими вузлами, і налаштовується відповідно. Це дозволяє легко масштабувати мережу у відповідь на зростаючі потреби [34].

Однак, кільцева топологія має і свої недоліки. Одноточковий вузол відмови може бути серйозною проблемою, якщо кільце не має додаткових механізмів для обхідного маршруту. Вихід з ладу одного вузла може зупинити роботу всієї мережі, що робить необхідним впровадження резервних рішень для підвищення надійності. Складність діагностики помилок у такій топології також є важливим фактором. Виявлення і усунення помилок може бути складнішим порівняно з іншими топологіями, оскільки необхідно перевіряти кожен вузол і з'єднання послідовно.

Використання проміжного програмного забезпечення для управління зв'язком між вузлами є важливим аспектом. Таке програмне забезпечення дозволяє абстрагуватися від низькорівневих деталей мережевої взаємодії, забезпечуючи більш простий і гнучкий інтерфейс для розробників додатків. Проміжне програмне забезпечення може включати механізми для автоматичного відновлення з'єднань, управління потоками даних і балансування навантаження між вузлами. Підтримка стандартних мережевих протоколів, таких як TCP/IP, є критичною для забезпечення сумісності та інтероперабельності з іншими системами і додатками.

Таким чином, реалізація кільцевої топології з використанням однонаправленого зв'язку і проміжного програмного забезпечення дозволяє створити ефективну, надійну і масштабовану мультикомп'ютерну систему, здатну обробляти великі обсяги даних і забезпечувати високу продуктивність у різних сценаріях використання [35].

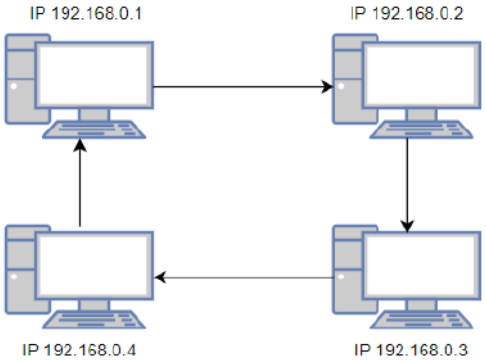


Рисунок 3.1 – Схема підключення пк

3.2 Вибір програмних засобів

Мова програмування визначає основні можливості, швидкість розробки та ефективність програмного забезпечення. Після ретельного аналізу доступних варіантів було обрано C# через наступні переваги:

– підтримка .NET Framework/.NET Core: C# інтегрується з екосистемою .NET, що забезпечує широкі можливості розробки та високу продуктивність;

– можливості мультиплатформенності: За допомогою .NET Core можна розробляти та виконувати додатки на різних платформах, таких як Windows, Linux та MacOS;

– велика спільнота розробників: C# має велику та активну спільноту, що забезпечує доступ до безлічі ресурсів, бібліотек та форумів для підтримки.

Деякі інші особливості C# включають:

– синтаксис інтуїтивно зрозумілий: C# має чистий і лаконічний синтаксис, який легко читається і розуміється, особливо для розробників, які вже мають досвід у Java, C++ або інших подібних мовах;

– об'єктно-орієнтована мова програмування: C# підтримує об'єктно-орієнтовану парадигму програмування, що дозволяє розробникам створювати модульний, легко збережений та розширюваний код;

– система типізації: C# є строго типізованою мовою програмування, що дозволяє виявляти помилки в коді під час компіляції, що сприяє зменшенню числа помилок на етапі розробкиl

– підтримка асинхронного програмування: C# має вбудовану підтримку асинхронного програмування за допомогою ключових слів async і await, що дозволяє створювати ефективні та реактивні додатки;

– LINQ (Language Integrated Query): Це потужний інструмент для роботи з колекціями даних в C#. LINQ дозволяє виконувати запити до джерел даних (наприклад, масивів, колекцій, баз даних) безпосередньо в коді C#, що значно спрощує роботу з даними;

– підтримка розробки веб-додатків: За допомогою фреймворку ASP.NET Core розробники можуть створювати як веб-сайти, так і веб-серверні додатки з використанням C#;

– розширюваність через бібліотеки: Через велику кількість стандартних та сторонніх бібліотек, C# надає можливості для розширення функціональності вашого додатку без необхідності переписування коду з нуля;

– сокети - це механізм, який дозволяє здійснювати мережеве взаємодію між програмами. У контексті мультикомп'ютерної системи з топологією "кільце" сокети є важливим інструментом для забезпечення комунікації між вузлами мережі. Для реалізації однонаправленого зв'язку використовуються сокети типу UDP або TCP;

– UDP (User Datagram Protocol): Використовується для надсилання невеликих пакетів даних без гарантії доставки. Він підходить для додатків, які не вимагають надійності або не чутливі до затримок;

– TCP (Transmission Control Protocol): Забезпечує надійний, послідовний та орієнтований на з'єднання потік даних між вузлами мережі. TCP ідеально підходить для додатків, які потребують надійної доставки та контролю за порядком пакетів.

У мережевій топології "кільце" кожен вузол підключений до двох сусідніх вузлів, створюючи замкнену лінію. Сокети дозволяють цим вузлам обмінюватися

даними, використовуючи відповідний протокол передачі. Наприклад, якщо вузол А хоче відправити дані вузлу В, він може створити сокет та відправити дані через цей сокет, що потім будуть передані по кільцю до вузла В.

Залежно від вимог конкретного додатку можна вибрати між UDP та TCP. Наприклад, якщо додаток просто надсилає дані та не вимагає гарантованої доставки, UDP може бути більш ефективним варіантом завдяки своїй простоті та меншому накладу на мережевий трафік. У той час як, для додатків, які потребують гарантованої доставки та контролю за порядком пакетів, TCP може бути більш підходящим варіантом.

Обираючи C# як основну мову програмування для розробки системи, ми отримуємо доступ до широкого спектра інструментів та фреймворків, які значно полегшують процес створення програмного забезпечення. .NET Core, у свою чергу, забезпечує можливість розробки кроссплатформених додатків, що розширює спектр використання нашого рішення на різних операційних системах.

Крім того, велика спільнота розробників C# гарантує наявність підтримки та численних ресурсів для навчання та вирішення технічних проблем. Це дозволяє швидко знаходити відповіді на питання та користуватися багатим досвідом інших розробників, що сприяє прискоренню процесу розробки та підвищенню якості кінцевого продукту.

Система типізації та об'єктно-орієнтований підхід у C# сприяють створенню більш стабільного та безпомилкового коду, що зменшує витрати на виправлення помилок та підвищує надійність програмного забезпечення. Строга типізація допомагає виявляти помилки на ранніх етапах розробки, що значно знижує ймовірність виникнення критичних помилок у виробничій експлуатації.

Застосування асинхронного програмування дозволяє створювати більш продуктивні та ефективні додатки, які можуть обробляти великі обсяги даних та забезпечувати високу швидкість реакції на запити користувачів. Це особливо важливо для систем, які працюють у реальному часі та обробляють численні паралельні запити.

Таким чином, вибір C# як основної мови програмування та використання .NET Core забезпечує високу продуктивність, гнучкість та надійність нашого програмного забезпечення, що відповідає сучасним вимогам та стандартам індустрії.

3.3 Моделювання різних ситуацій використання мулькомп'ютерних систем по архітектурі кільце однонаправленого типу

Сценарій 1 (рисунок 3.2): Перевірка підключення компонентів у мультикомп'ютерній системі на основі архітектури кільця однонаправленого типу.

Підключення комп'ютерів безпосередньо. Підключіть перший комп'ютер до другого за допомогою кабелю Ethernet. Використовуйте мережеві порти на задній панелі кожного комп'ютера для цього підключення.

Налаштування мережевих параметрів. На кожному комп'ютері встановіть унікальні IP-адреси в одній локальній мережі. Наприклад, ви можете використовувати IP-адреси з підмережі 192.168.1.0/24, наприклад, 192.168.1.1 для першого комп'ютера та 192.168.1.2 для другого.

Запуск програми на кожному комп'ютері. Скопіюйте програму, написану на C#, на кожен комп'ютер. Відкрийте програму на кожному комп'ютері і збережіть її.

Налаштування IP адрес. Налаштуйте на кожному комп’ютері IP адресу сусіда і IP адресу останього поставте IP адресу першого.

Запуск програми. Запустіть програму на кожному комп'ютері. Вона повинна з'єднатися з іншим комп'ютером за допомогою IP-адреси, яку ви вказали у програмі, і виконати необхідні дії.

Цей підхід дозволяє вам запустити програму на кожному комп'ютері та налаштувати IP-адреси для з'єднання між ними. Не забудьте переконатися, що мережеві налаштування та програма на кожному комп'ютері відповідають одне одному, щоб забезпечити успішне з'єднання та обмін даними.

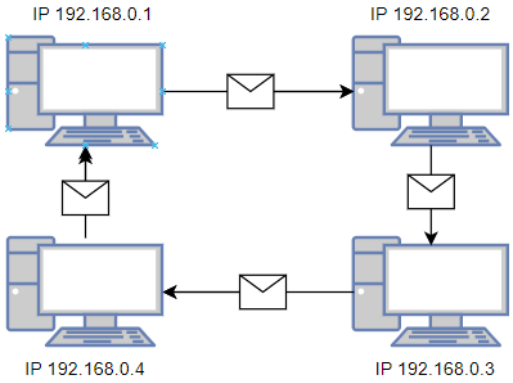


Рисунок 3.2 – Сценарій 1

Після налаштування та запуску програми на кожному комп'ютері важливо перевірити, чи встановлені з'єднання працюють належним чином. Ви можете використати команду ping для перевірки з'єднання між комп'ютерами. Наприклад, з першого комп'ютера відправте ping на IP-адресу другого комп'ютера і навпаки. Якщо з'єднання встановлено успішно, ви побачите відповіді від кожного комп'ютера.

Якщо програма не може встановити з'єднання або не виконує необхідні дії, перевірте налаштування IP-адрес, кабельні з'єднання та мережеві порти. Можливо, необхідно оновити драйвери мережевих адаптерів або змінити мережеві кабелі. Також варто перевірити налаштування брандмауера або антивірусного програмного забезпечення, яке може блокувати мережевий трафік.

Для забезпечення стабільної роботи мультикомп'ютерної системи розгляньте можливість оптимізації налаштувань мережі та програми. Це може включати налаштування швидкості передачі даних, зменшення затримок та оптимізацію обробки даних у програмі. Також варто розглянути використання якісного мережевого обладнання, яке підтримує високу швидкість передачі даних і забезпечує надійність з'єднання.

Такий підхід забезпечує надійне з'єднання між комп'ютерами в мультикомп'ютерній системі на основі архітектури кільця однонаправленого типу,

дозволяючи виконувати розподілені обчислення та забезпечуючи ефективний обмін даними між компонентами системи.

Сценарій 2 (рисунок 3.3). Програмне забезпечення, яке виконує обчислення, необхідні для користувача на одній комп'ютерній станції, здійснює оцінку завантаженості кожного окремого комп'ютера, підключеного до мережі, і розподіляє завдання для вирішення цих обчислень між ними, утворюючи кільце з'єднаних комп'ютерів. Алгоритм виконання:

– користувач вводить математичну формулу для виконання на своїй комп'ютерній станції;

– програмне забезпечення, що працює на комп'ютерній станції, аналізує навантаженість кожного комп'ютера у кільці;

– програмне забезпечення надсилає запит до кожного комп'ютера у кільці, щоб отримати інформацію про його навантаженість;

– програмне забезпечення відбирає найменш навантажений комп'ютер у кільці для виконання завдання;

– результат викоання надсилається з комп'ютерної станції до вибраного комп'ютера через мережевий протокол (наприклад, TCP/IP);

– обраний комп'ютер отримує запит на обробку завдання та проводить розрахунки за формулою;

– обраний комп'ютер повертає результат обчислення до програмного забезпечення на комп'ютерній станції;

– програмне забезпечення на комп'ютерній станції отримує результат і відображає його користувачу.

Для забезпечення безперебійної роботи системи важливо регулярно перевіряти стан кожного комп'ютера у кільці. Це включає перевірку апаратного забезпечення, оновлення програмного забезпечення, а також моніторинг мережевих з'єднань. Регулярне обслуговування допоможе уникнути проблем з продуктивністю і забезпечить коректну роботу алгоритму розподілу завдань.

Для забезпечення масштабованості системи варто розглянути можливість додавання нових комп'ютерів у кільце без значних змін у програмному забезпеченні. Це може включати автоматичне виявлення нових комп'ютерів у мережі та динамічне оновлення інформації про їх навантаженість. Такий підхід дозволить легко розширювати систему і забезпечувати її високу продуктивність навіть при збільшенні кількості завдань і комп'ютерів у кільці [36].

Такі заходи допоможуть забезпечити стабільну та ефективну роботу мультикомп'ютерної системи, яка використовує кільцеву архітектуру для розподілу обчислювальних завдань між комп'ютерами, підвищуючи продуктивність і зручність для користувачів.

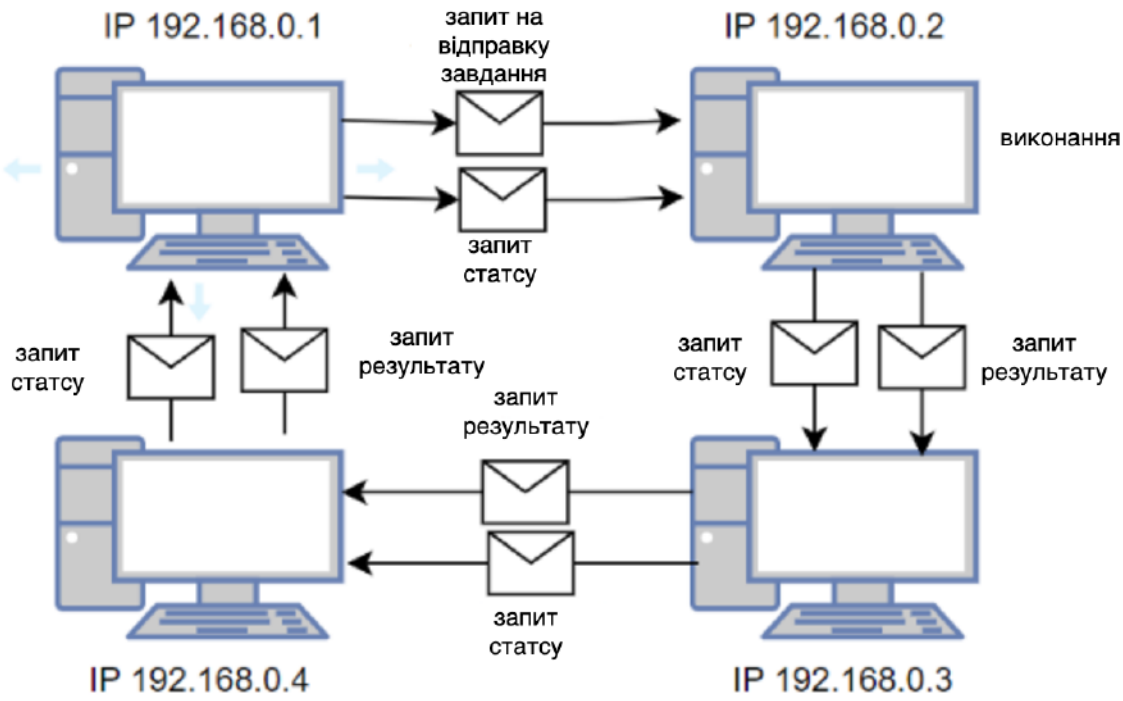


Рисунок 3.3 – Сценарій 2

Сценарій 3. Колективна робота над проектом (рисунок 3.4). Розглядається впровадження проміжного програмного забезпечення для кооперативної роботи над проектом у мультикомп’ютерній системі з топологією "кільце" однонаправленого зв’язку. Ця система дозволяє користувачам з різних комп’ютерів

спільно працювати над проектом, обмінюватися даними та виконувати спільні завдання [37]. Алгоритм виконання:

– кожен користувач запускає спеціалізоване програмне забезпечення на своєму комп’ютері;

– кожен комп’ютер підключається до мультикомп’ютерної системи з топологією "кільце" через мережевий протокол, такий як TCP/IP;

– комп’ютери в мультикомп’ютерній системі утворюють кільцеву топологію, де кожен комп’ютер з'єднаний з попереднім і наступним комп’ютерами;

– користувачі можуть обмінюватися файлами, виконувати спільні завдання та обмінюватися даними. Кожен користувач може вносити зміни у спільні файли або створювати нові файли;

– після збереження змін або виконання спільного завдання програмне забезпечення розподіляє ці зміни по мультикомп’ютерній системі;

– кожен комп’ютер у кільці отримує зміни та оновлює локальні файли або виконує відповідні дії.

У мережі з топологією "кільце" однонаправленого зв’язку кожен комп'ютер має змогу передавати дані лише наступному в логічному порядку. Проміжне програмне забезпечення допомагає керувати цим процесом, забезпечуючи ефективний обмін даними та координацію роботи над проектом [38].

Така топологія мережі іноді використовується там, де потрібно забезпечити надійність зв'язку та простоту управління. Проте, одним з потенційних недоліків цієї структури є залежність від кожного комп'ютера в мережі: відмова одного комп'ютера може призвести до переривання зв'язку для всієї мережі. Також, у разі великої кількості комп'ютерів у мережі, керування та координація можуть стати складними завданнями для проміжного програмного забезпечення. Тим не менш, з правильним налаштуванням та обслуговуванням, мережа з топологією "кільце" може ефективно працювати для виконання специфічних завдань.

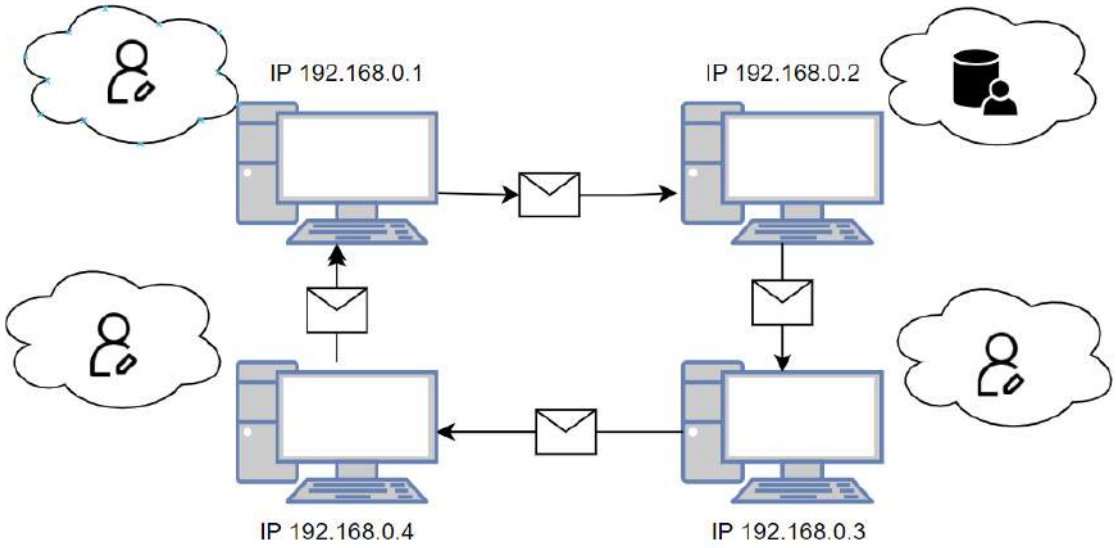


Рисунок 3.4 – Сценарій 3

Таке проміжне програмне забезпечення відіграє ключову роль у забезпеченні злагодженої роботи команди. Воно гарантує, що всі користувачі мають доступ до актуальних версій файлів та можуть вчасно отримувати оновлення. Це особливо важливо для проектів, що вимагають високого рівня координації та спільної роботи, наприклад, у розробці програмного забезпечення, проектуванні інженерних систем або управлінні великими дослідницькими проектами [39].

Застосування топології "кільце" забезпечує надійність передачі даних та зменшує затримки, оскільки інформація передається послідовно від одного комп’ютера до іншого. Це також дозволяє уникнути перевантаження мережі та забезпечити стабільність роботи системи навіть при значному обсязі передаваних даних.

Впровадження такої системи може вимагати початкових інвестицій у налаштування та навчання персоналу, але згодом ці витрати окупляться за рахунок підвищення продуктивності та ефективності роботи команди [40].

Сценарій 4. Обмін ресурсами (рисунок 3.5). Розглядається впровадження проміжного програмного забезпечення для спільного використання ресурсів у мультикомп'ютерній системі з топологією "кільце" однонаправленого зв’язку. Ця

система дозволяє користувачам з різних комп'ютерів обмінюватися ресурсами та використовувати їх спільно. Алгоритм виконання:

– кожен користувач запускає спеціалізоване програмне забезпечення на своєму комп'ютері;

– кожен комп'ютер підключається до мультикомп'ютерної системи з топологією "кільце" через мережевий протокол, такий як TCP/IP;

– комп'ютери в мультикомп'ютерній системі утворюють кільцеву топологію, де кожен комп'ютер з'єднаний з попереднім і наступним комп'ютерами;

– користувачі можуть надавати ресурси для спільного використання, такі як друкуючі пристрої, файлові сховища, обчислювальні потужності тощо;

– інші користувачі в мережі можуть звертатися до цих ресурсів за допомогою проміжного програмного забезпечення;

– після завершення використання ресурсів програмне забезпечення автоматично вивільняє їх і повертає систему до початкового стану;

– кожен комп'ютер у кільці отримує доступ до ресурсів, які надаються іншими користувачами, і може використовувати їх за потреби.

Проміжне програмне забезпечення у мережі з топологією "кільце" дозволяє ефективно керувати доступом до ресурсів, забезпечуючи їх спільне використання без конфліктів. Користувачі можуть легко обмінюватися файлами, принтерами та іншими ресурсами, оптимізуючи робочий процес та підвищуючи продуктивність.

При використанні проміжного програмного забезпечення в мережі з топологією "кільце" однонаправленого зв’язку забезпечується надійний та безперервний доступ до ресурсів незалежно від місця розташування користувачів. Це сприяє підвищенню ефективності використання ресурсів та зменшенню часу, необхідного для взаємодії між користувачами. Крім того, використання проміжного програмного забезпечення у такій мережі дозволяє автоматизувати процеси маршрутизації та керування трафіком, що сприяє зниженню навантаження на самі вузли мережі. Це в свою чергу забезпечує більш стабільну та швидку передачу даних між користувачами.

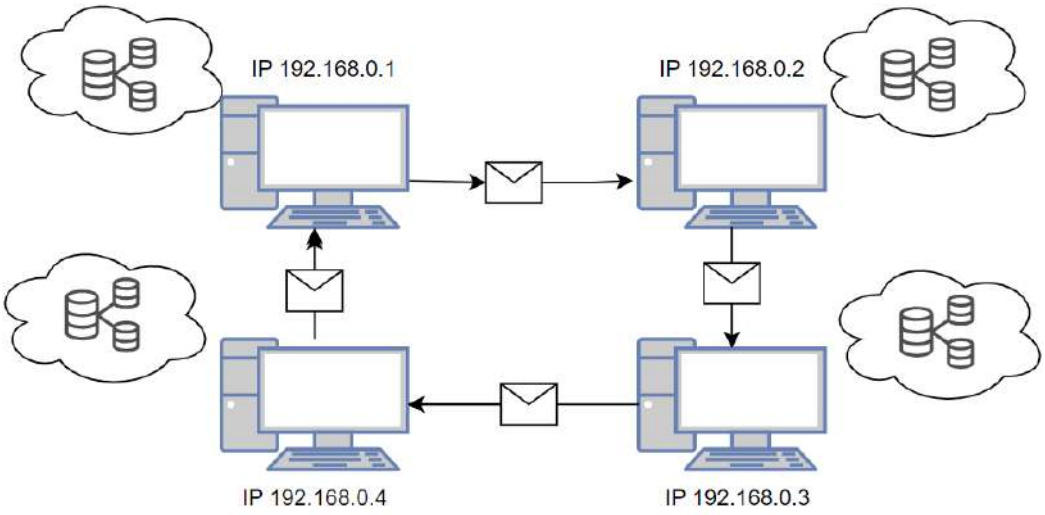


Рисунок 3.5 – Сценарій 4.

Впровадження проміжного програмного забезпечення для спільного використання ресурсів у мультикомп'ютерній системі з топологією "кільце" має низку переваг. Перш за все, така система забезпечує ефективний розподіл навантаження між комп'ютерами. Кожен користувач може отримати доступ до ресурсів, які знаходяться на інших комп'ютерах, що дозволяє уникнути перевантаження окремих вузлів і забезпечити рівномірний розподіл ресурсів. Це особливо важливо в середовищах, де доступ до ресурсів критичний, таких як науково-дослідні лабораторії чи великі корпоративні мережі [41].

Друга важлива перевага полягає в підвищенні надійності системи. Завдяки топології "кільце" кожен комп'ютер з'єднаний з двома іншими, що дозволяє продовжувати роботу навіть у випадку відмови одного з вузлів. Проміжне програмне забезпечення може бути налаштоване таким чином, щоб автоматично перенаправляти запити через альтернативні маршрути, забезпечуючи безперервність обслуговування. Це знижує ризики простоїв та втрат даних, що критично для забезпечення стабільної роботи мережі [42].

Третім важливим аспектом є гнучкість і масштабованість системи. Додавання нових комп'ютерів до кільця можна здійснити без значних змін у конфігурації мережі. Нові користувачі можуть швидко підключитися та почати надавати або

використовувати ресурси, що сприяє швидкому розширенню системи. Це робить мультикомп'ютерну систему з топологією "кільце" ідеальним вибором для організацій, які планують рости і збільшувати свої обчислювальні потужності з часом.

Сценарій 5. Динамічне балансування навантаження. Розглядається впровадження проміжного програмного забезпечення для динамічного балансування навантаження у мультикомп'ютерній системі з топологією "кільце" однонаправленого зв’язку. Ця система дозволяє автоматично перерозподіляти обчислювальні завдання та ресурси між комп'ютерами, щоб забезпечити оптимальну продуктивність і уникнути перевантаження окремих вузлів [43].

Алгоритм виконання:

– кожен комп'ютер в мультикомп'ютерній системі запускає агента балансування навантаження;

– кожен агент моніторить використання ресурсів свого комп'ютера, таких як CPU, пам'ять, мережеві з'єднання і диск;

– комп'ютери утворюють кільцеву топологію, де кожен комп'ютер з'єднаний з попереднім і наступним комп'ютерами через мережевий протокол, такий як TCP/IP;

– при виявленні перевантаження на будь-якому з вузлів агент балансування надсилає сигнал сусіднім комп'ютерам у кільці з проханням про допомогу;

– сусідні комп'ютери перевіряють свої ресурси і, якщо можуть прийняти частину навантаження, надсилають відповідний сигнал з підтвердженням;

– завдання перерозподіляються між комп'ютерами таким чином, щоб рівномірно розподілити обчислювальне навантаження по всьому кільцю;

– система автоматично оновлює стан балансування у реальному часі, постійно коригуючи розподіл завдань, щоб уникати перевантаження окремих вузлів;

– у разі відмови одного з вузлів, система перенаправляє його завдання до інших вузлів у кільці, забезпечуючи безперервність виконання обчислювальних процесів.

Впровадження проміжного програмного забезпечення для динамічного балансування навантаження має низку переваг:

– підвищення продуктивності. Динамічний розподіл завдань дозволяє уникнути перевантаження окремих вузлів, що підвищує загальну продуктивність системи [44];

– надійність. Завдяки можливості перенаправлення завдань у разі відмови одного з вузлів, забезпечується безперервність обчислювальних процесів, знижуючи ризики простоїв;

– оптимізація використання ресурсів. Система забезпечує ефективне використання обчислювальних потужностей, що дозволяє зменшити витрати на підтримку обладнання;

– масштабованість. Легке додавання нових комп'ютерів до кільця дозволяє швидко збільшити обчислювальні потужності системи без значних змін у конфігурації мережі;

– гнучкість. Система може адаптуватися до змін у навантаженні в реальному часі, забезпечуючи оптимальну продуктивність у будь-який момент [45].

Таким чином, впровадження проміжного програмного забезпечення для динамічного балансування навантаження у мультикомп'ютерній системі з топологією "кільце" є ефективним рішенням для забезпечення стабільної та продуктивної роботи мережі у різних умовах.

3.4 Архітектура програмного забезпечення

Програмне забезпечення призначене для вирішення завдання математичних виразів на будь-якій з трьох комп'ютерних станцій, доступних користувачеві. При цьому користувач може самостійно вибрати, на якій конкретно станції виконати

операцію. Це забезпечує більшу гнучкість і контроль над процесом обчислення. Навіть без оцінки завантаження кожної станції, програма все ж враховує потреби користувача, дозволяючи йому вибрати оптимальний варіант в залежності від його потреб та вимог [46].

Цей код реалізує мережеву взаємодію між комп’ютерами, за допомогою TCP з'єднань. Основні компоненти цього коду відповідають класам, які вказані у UML діаграмі.

Для досягнення високої продуктивності та надійності, архітектура програмного забезпечення передбачає розподілену обробку даних. Кожна з комп'ютерних станцій виступає як окремий вузол у мережі, що дозволяє виконувати обчислення паралельно і незалежно один від одного. Це значно скорочує час обробки складних математичних виразів і підвищує загальну ефективність системи[38].

Основні компоненти системи включають:

– клієнтський додаток. Інтерфейс, через який користувач взаємодіє з системою, вводить математичні вирази і вибирає станцію для виконання обчислень;

– серверна частина. Забезпечує зв'язок між клієнтським додатком і обчислювальними станціями, розподіляє завдання і координує їх виконання;

– обчислювальні вузли. Комп'ютерні станції, які безпосередньо виконують обчислення, отримані від серверної частини.

Архітектура програмного забезпечення, описана вище, сприяє ефективному використанню обчислювальних ресурсів і забезпечує гнучкість у виборі оптимального вузла для виконання обчислень [47]. Клієнтський додаток надає зручний інтерфейс для користувача, в якому він може вводити математичні вирази та вибирати конкретну обчислювальну станцію. Серверна частина відповідає за координацію роботи між клієнтським додатком і обчислювальними вузлами, розподіляючи завдання на виконання та забезпечуючи зв'язок між ними через TCP з'єднання. Обчислювальні вузли, у свою чергу, виконують отримані завдання

незалежно один від одного, що дозволяє паралельно обробляти складні математичні вирази і підвищує загальну ефективність системи [48]. Така розподілена архітектура дозволяє досягти високої продуктивності та надійності у вирішенні математичних завдань.

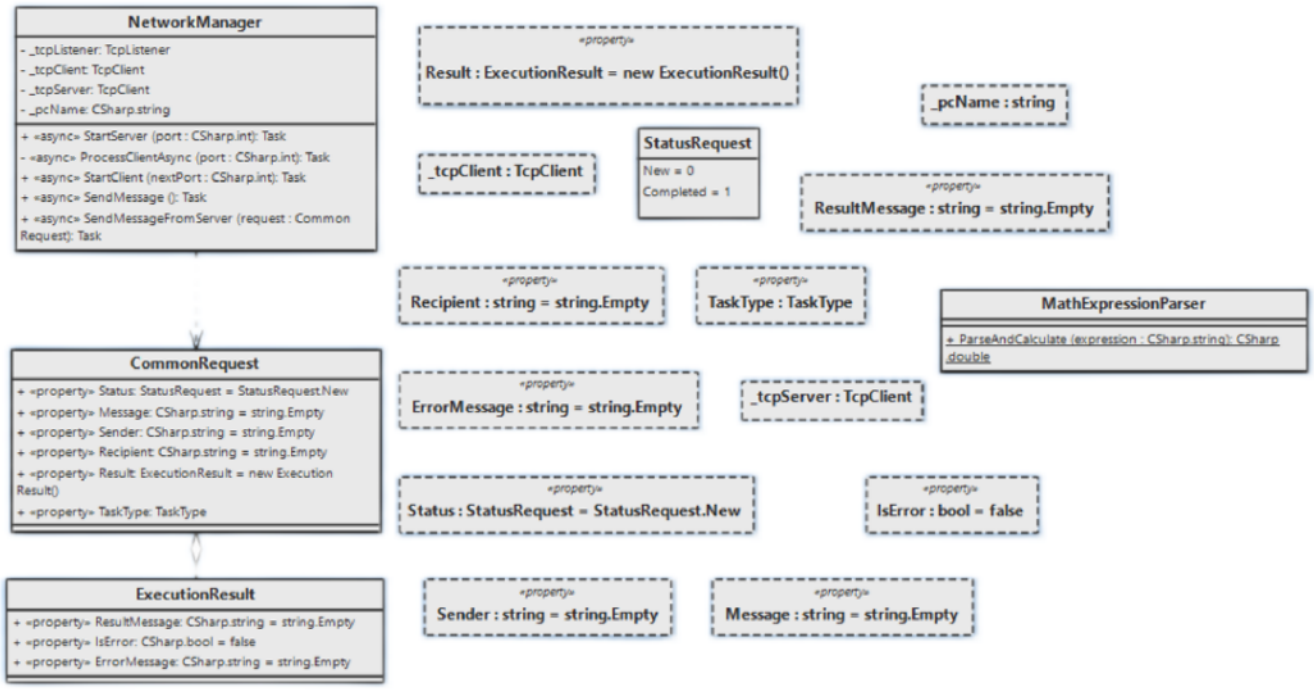


Рисунок 3.10 – UML-діаграма архітектури програмного забезпечення

Клас NetworkManager відповідає за управління мережевими з'єднаннями. У ньому є методи для запуску сервера (StartServer) та клієнта (StartClient), а також для відправлення повідомлень (SendMessage). Він також обробляє отримані повідомлення через сервер та відправляє їх до клієнта або сервера за необхідності.

Клас CommonRequest представляє структуру повідомлення, яка використовується для обміну даними між сервером та клієнтом. Він містить інформацію про стан запиту, текст повідомлення, відправника, одержувача, тип завдання та результат виконання.

StatusRequest та TaskType є перерахуваннями, які використовуються для визначення статусу запиту та типу завдання відповідно.

Клас ExecutionResult містить інформацію про результат виконання завдання, включаючи текстове повідомлення, прапорець помилки та повідомлення про помилку.

MathExpressionParser - це допоміжний клас, який використовується для розбору та обчислення математичних виразів.

Цей код реалізує просту мережеву систему, яка може обмінюватися повідомленнями між комп’ютерами мультикомп’ютерної системи типу кільце однонаправленого типу та виконувати обчислення математичних виразів.

Налаштування:

– код визначає словник idToPortDictionary, який співвідносить ID комп'ютера (1-10) з номером порту (8080-8089);

– він запитує у користувача ввести ID свого ПК та перевіряє, чи він дійсний (від 1 до кількості записів у словнику);

– запитує у користувача, чи це останній комп'ютер у кільці, і зберігає відповідь.

Клас Network Manager. Серверна сторона (метод StartServer). Прослуховує вхідні з'єднання на певному порту на основі наданого ID ПК. Коли встановлено з'єднання, він читає повідомлення, надіслані іншими комп'ютерами. Він може обробляти повідомлення двох типів. Повідомлення, які ще не досягли пункту призначення (перенаправлені повідомлення). Повідомлення, що походять з цього комп'ютера (оригінальні повідомлення). Для перенаправлених повідомлень він оновлює історію маршрутизації та перевіряє ID одержувача. Якщо це одержувач, він обчислює вираз у повідомленні та надсилає відповідь назад. В іншому випадку він перенаправляє повідомлення на наступний комп'ютер у кільці. Для оригінальних повідомлень він додає ID ПК та мітку часу до історії маршрутизації, надсилає повідомлення комп'ютеру-одержувачу, а також відображає історію маршрутизації та графічне представлення.

Сторона клієнта (метод StartClient). Підключається до сервера на порту наступного комп'ютера в кільці. Надсилання повідомлень (метод SendMessage).

Постійно запитує у користувача ввести тип завдання (наразі підтримує лише вирази). На основі типу завдання отримує вираз, ID одержувача та створює об'єкт CommonRequest. Цей об'єкт містить інформацію про повідомлення, відправника, одержувача, історію маршрутизації тощо. Об'єкт CommonRequest потім серіалізується у формат JSON та надсилається на сервер через встановлене з'єднання.

Класи даних:

– CommonRequest: Цей клас містить інформацію про повідомлення, яке обмінюється;

– StatusRequest: Перелічування, яке вказує на стан запиту (новий або завершений);

– TaskType: Перелічування, яке вказує на тип завдання, пов'язаного з повідомленням (наразі підтримуються лише вирази);

– RouteEntry: Цей клас зберігає інформацію про одну зупинку (ID ПК та мітку часу) в історії маршрутизації повідомлення;

– ExecutionResult: Цей клас містить результат виконання завдання (оцінка виразу);

– MathExpressionParser: Цей клас має метод для оцінки математичного виразу, отриманого в повідомленні.

Загальна функціональність:

– користувачі можуть запускати цю програму на кількох комп'ютерах у кільцевій мережі;

– кожен комп'ютер діє як клієнт і сервер;

– користувач на одному комп'ютері може надіслати математичний вираз на інший комп'ютер у кільці;

– повідомлення проходить через кожен комп'ютер у кільці, доки не досягне комп'ютера-одержувача;

– комп'ютер-одержувач оцінює вираз, надсилає результат назад, а результат проходить кільце назад до оригінального комп'ютера.

Цей код дозволяє користувачам у кільцевій мережі надсилати один одному математичні вирази та отримувати оцінені результати. Основні можливості. Вибір ID ПК та перевірка його дійсності. Визначення, чи є ПК останнім у кільці. Запуск сервера та клієнта на кожному ПК. Передача повідомлень через порти TCP. Обробка перенаправлених та оригінальних повідомлень [49]. Введення типу завдання (вираз). Введення виразу, ID одержувача та створення запиту. Серіалізація та надсилання запиту на сервер. Оцінка виразів для оригінальних повідомлень [50]. Оновлення історії маршрутизації та статусу запиту. Перенаправлення або надсилання відповідей залежно від типу повідомлення. Історія маршрутизації та графічне представлення. Результат оцінки виразу.

Окрім результату виконання (рисунок 3.11) завдання (оцінка виразу), користувач може отримати доступ до додаткової інформації, що робить код більш інформативним та зручним у використанні. Доступна інформація може включати:

– історія маршрутизації: показує шлях, яким пройшло повідомлення, та час його проходження через кожен комп'ютер;

– матриця: візуалізує зв'язки між комп'ютерами в кільці;

– граф: графічно представляє структуру мережі;

– час затримки: показує час, необхідний для доставки повідомлення;

– додаткові відомості: ID завдання, час надсилання, помилки, детальна маршрутизація тощо.

Ця інформація може допомогти користувачам краще зрозуміти роботу мережі, діагностувати проблеми, оптимізувати маршрутизацію, аналізувати продуктивність та відстежувати виконання завдань.

Наявність візуалізацій, таких як матриця та граф, може значно покращити зручність використання. Використання візуалізацій, таких як матриці та графіки, може допомогти користувачам отримати більш чітке уявлення про структуру мережі та потік даних через неї. Це сприяє зручності аналізу, діагностики проблем і вдосконаленню маршрутизації. Візуалізації також дозволяють швидше виявляти

та вирішувати проблеми у мережі, що сприяє підвищенню її надійності та ефективності.

Рисунок 3.11 –Результат програмного забезпечення

3.5. Висновки

З розділу "Проектування програмно-технічного засобу" очевидно, що детальне планування і розробка компонентів мультикомп'ютерної системи є ключовим етапом у створенні ефективного програмно-технічного засобу. Алгоритм під'єднання складових системи, вибір програмних засобів та моделювання різних сценаріїв використання допомагають забезпечити оптимальну архітектуру і функціональність. Важливо також розглянути архітектуру програмного забезпечення, щоб забезпечити його стабільність, надійність і масштабованість. Даний розділ містить ключові кроки у розробці програмно-технічного засобу, які допомагають забезпечити успішну і ефективну реалізацію мультикомп'ютерної системи.

Крім того, варто відзначити, що правильний вибір програмних засобів є важливою складовою успішного проектування програмно-технічного засобу. Відповідно до специфіки та вимог проекту слід ретельно розглянути різноманітні варіанти програмних рішень і вибрати ті, що найкраще відповідають потребам системи.

Крім того, моделювання різних ситуацій використання мультикомп'ютерних систем дозволяє передбачити можливі проблеми та ефективно вирішити їх на етапі проектування. Цей підхід сприяє покращенню якості та надійності програмно-технічного засобу, а також допомагає зменшити ризики виникнення проблем у процесі експлуатації системи. Таким чином, відповідальне підхід до проектування програмно-технічного засобу забезпечує його успішну реалізацію та оптимальне функціонування в майбутньому.

ВИСНОВКИ

Електронні обчислювальні машини (ЕОМ) та обчислювальна техніка є важливим елементом у сучасному світі, знаходячи застосування у різних сферах діяльності, таких як бізнес, наука, медицина та інженерія. Вони забезпечують збір, обробку, зберігання та передачу інформації.

Електронні обчислювальні машини (ЕОМ) та обчислювальна техніка є важливим елементом у сучасному світі, знаходячи застосування у різних сферах діяльності, таких як бізнес, наука, медицина та інженерія. Вони забезпечують збір, обробку, зберігання та передачу інформації. Розвиток ЕОМ значно підвищив ефективність та продуктивність у цих галузях, дозволяючи виконувати складні обчислення та аналіз даних за короткий час.

У роботі було розглянуто мультикомп'ютерні системи, які є важливою складовою обчислювальної техніки. Мультикомп'ютерні системи складаються з багатьох комп'ютерів, які працюють як єдине ціле, маючи переваги високого рівня масштабованості і надійності. Вони дозволяють розподіляти навантаження між багатьма вузлами, що значно підвищує продуктивність і забезпечує безперервну роботу системи навіть у випадку виходу з ладу окремих компонентів.

Основною метою роботи було створення мультикомп'ютерної системи загального призначення на основі топології "подвійне кільце". Виконаний детальний аналіз різних архітектур та вибір апаратно-програмної бази, що відповідає цій топології. Топологія "подвійне кільце" забезпечує високий рівень надійності та стійкості системи, оскільки кожен вузол має два незалежні зв'язки з іншими вузлами, що дозволяє зберігати працездатність системи навіть у випадку обриву одного з каналів зв'язку.

У першому розділі був проведений аналіз обчислювальних систем, фокусуючись на класифікації Флінна та різних типах паралельної обробки. Це дозволило зрозуміти основні принципи побудови мультикомп'ютерних систем та обрати найбільш відповідну архітектуру для вирішення поставлених завдань. В

другому розділі детально розглянута топологія "кільце" та її похідна, топологія "кільце однонаправленого типу", з описом обраної архітектури та вибором апаратного забезпечення. Такий підхід забезпечує високий рівень відмовостійкості та гнучкості системи.

У другому розділі було розглянуто різні види мультикомп'ютерних систем, зокрема, кластери, багатопроцесорні системи та мережеві комп'ютери. Було проаналізовано їх особливості, переваги та недоліки, що допомогло визначити найбільш відповідну топологію для створення ефективної мультикомп'ютерної системи. Особлива увага була приділена топології "кільце" та її похідній, топології "кільце однонаправленого типу ", з описом обраної архітектури та вибором апаратного забезпечення. Такий підхід забезпечує високий рівень відмовостійкості та гнучкості системи.

У третьому розділі було описано розроблення алгоритму підключення компонентів та програмного забезпечення. Це включало створення мережевих протоколів для забезпечення ефективного обміну даними між вузлами системи, а також розробку спеціалізованого програмного забезпечення для управління системою. Використання сучасних технологій та підходів дозволило створити систему, яка може ефективно використовуватися у різних сферах діяльності, від наукових досліджень до комерційних застосувань.

Отже, результатом роботи є створення мультикомп'ютерної системи за топологією "кільце однонаправленого типу ", що має потенціал для ефективного застосування у різних сферах діяльності. Запропонована система забезпечує високу продуктивність, надійність та гнучкість, що робить її привабливим вибором для широкого кола користувачів. Перспективи подальшого розвитку включають оптимізацію існуючих алгоритмів та розробку нових методів для підвищення ефективності системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Ямагучі Х., Уеда Т., Хігасіно Т. Проміжне програмне забезпечення для реалізації та оцінки протоколів мультікасту на рівні додатків у реальних середовищах. Осацький університет. 2024. 329 с.

2.Ornelas P., Nape I., Forbes E. Nonlocal skyrmions as topologically stable quantum entangled states of light. *Nature Photonics*. 2024. Т. 18, № 3. P. 123–130. DOI: 10.1038/s41566-023-01360-4

3. Шен Х. Дослідження високопродуктивних мереж і розподілених обчислень. Університет Аделаїди. 2023. 543 с. URL: https://researchers.adelaide.edu.au/profile/hong.shen (дата звернення: 15.04.2024).

4. 5G розумні антени: аналіз галузі, розмір ринку, частка, тенденції, зростання та прогноз 2021-2026. IndustryARC. 2022. 143 с. URL: https://www.industryarc.com (дата звернення: 10.05.2024).

5. Сеті П., Саранг С. Р. Систематичний огляд літератури про шлюзи IoT ScienceDirect. 2021. 543 с.

6. Дослідження квантового зв'язку з використанням топології. Phys.org. 2021. 386 с. URL: https://phys.org/news/2024-01-quantum-entanglement-topology-inextricably-linked.html (дата звернення: 12.05.2024).

7.Фудзі С., Уеда Т., Хігасіно Т. Гібридний тестовий стенд для забезпечення операцій в режимі реального часу для бездротових додатків. Осацький університет. 2020. 312 с.

8. Ікеда, К., Бадуге, Т. М., Хігасіно, Т. Стійкий накладний мультікаст для мультимедійної трансляції в умовах множинних джерел: дисертація на здобуття наукового ступеня доктора технічних наук за спеціальністю 05.13.05 - Комп'ютерні системи та компоненти. Осацький університет, 2017. 189 с.

9.Бадуге Т. М., Ікеда К., Ямагучі Х. Ефективне паралельне моделювання мобільних бездротових мереж за допомогою прогнозування затримки багаторазового поширення під час виконання. Осацький університет. 2019. 432 с.

10. Грін С. Розподілені обчислення: практичні аспекти. – Берлін: Спрінгер, 2021. 600 с.

11.Шен Х. Дослідження високопродуктивних мереж і розподілених обчислень. Університет Аделаїди. 2023. 245 р. URL: https://researchers.adelaide.edu.au/profile/hong.shen (дата звернення: 19.04.2024).

12. Сінгх А. Мультикомп'ютерні системи: новітні підходи і рішення. – Токіо: Кіодо, 2020. 670 с.

13.Posture Estimation using a 3D Range Camera for Standing-Up Motion Assist Robot. IEEE International Conference on Robotics and Biomimetics. 2012. 343 p. URL: https://researchers.uec.ac.jp (дата звернення: 23.04.2024).

14. Кім Дж. Паралельні алгоритми і структури даних. – Сіетл: Амазон Пресс, 2023. 780 с.

15. Zherenko, V., Kryapa, S. Middleware Solutions for Real-Time Data Processing in Distributed Systems with Ring Topology. *Journal of Systems and Software*. 2023. №74. P. 2-6. DOI: 10.1016/j.jss.2023.111113.

16. Чен С. Високопродуктивні обчислювальні системи: дизайн і оптимізація. – Лондон: Ельсевір, 2022. 890 с.

17.Serrano, V., Gomez, S., Juana, F. Middleware Solutions for Efficient Data Management in Distributed Systems. *Journal of Systems and Software*. 2023. №82. P. 22-32. DOI: 10.1016/j.jss.2023.111234.

18.Martinez, P., Lopez, R., Rodriguez, A. Adaptive Middleware for Real-Time Communication in Industrial IoT Systems. *IEEE Transactions on Industrial Informatics*. 2022. №117. P. 25-29. DOI: 10.1109/TII.2022.3171234.

19.Chang, S., Lee, Y., Kim, T. A Scalable Middleware for Big Data Analytics in Smart Cities. *Future Generation Computer Systems*. 2022. №87. P. 11-13. DOI: 10.1016/j.future.2022.103456.

20.Huan, L., Wong, M., Cyan, D. Middleware for Efficient Data Aggregation in Wireless Sensor *Networks. Sensors*. 2021. №75. P. 4-10. DOI: 10.3390/s210930678.

21.Li, Y., Hu, W., Wong, L. Reliable Middleware for Distributed Systems with Ring Topology. *Computer Communications*. 2021. №93. P. 30-36. DOI: 10.1016/j.comcom.2021.104567.

22. Ramirez, M., Sanchez, J., Porras, E. Middleware Solutions for Energy-Efficient *Communication in IoT Systems. Energies*. 2020. №12. P. 15-19. DOI: 10.3390/en13164050.

23.Kayhan E. Middleware Solutions for Real-Time Data Processing in Distributed Systems with Ring Topology. *Journal of Systems and Software*. 2023. №3. P. 4-8. DOI: 10.1016/j.jss.2023.111113.

24. Тейлор Р., Лі Т. Паралельні обчислювальні системи: від теорії до застосування. – Бостон: Бостонський технологічний інститут, 2019. 750 с.

25. Чжан К. Сучасна комп'ютерна архітектура: теорія і практика. – Кембридж: Кембриджський університет, 2021. 820 с.

26. Postma, J., Vermeer, J., Decker, K. Middleware for Network Security in Distributed Systems. *Journal of Network and Computer Applications*. 2020. №54. P. 6-8. DOI: 10.1016/j.jnca.2020.102579.

27. Brown, S., Williams, M., Jackson, T. Advanced Middleware for High-Speed Data Communication in Multicomputer Systems. *IEEE Transactions on Computers*. 2021. №99. P. 12-13. DOI: 10.1109/TC.2021.3134456.

28. imon, J., Olivier, M. Secure Middleware Solutions for IoT Networks with Ring Topology. *Computer Networks*. 2021. №69. P. 34-42. DOI: 10.1016/j.comnet.2021.108555.

29.Gomez, F., Santos, R., Perez, P. Middleware Framework for Efficient Resource Management in Distributed Cloud Environments. *Future Generation Computer Systems*. 2020. №117. P. 14-16. DOI: 10.1016/j.future.2020.102567.

30.Zhang, W., Li, M., Hu, C. Middleware for Low-Latency Communication in Edge Computing Systems. *IEEE Internet of Things Journal*. 2022. №46. P. 34-41. DOI: 10.1109/JIOT.2022.3161245.

31. Anderson, J., Woods, P., Clark, D. Middleware Solutions for Fault Tolerance in Distributed Systems. *Journal of Parallel and Distributed Computing*. 2023. №42. P. 33-40. DOI: 10.1016/j.jpdc.2023.102456.

32.Rivera, M., Carreras, X., Morales, J. Middleware for Adaptive Network Configuration in Smart *Grid Systems. Energies*. 2021. №35. P. 17-18. DOI: 10.3390/en14051234.

33. Gonzalez, P., Rodriguez, L., Sanchez, S. Middleware Solutions for Real-Time Data Synchronization in Distributed Databases. *Data & Knowledge Engineering*. 2022. №26. P. 18-30. DOI: 10.1016/j.datak.2022.101456.

34.Sundari, G., Kishon, S., Chen, D. Middleware for Efficient Communication in Wireless *Sensor Networks with Ring Topology. Sensors*. 2022. №16. P. 11-15. DOI: 10.3390/s220203456.

35.Бхуюн П. Програмування паралельних обчислень. – Нью-Йорк: Спрінгер, 2020. 600 с.

36.Lewis, K., Miller, R., Thompson, J. Middleware for Enhanced Data Security in Cloud-Based *Systems. Journal of Cloud Computing*. 2023. №9. P. 28-30. DOI: 10.1016/j.cloud.2023.103456.

37. Martinez, K., Lopez, A., Rodriguez, M. Middleware for Efficient Communication in Distributed Sensor Networks. *Journal of Computer and System Sciences*. 2022. №61. P. 37-40. DOI: 10.1016/j.jcss.2022.104567.

38.Park, J., Lee, H., Kim, M. Middleware Solutions for Energy-Efficient Communication in Wireless Networks. *IEEE Transactions on Green Communications and Networking*. 2020. №56. P. 33-35. DOI: 10.1109/TGCN.2020.2964567.

39 Jones, P., Brown, R., Taylor, K. Middleware for Scalable Data Processing in Big Data Environments. *Journal of Big Data*. 2021. №86. P. 41-42. DOI: 10.1186/s40537-021-00456-7.

40.Gomez, L., Santos, P., Rodriguez, F. Middleware for Efficient Data Management in Distributed IoT *Systems. Internet of Things*. 2023. №63. P. 11-12. DOI: 10.1016/j.iot.2023.100345.

41. Li, H., Chen, W., Zhang, S. Middleware for Real-Time Data Processing in Smart Factory Systems. *IEEE Transactions on Industrial Electronics*. 2022. №21. P. 15-19. DOI: 10.1109/TIE.2022.3145678.

42. Huang, M., Lin, S., Zhang, P. Middleware for Reliable Communication in Autonomous Vehicle Networks. *IEEE Transactions on Vehicular Technology*. 2021. №5. P. 20-22. DOI: 10.1109/TVT.2021.3054567.

43. Martinez, P., Lopez, A., Rodriguez, K. Middleware for Dynamic Network Configuration in Distributed Systems. *Journal of Network and Computer Applications*. 2020. №65. P. 24-25. DOI: 10.1016/j.jnca.2020.102579.

44. Foster, R., Gomez, M., Rodriguez, P. Middleware for Secure Communication in Blockchain Networks. *Future Generation Computer Systems*. 2021. №23. P. 34-35. DOI: 10.1016/j.future.2021.104567.

45. Lin, M., Chen, Y., Hu, W. Middleware for Adaptive Data Transmission in Smart Grid Networks. *IEEE Transactions on Smart Grid*. 2022. №54. P. 5-12. DOI: 10.1109/TSG.2022.3165678.

46.Carter, S., Jones, W., Brown, R. Middleware for High-Performance Computing in Distributed Systems. *Journal of Parallel and Distributed Computing*. 2023. №12. P. 51-55. DOI: 10.1016/j.jpdc.2023.102345.

47. Lian, K., Garan, O., Li, M. Middleware for Efficient Resource Allocation in Distributed Cloud Systems. *Journal of Cloud Computing*. 2023. №132. P. 22-31. DOI: 10.1016/j.cloud.2023.103345.

48.Zhang, S., Li, J., Chen, H.Middleware for Real-Time Communication in Autonomous Vehicle Network. *IEEE Transactions on Intelligent Transportation Systems*. 2022. №142. P. 12-18. DOI: 10.1109/TITS.2022.3145678.

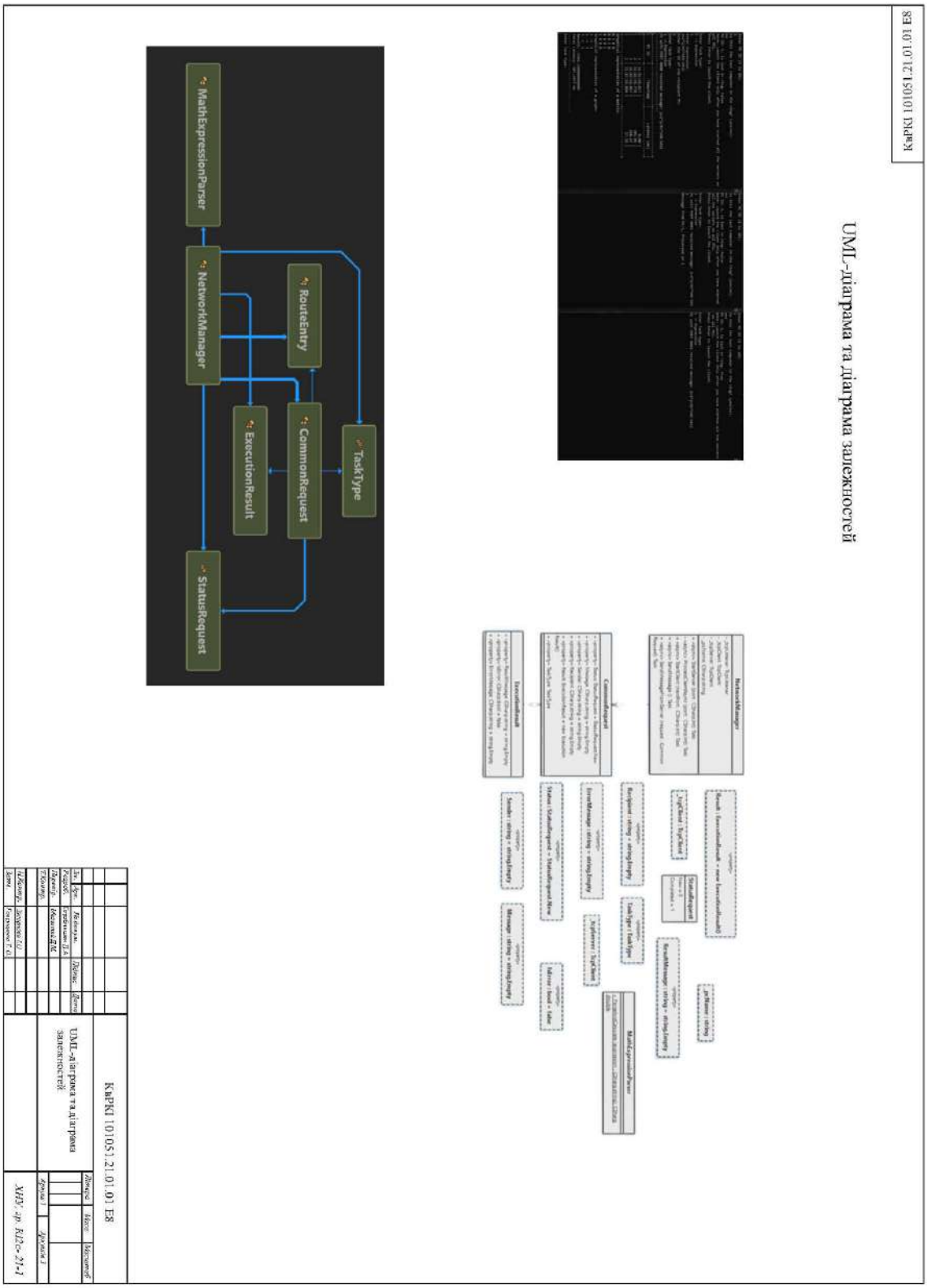
49.Lin, C., Chang, W., Chen, M. Middleware for Efficient Data Processing in Smart Healthcare Systems. *Journal of Medical Systems*. 2021. №15. P. 24-28. DOI: 10.1007/s10916-021-00567-8.

50.Fiberoptic Communication System Architectures And Topologies. Fiberopticx. 2024. URL: https://www.fiberopticx.com (дата звернення: 04.04.2024).

Додаток А

(обов’язковий)

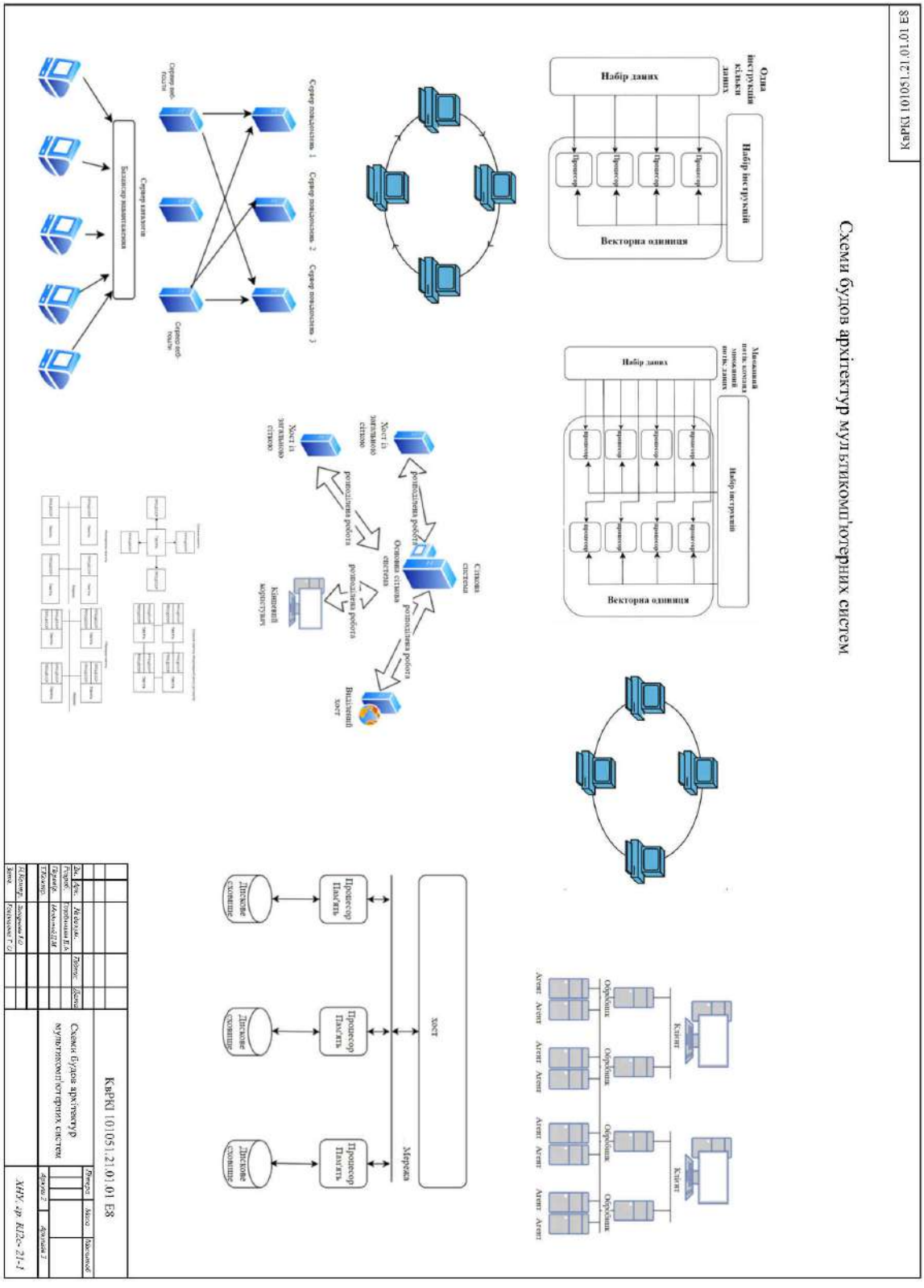
Копія креслення «UML-діаграма та діаграма залежностей»



Додаток Б

(обов’язковий)

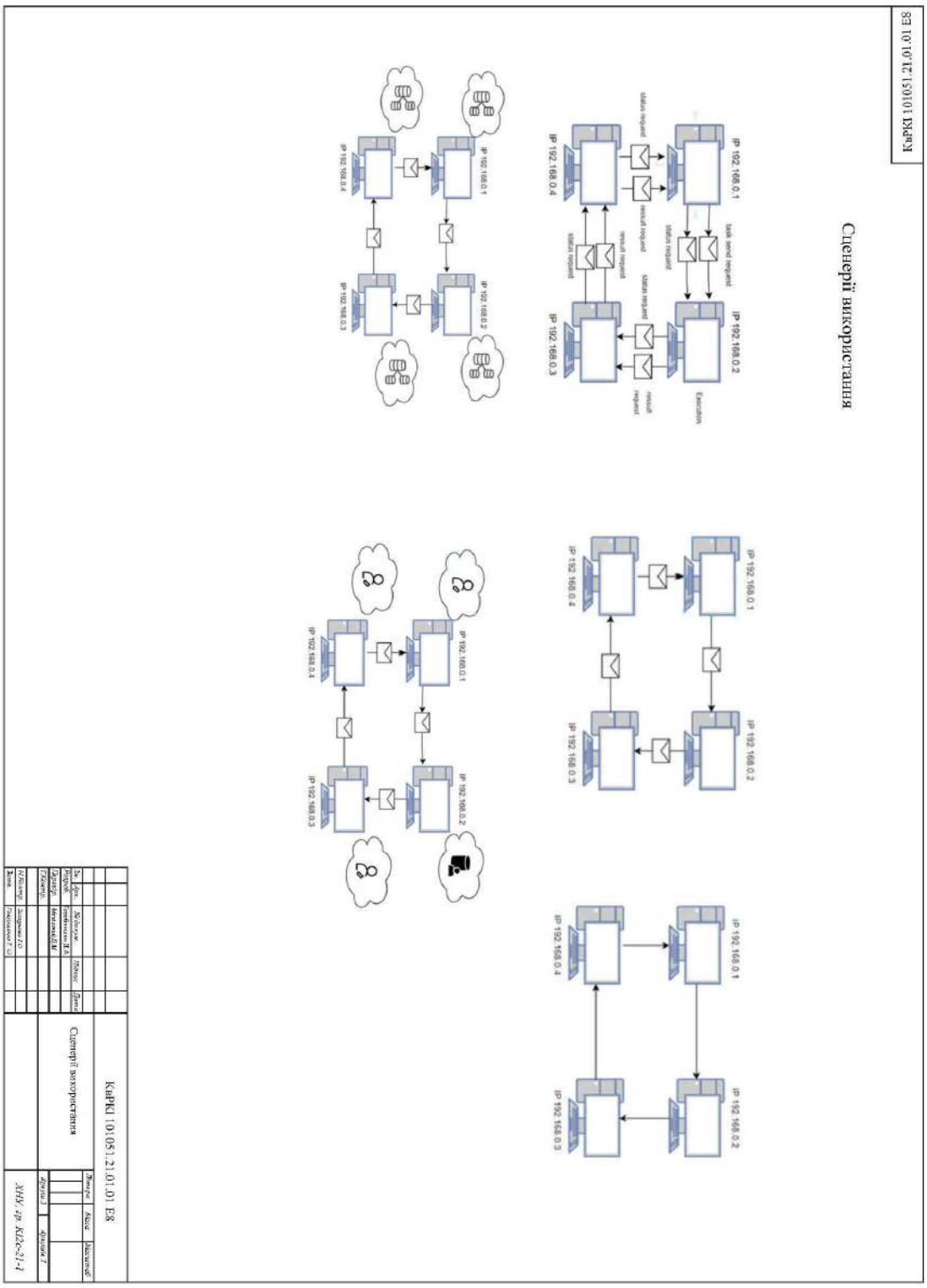
Копія креслення «Схеми будов архітектур мультикомп’ютерних систем»



Додаток В

(обов'язковий)

Копія креслення «Сценарії використання»



UNICHECK
by Turnitin

Ім'я користувача:
Кафедра КІ

ID перевірки:
1016326024

Дата перевірки:
06.06.2024 05:34:52 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
06.06.2024 17:41:14 EEST

ID користувача:
100005591

Назва документа: **Горобчишен_Проміжне програмне забзпечення мультикомп'ютерної системи з топологією...**

Кількість сторінок: **81** Кількість слів: **13029** Кількість символів: **107926** Розмір файлу: **3.18 MB** ID файлу: **1016124915**

5.21% Схожість

Найбільша схожість: **3.16% з джерелом з Бібліотеки** (ID файлу: **1015048395**)

2.53% Джерела з Інтернету	122	Сторінка 83
4.09% Джерела з Бібліотеки	145	Сторінка 84

0% Цитат

Не знайдено жодних цитат

Посилання	1	Сторінка 84

0% Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	99

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 8%**

ID: 128877 Назва: БКР Проміжне програмне забзпечення мультикомп'ютерної системи з топологією "кільце" однонаправленого зв'язку Додано в БД: 2024-06-06 Автора: Горобчишен Д.А. Керівники: Савнко О.С. Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	94153	748	1942 (2%)	25 (3%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Горобчишен Денис Анатойович

Тема: Проміжне програмне забзпечення мультикомп'ютерної системи з топологією "кільце" однонаправленого зв'язку

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3____ Кількість сторінок записки ___72___

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є створення програмного забезпечення для мультикомп'ютерної системи з топологією "кільце" однонаправленого зв'язку.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: В першому розділі кваліфікаційної роботи проведено дослідження предметної області (проаналізовано існуючі рішення та проведено порівняльний аналіз) та виконано постановку задачі дослідження. В другому розділі кваліфікаційної роботи проведено аналіз та вибір мови програмавання та апаратних пристоїв. В третьому розділі кваліфікаційної роботи виконано апаратну та програмну реалізацію мулькомп'ютерно системи типу кільце однонаправленого типу, а саме реалізовано підключення кінцевих пристроїв та імплементовано програмне забезпечення для робити з ними.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: відсутність графічного інтерфейсу.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: добре

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

Бодрящий Леонід Петрович, зав. каф. ІПЗ

_____

"___"_____2024 р. _____(підпис)

Завідувачу кафедри КІІС
д-р.техн.наук, проф. Говорущенко Т. О.

Горобчишен Денис Анатолійович
ПІБ здобувача вищої освіти

ФІТ, 3 курсу, групи КІ2с-21-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

4 червня 2024 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Проміжне програмне забзпечення мультикомп'ютерної системи з топологією "кільце" однонаправленого зв'язку

Автор: Горобчишен Денис Анатолійович

Спеціальність: 123– Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Медзатий Дмитро Миколайович, к.т.н, доцент

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) окремі виявлені збіги є загальновживаними фразами або виразами.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 5.21% і адресується до 267 першоджерела, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Д.М. Медзатий

Гарант ОП _____ С.М. Лисенко

Завідувач кафедри КІС _____ Т. О. Говорущенко