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

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Мультикомп'ютерна система з топологією «Квітка»
Назва теми

КвРКІ 200230.20.02.06 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 IV курсу, група КІ2-20-2 _____ Ю. Ю. Головатий
Підпис Ініціали, прізвище

Керівник _____ К. М. Березька
Підпис, дата Ініціали, прізвище

Нормоконтролер _____ С.М. Лисенко
Підпис, дата Ініціали, прізвище

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

«24» 06 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

Головатому Юрію Юрійовичу

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Мультикомп'ютерна система з топологією "квітка"

Керівник проекту (роботи) Березька К.М., д.т.н., проф.

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 01.03.2024 р. № 5

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)

Аналіз предметної області та постановка задачі

Проєктування проміжного програмного забезпечення мультикомп'ютерної системи з топологією квітка

Програмно-апаратна реалізація проміжного програмного забезпечення мультикомп'ютерної системи з топологією квітка

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

Топологія мереж

Алгоритм роботи програми

Оцінка результатів ефективності програми

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Лисенко С.М. професор кафедри КПС		
Антиплагіат	Нічепорук А.О., доцент кафедри КПС		

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 – проєктування мультикомп'ютерної системи з топологією «квітка», її архітектура	01.04.2024	виконано
5	Робота над розділом 3 – програмна реалізація проміжного програмного забезпечення мультикомп'ютерної системи з топологією квітка	29.04.2024	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2024	виконано
7	Попередній захист ВКР	26.05.2024	виконано
8	Захист ВКР на засіданні ЕК	Червень 2024 року	

Студент _____ Ю. Ю. Головатий
Підпис Ініціали, прізвище

Керівник роботи _____ К. М. Березька
Підпис Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 200230.20.02.06 ПЗ	Пояснювальна записка	57		
			Графічні матеріали			
2		КвРКІ 200230.20.02.06 Е8	Топології мереж	1		
3		КвРКІ 200230.20.02.06 Е8	Блок-схема алгоритму	1		
			роботи програми			
4		КвРКІ 200230.20.02.06 Е8	Оцінка результатів	1		
			ефективності програми			

КвРКІ 200230.20.02.06 ВП

Зм	Арк	№ докум	Підпис	Дата
Розробив		Головатий		
Перевір.		Березька		24.06
Н. контр.		Лисенко		
Затв.		Говорущенко		24.06

Відомість проекту

Літера	Аркуш	Аркушів
У	1	1

ХНУ, КІ2-20-2

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Мультикомп'ютерна система з топологією квітка».

Автор роботи: Головатий Юрій.

Керівник роботи: Березька Катерина Миколаївна.

Пояснювальна записка: 57 с., 33 рис., 1 табл., 3 дод., 45 джерел.

Графічна частина: 3 креслення.

МУЛЬТИКОМП'ЮТЕРНА СИСТЕМА, ТОПОЛОГІЯ.

Метою дипломної роботи є розроблено програмне забезпечення, що працює відповідно до топології "квітка" в мультикомп'ютерній системі.

Об'єктом дослідження є є розроблено програмне забезпечення, що працює відповідно до топології "квітка" в мультикомп'ютерній системі.

Предметом дослідження є топології "квітка" в мультикомп'ютерній системі.

Під час проведення даного дослідження був використаний метод систематичного огляду літератури для вивчення і аналізу предметної області даного дослідження з текстових джерел інформації.

Підпис студента

30.05.2024

Дата

ЗМІСТ

ВСТУП

Мультикомп'ютерні системи є складними обчислювальними структурами, які базуються на спільному використанні ресурсів різних комп'ютерів для вирішення великих обчислювальних завдань. Цей підхід виник з необхідності розв'язання складних проблем, таких як обробка великих обсягів даних або моделювання складних систем, які вимагають великої обчислювальної потужності.

Наукове дослідження у цій області включає в себе розробку ефективних алгоритмів розподіленого обчислення, розробку систем управління ресурсами та механізмів комунікації між вузлами, а також вивчення питань безпеки та надійності в мультикомп'ютерних середовищах.

Важливою характеристикою мультикомп'ютерних систем є їх масштабованість. Вони можуть включати в себе велику кількість вузлів, які можуть бути фізично розташовані в різних місцях і підключені до мережі. Це робить їх ідеальними для виконання завдань, які вимагають великої кількості обчислювальних ресурсів або великої кількості даних.

У сучасному світі інформаційних технологій мережеві топології відіграють ключову роль у забезпеченні ефективної і надійної комунікації між комп'ютерами та іншими пристроями. Топології "зірка" та "квітка" є двома з найпоширеніших і найвідоміших структур, що використовуються для побудови мереж різного масштабу і призначення. Вони забезпечують фундамент для організації передачі даних, визначаючи спосіб з'єднання вузлів та управління трафіком в мережі.

Топологія "зірка" є однією з найпростіших і водночас найбільш надійних форм організації мереж. Вона характеризується центральним вузлом, до якого підключаються всі інші вузли, що дозволяє легко додавати нові пристрої і швидко виявляти несправності. Ця топологія широко використовується в корпоративних мережах і домашніх середовищах завдяки своїй простоті та ефективності.

Топологія "квітка", також відома як гібридна топологія, базується на принципах топології "зірка", але доповнена додатковими з'єднаннями між периферійними вузлами. Це забезпечує підвищену надійність і стійкість до збоїв, оскільки трафік може перенаправлятися через альтернативні шляхи у разі відмови окремих з'єднань. Така структура є особливо корисною в критичних мережах, де безперебійна робота і висока надійність є пріоритетом.

Дослідження та аналіз цих двох топологій є важливими для розуміння їх переваг і недоліків, а також для вибору найбільш підходящої структури для конкретних мережевих потреб. У даній роботі буде розглянуто основні характеристики топологій "зірка" та "квітка", їхні принципи роботи, а також буде обговорено приклади практичного застосування і оптимальні сценарії використання.

1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області та її особливостей

Мультикомп’ютерні системи - це системи комп’ютерів, де декілька пристроїв об’єднані між собою для виконання спільної роботи. Для виконання роботи слід забезпечити між всіма вузлами системи достатньо швидкий та надійний зв’язок. Проведемо аналіз предметної області і зокрема розглянемо використання мультикомп’ютерних систем з топологією “квітка”.

Існують такі типи зв’язку, як фізичний та логічний. Фізичний зв’язок - це тип зв'язку, який використовується для передачі даних між вузлами на фізичному рівні. Він включає в себе такі способи передачі, як:

- Мідні кабелі.
- Оптичні кабелі.
- Бездротовий зв’язок.

Також окремо можна виділити інфрачервоний зв’язок та провідне з’єднання (USB-кабель, тощо).

До мідних кабелів відносяться такі приклади, як вита пара та коаксіальний кабель. Такий спосіб використовується для телефонних ліній, мереж Ethernet та інших мереж даних. Вони дешеві, та прості в установці, але мають обмежену пропускну здатність. Коаксіальний кабель використовується для телебачення, радіо та кабельного Інтернету.

Оптичні кабелі бувають одномодовими або багатомодовими. Одномодові використовуються для передачі даних на великі відстані, наприклад підводні кабелі. Багатомодові частіше використовуються для коротких зв’язків наприклад в локальних мережах.

Прикладами бездротового зв’язку Wifi, Bluetooth та сотовий зв’язок, які ми використовуємо у повсякденні, наприклад у офісі, вдома, або для мобільного інтернету та телефонних дзвінків.

Логічний зв’язок – це тип зв'язку, який використовується для передачі даних між вузлами на програмному рівні. До логічного зв’язку відносяться комутація каналів і комутація пакетів. Комутація каналів використовує виділений канал зв'язку для передачі даних між двома вузлами, а комутація пакетів - це метод розбиває дані на пакети, які передаються по мережі незалежно один від одного. Саме метод комутації пакетів найчастіше використовується для передачі даних в мультикомп’ютерних системах.

Мультикомп'ютерні системи (МКС) використовуються в широкому спектрі задач, де потрібна висока продуктивність, масштабованість та доступність. Їх використовують в наукових дослідженнях, інженерному проектуванні, фінансових послугах, медицині та розвагах. Вони виконують такі функції, як моделювання складних систем, таких як клімат Землі або людський геном, проектування та моделювання продуктів, таких як літаки, автомобілі та будівлі, аналіз даних, прогнозування ринкових тенденцій та обробки торгів, розробка нових ліків та моделювання біологічних процесів, створення спецефектів, рендерингу відео та трансляції контенту.

На сьогодні відомо багато видів топологій мультикомп’ютерних систем, таких як: центрелізована, магістральна, кільце, зірка, сітка, гібридна. Кожна з них має свої переваги та недоліки. В нашому випадку, відповідно до завдання детальніше розглядатиметься саме топологія “квітка”.

Топологія «квітка» — це тип мережевої топології, в якій усі пристрої підключені до центрального комутатора. Центральний комутатор відповідає за маршрутизацію трафіку між пристроями.

Топології «квітка» прості в налаштуванні та управлінні. Їх також легко масштабувати, додаючи більше пристроїв до центрального комутатора. Однак топології «квітка» можуть бути дорогими, оскільки їм потрібен центральний

комутатор із великою кількістю портів. Крім того, якщо центральний комутатор вийде з ладу, вся мережа вийде з ладу.

Мультикомп'ютерні системи з топологією "Квітка" використовуються в різних сферах, де важливі простота, надійність та масштабованість. Ось декілька прикладів:

1) Локальні мережі (LAN). МКС з топологією "Квітка" може використовуватися для з'єднання комп'ютерів в невеликій локальній мережі.

Переваги: простота налаштування та обслуговування, доступність даних.

Недоліки: може бути не оптимальним для великих мереж, центральний вузол може стати точкою відмови.

2) Системи домашньої автоматизації. МКС з топологією "Квітка" може використовуватися для з'єднання пристроїв домашньої автоматизації, таких як розумні лампочки, термостати та датчики.

Переваги: простота, надійність, гнучкість.

Недоліки: може бути не оптимальним для складних систем з великою кількістю пристроїв.

3) Системи моніторингу. МКС з топологією "Квітка" може використовуватися для з'єднання датчиків в системі моніторингу, наприклад, для моніторингу температури, тиску, або вологості.

Переваги: простота, надійність, масштабованість.

Недоліки: може бути не оптимальним для систем, які потребують високої пропускної здатності.

4) Кластерні системи. МКС з топологією "Квітка" може використовуватися для з'єднання вузлів кластера, де простота та надійність важливіші, ніж максимальна продуктивність.

Переваги: простота, надійність, доступність даних.

Недоліки: може бути не оптимальним для кластерних систем, які потребують високої продуктивності.

5) Розподілені системи. МКС з топологією "Квітка" може використовуватися для з'єднання серверів в розподіленій системі, де не потрібна висока продуктивність.

Переваги: простота, надійність, масштабованість.

Недоліки: може бути не оптимальним для розподілених систем, які потребують високої продуктивності.

1.2.Порівняльний аналіз переваг та недоліків сучасних мультикомп’ютерних систем з топологією “Квітка”

Для проведення аналізу переваг та недоліків існуючих рішень спершу слід визначити критерії, згідно яких буде відбуватись оцінка тієї чи іншої МКС.

Цінним ресурсом є пропускна здатність мережі - максимальна швидкість, з якою дані можуть передаватися каналом зв’язку. Вона вимірюється в бітах за секунду (біт/с).

Її важливо застосовувати ефективно. Тому вона стане важливим параметром оцінки мультикомп’ютерної системи.

Також для оцінки швидкодії мережі варто враховувати затримку. Затримка, також відома як час очікування або лаг, – це час, який потрібен для передачі сигналу або даних від джерела до приймача. Вона вимірюється в мілісекундах (мс) або мікросекундах (мкс).

Затримка може виникати з кількох причин, таких як:

- Відстань. Чим більша відстань, яку має пройти сигнал, тим більша затримка.
- Обробка. Час, необхідний для обробки сигналу маршрутизаторами, комутаторами та іншими мережевими пристроями.
- Навантаження. Затримка може збільшуватися, коли мережа перевантажена.

Масштабованість – це складне питання, але воно є важливим для багатьох систем та програм.

Масштабованість являє собою здатність системи працювати з більшою кількістю вузлів.

За допомогою правильних технологій та методів можна покращити масштабованість, щоб система або програма могла ефективно працювати з більшим навантаженням.

І останніми не менш важливими критеріями є вартість і складність системи. Вартість включає в себе не лише ціну покупки системи, а також витрати на встановлення, експлуатацію та утримування системи.

Складність включає в себе технічну складність, складність використання а також складність навчання користувачів, що цю технологію використовують. Система або технологія, яка є найдешевшою або найпростішою, може не мати всіх необхідних функцій.

З іншого боку, система або технологія, яка має всі необхідні функції, може бути занадто дорогою або складною.

Високопродуктивна мережа взаємозв’язків є ключем до реалізації високошвидкісних спільних паралельних обчислень на кожному вузлі високопродуктивної комп’ютерної системи.

З безперервним покращенням продуктивності високопродуктивних комп’ютерних систем тенденція розвитку високопродуктивних мереж взаємозв’язку в основному відображається на розмірах мережі та її пропускній здатності. З уповільненням закону Мура необхідно прийняти нові технології дизайну упаковки для впровадження високопродуктивних мереж з’єднання для високопродуктивних обчислень.

А також розробляється дизайн репрезентативних, найсучасніших високопродуктивних мереж взаємозв’язку, зокрема NVIDIA InfiniBand, Intel Omni. -Path, Cray Slingshot/Aries та користувацькі або власні мережі, включаючи Fugaku Tofu, Bull BXI, TH Express тощо.

Розмовляючи про мультикомп’ютерні системи варто відмітити деякі приклади використання даної технології в сучасності.

Прозглянемо кластерні системи.

1.1.1 Google Kubernetes Engine:

Ця система використовується Google для розгортання та керування контейнеризованими додатками.

1.1.2 Amazon Elastic Kubernetes Service (EKS)

Amazon Elastic Kubernetes Service (EKS) - це система, яка використовується Amazon Web Services (AWS) для розгортання та керування контейнеризованими додатками.

1.1.3 Microsoft Azure Kubernetes Service (AKS)

Microsoft Azure Kubernetes Service (AKS) – це система, яка використовується Microsoft Azure для розгортання та керування контейнеризованими додатками.

Хмарні обчислення. Розглянемо Amazon Web Services (AWS). Ця платформа хмарних обчислень пропонує широкий спектр послуг, таких як обчислення, зберігання, мережі та бази даних.

Розглянемо Microsoft Azure. Ця платформа хмарних обчислень пропонує широкий спектр послуг, таких як обчислення, зберігання, мережі та бази даних.

Розглянемо Google Cloud Platform (GCP). Ця платформа хмарних обчислень пропонує широкий спектр послуг, таких як обчислення, зберігання, мережі та бази даних.

Розглянемо сітки даних:

- Hadoop. Ця платформа з відкритим кодом використовується для зберігання та аналізу великих обсягів даних.
- Spark. Ця платформа з відкритим кодом використовується для обробки даних на кластерах комп'ютерів.
- Apache Hive. Ця платформа з відкритим кодом використовується для запитів до даних, що зберігаються в Hadoop.

Системи з високою продуктивністю (HPC):

- Summit. Цей суперкомп'ютер, розташований в Оук-Ридж, штат Теннессі, США, є найпотужнішим комп'ютером у світі.
- Sierra. Цей суперкомп'ютер, розташований в Лоуренс-Ліверморській національній лабораторії, штат Каліфорнія, США, є другим найпотужнішим комп'ютером у світі.
- Fugaku. Цей суперкомп'ютер, розташований в Кобе, Японія, є третім найпотужнішим комп'ютером у світі.

Оскільки робота ведеться саме з МКС з топологією “квітка”, то розглянемо декілька прикладів сучасних мультикомп’ютерних систем з даною топологією.

Infiniband - це високопродуктивна мережа, що використовується для з'єднання комп'ютерів в кластерах та суперкомп'ютерах. Вона використовує топологію "квітка", де центральний комутатор з'єднується з усіма іншими вузлами. Infiniband може забезпечити дуже високу пропускну здатність та низьку затримку, що робить її ідеальною для HPC-додатків. Серцем будь-якого сучасного суперкомп’ютера є мережеве з’єднання з низькою затримкою і високою пропускною здатністю, яке створює єдину систему з набору вузлів. На даний момент Infiniband є основним комерційно доступним типом інтерконнекту без жодної реальної конкуренції в усьому світі.

Розглянемо переваги Infiniband. Висока пропускна здатність (Infiniband може забезпечити пропускну здатність до 100 Гбіт/с на канал). Низька затримка (Infiniband може забезпечити затримку менше 1 мікросекунди). Масштабованість

(Infiniband може масштабуватися до тисяч вузлів). А також Infiniband має високу надійність і ефективність.

Розглянемо недоліки Infiniband. Вартість (Infiniband може бути дорогим порівняно з іншими мережами). Складність (Infiniband може бути складним у налаштуванні та управлінні).

Myrinet - це високопродуктивна комунікаційна мережа, розроблена компанією Myricom для з'єднання комп'ютерів у кластерах та суперкомп'ютерах. Вона використовує топологію "квітка", де центральний комутатор з'єднаний з усіма іншими вузлами.

Хоча Myrinet зараз не так широко використовується, як раніше, він все ще має певні переваги та особливості, які варто розглянути:

Розглянемо переваги Myrinet. Висока пропускна здатність: Myrinet пропонує декілька рівнів пропускної здатності, доходячи до 20 Гбіт/с на канал. Це робить його хорошим вибором для додатків, які потребують швидкого обміну даними. Затримка Myrinet також відносно низька, менше 2 мікросекунд на канал. Це важливо для додатків, які потребують швидкої передачі даних в режимі реального часу. Масштабованість (Myrinet можна масштабувати до сотень вузлів, що робить його придатним для великих кластерів та суперкомп'ютерів). Надійність: Myrinet має високу надійність та доступність.

Розглянемо ндоліки Myrinet. Вартість (Myrinet може бути дорогим порівняно з іншими мережами, такими як Infiniband). Складність (Myrinet може бути складним у налаштуванні та управлінні). Підтримка: Компанія Myricom була придбана в 2013 році, а підтримка продукту зараз обмежена.

Quadrics - це високопродуктивна мережа, але зараз вона не така популярна, як раніше. Її переваги у вигляді високої пропускної здатності, низької затримки та гнучкості можуть мати значення для певних випадків. Однак вартість, складність і обмежена підтримка роблять її менш привабливим вибором порівняно з більш сучасними рішеннями.

Розглянемо переваги Quadrics. Висока пропускна здатність (до 16 Гбіт/с на канал). Низька затримка (менше 3 мікросекунд на канал). Масштабованість (Quadrics масштабується до сотень вузлів). Гнучкість - Quadrics може бути налаштований для використання з різними типами процесорів. Ефективність енергоспоживання: Quadrics відомий своїм низьким енергоспоживанням.

Розглянемо недоліки Quadrics.

Вартість: Quadrics може бути дорожчим, ніж деякі інші мережі з топологією "квітка". Складність: Quadrics може бути складним у налаштуванні та управлінні. Підтримка: Більше не випускаються нові апаратні продукти Quadrics, а підтримка існуючих систем може бути обмежена.

SCI Express (Scalable Coherent Interface Express) - це високопродуктивна мережа, розроблена компанією Mellanox для з'єднання комп'ютерів у кластерах та суперкомп'ютерах. Вона використовує топологію "квітка", де центральний комутатор з'єднується з усіма іншими вузлами.

Хоча зараз SCI Express не так широко використовується, він все ж має певні особливості, які можуть бути корисними в певних ситуаціях.

Розглянемо переваги SCI Express.

Висока пропускна здатність (до 12 Гбіт/с на канал). Затримка SCI Express досить низька, менше 4 мікросекунд на канал. SCI Express масштабується до сотень вузлів. Інтеграція з PCIe: SCI Express тісно інтегрується з шиною PCIe, що може знизити затримку та складність. Підтримка від галузевих лідерів: SCI Express підтримується багатьма виробниками серверів та кластерних систем.

Розглянемо недоліки SCI Express. SCI Express як і Quadrics може бути дорожчим, ніж деякі інші мережі з топологією "квітка". Складність: SCI Express може бути складним у налаштуванні та управлінні. SCI Express більше не є основним продуктом Mellanox, хоча і залишається актуальним для певних застосувань.

Ethernet – це найвідоміша технологія мережі, широко використовувана у всьому, від дротових домашніх мереж до складних суперкомп'ютерних кластерів.

Хоча він не спеціалізується на топології "квітка", Ethernet може використовуватися та адаптуватися для підтримки цієї конфігурації.

Коли в кластері найбільше потрібні висока пропускна здатність і використання інфраструктури на рівні ланки, Ethernet є потенційним кандидатом, конкуруючи з традиційними мережевими з'єднаннями HPC. Розподілений збір даних у реальному часі під час експериментів з фізики елементарних частинок представляє цікавий варіант використання. Оцінюються можливі рішення на основі Ethernet для агрегування даних із сотень джерел із пропускною здатністю десятків Тбіт/с. Це веде нас до обміну даними «багато-до-одного», де ми прагнемо створити економічно оптимізовану установку, яка підтримує понад 80 % теоретичного навантаження на канал. Ми досліджуємо можливі моделі трафіку на основі Ethernet для обробки збору даних на великих пристроях із кількома джерелами. У великомасштабній мережі допускається різна кількість виробників і приймачів, а також різні швидкості з'єднання. Випробування продуктивності проводилися з використанням налаштованих тестів і тестових стендів для оцінювання.

Розглянемо переваги Ethernet. Низька вартість, Ethereum є однією з найдешевших мережевих технологій, доступних на ринку. Ethernet порівняно легкий для налаштування та управління, навіть для людей без значних технічних знань.

Широка поширеність Ethernet є стандартом на ринку, підтримується широким спектром пристроїв і виробників. Ethernet можна масштабувати до великих кластерів із сотнями вузлів за допомогою комутаторів та інших технологій.

Розглянемо Недоліки Ethernet. Пропускна здатність: Пропускна здатність Ethernet нижча, ніж у спеціалізованих технологій МКС, як-от Infiniband, досягаючи максимум 100 Гбіт/с на канал.Затримка Ethernet також вища, ніж у спеціалізованих МКС, приблизно 5 мікросекунд на канал. Основна топологія

Ethernet – шина або зірка, а не квітка. Розгортання "квітки" вимагає додаткових компонентів та конфігурації.

Застосування Ethernet в МКС. Ethernet іноді використовується в кластерах для локальних з'єднань між вузлами та комутаторами.

Він може бути частиною гібридної мережі, де високопродуктивні технології МКС з'єднують кластери, а Ethernet використовується для внутрішніх підключень.

Деякі суперкомп'ютери використовують Ethernet у спеціальних конфігураціях для досягнення топології "квітка", але з нижчою продуктивністю, ніж спеціалізовані рішення. Для більшої зручності описані характеристики систем наведено нижче в таблиці 1.1.

Таблиця 1.1 - Характеристики мультикомп’ютерних систем з топологією “квітка”

Система	Пропускна здатність	Затримка	Масштабованість	Вартість	Складність	Застосування
Infiniband	100 Гбіт/с на канал	< 1 мкс	До тисяч вузлів	Висока	Складна	Суперкомп'ютери, кластери, хмарні обчислення, зберігання даних
Myrinet	20 Гбіт/с на канал	< 2 мкс	До сотень вузлів	Середня	Складна	Суперкомп'ютери, кластери
Quadrics	16 Гбіт/с на канал	< 3 мкс	До сотень вузлів	Середня	Складна	Суперкомп'ютери, кластери
SCI Express	12 Гбіт/с на канал	< 4 мкс	До сотень вузлів	Середня	Складна	Суперкомп'ютери, кластери
Ethernet	10 Гбіт/с на канал	< 5 мкс	До тисяч вузлів	Низька	Проста	Локальні мережі

Як бачимо кожна МКС має свої переваги і недоліки, і як показує практика більшої поширеності набувають саме системи, які прості в налаштуванні та використанні, а також відносно дешеві за своєю вартістю.

Про те для більш складних робіт використовуються більш дорогі, більш складні МКС з високою пропускною здатністю. В будь-якому випадку для вибору потрібної нам системи характеристики будуть обиратись відповідно до потреб..

1.3 Постановка задачі та висновки.

Враховуючи матеріали, наведені в першому розділі, можна зробити висновок, що мультикомп’ютерні системи сьогодення мають надзвичайно широкий спектр застосування і характеризуються великою варіативністю. Це пов’язано з їхньою здатністю забезпечувати високу продуктивність, гнучкість та надійність у різних галузях, таких як наука, бізнес, індустрія розваг і багато інших. Мультикомп’ютерні системи дозволяють ефективно використовувати ресурси, розподіляти навантаження і забезпечувати безперебійну роботу навіть у складних умовах.

Особливої уваги заслуговують топології "квітка" та "зірка", які надають можливість оптимального з'єднання вузлів для забезпечення високої надійності та продуктивності мережі.

Завданням даного дослідження буде проектування, розробка та тестування проміжного програмного забезпечення для мультикомп’ютерної системи з топологією “квітка”.

Ця топологія, завдяки своїй гібридній природі, поєднує переваги зіркової топології з додатковими можливостями для забезпечення резервування та відмовостійкості.

Вона дозволяє створити більш гнучку і надійну мережу, здатну витримувати збої окремих вузлів або з'єднань без значного впливу на загальну продуктивність системи.

Проектування проміжного програмного забезпечення для такої мережі включає в себе розробку алгоритмів для ефективного управління ресурсами, обробки запитів і передачі даних між вузлами. Особливу увагу буде приділено забезпеченню надійності і безпеки передачі даних, а також оптимізації продуктивності мережі. Тестування програмного забезпечення дозволить оцінити його ефективність у реальних умовах експлуатації, виявити можливі недоліки і внести необхідні корективи для їх усунення.

Таким чином, дане дослідження спрямоване на створення високоефективного та надійного проміжного програмного забезпечення для мультикомп’ютерної системи з топологією "квітка", що дозволить максимально використати переваги цієї топології і забезпечити стабільну роботу мережі в умовах різноманітних навантажень і можливих збоїв.

2 ПРОЄКТУВАННЯ МУЛЬТИКОМП’ЮТЕРНОЇ СИСТЕМИ З ТОПОЛОГІЄЮ «КВІТКА», ЇЇ АРХІТЕКТУРА

2.1 Архітектура в мультикомп’ютерній системі

Архітектура в контексті обчислювальних систем відноситься до структурної організації та конструкції системи, включаючи її компоненти, взаємозв'язки між ними та принципи, за якими вони працюють разом для виконання завдань.

Це широке поняття, яке охоплює апаратні засоби, програмне забезпечення, мережеві компоненти та взаємодії між цими елементами.

Апаратна архітектура включає фізичні компоненти комп'ютера або комп'ютерної системи, такі як процесори, пам'ять, периферійні пристрої, шини та інші апаратні засоби. Апаратна архітектура визначає, як ці компоненти взаємодіють між собою, їхні характеристики та спосіб з'єднання.

Програмна архітектура включає структуру програмного забезпечення, яка включає компоненти, модулі, бібліотеки та інші програмні одиниці. Вона визначає, як програмні компоненти взаємодіють один з одним і з апаратними засобами, а також як вони організовані для виконання завдань.

Мережна архітектура включає компоненти і принципи організації мереж, які забезпечують зв'язок між комп'ютерами або вузлами. Це може включати фізичні топології, мережеві протоколи, маршрутизатори, комутатори та інші мережеві пристрої.

Архітектура даних описує структуру, організацію, зберігання, управління і доступ до даних у системі. Це включає бази даних, сховища даних, системи керування базами даних і методи забезпечення цілісності та безпеки даних.

Системна архітектура охоплює загальну структуру і організацію всіх компонентів системи (апаратних, програмних, мережевих) та їх взаємодію для забезпечення виконання завдань і досягнення цілей системи. Вона включає управління ресурсами, безпеку, продуктивність, масштабованість і надійність.

Архітектурні патерни є повторюваними рішеннями для загальних проблем, що виникають у розробці архітектури системи.

Вони забезпечують шаблони для проектування та реалізації систем, такі як клієнт-сервер, мікросервіси, модель-вид-контролер, однорангова мережа та інші.

Архітектура системи визначає її поведінку, продуктивність, надійність, безпеку і зручність використання. Вона є фундаментом для розробки, розгортання та експлуатації комп'ютерних систем і має критичне значення для їхнього успіху.

Клієнт-серверна архітектура є найпоширенішою в топології "квітка" через свою здатність забезпечувати ефективну, надійну і масштабовану взаємодію між вузлами системи. Ця архітектура дозволяє централізувати обробку даних і управління ресурсами, що значно спрощує адміністрування та підтримку системи. Центральний сервер забезпечує високий рівень контролю і координації, приймаючи запити від клієнтів і надаючи відповідні послуги. Це дозволяє клієнтам зосередитися на виконанні своїх завдань, тоді як сервер обробляє складніші операції і управління ресурсами.

Клієнт-серверна архітектура в топології "квітка" є ефективною моделлю організації мережевої взаємодії в мультикомп'ютерних системах. У цій топології є один центральний вузол, який виконує роль сервера, і декілька периферійних вузлів, що виконують функції клієнтів. Центральний вузол (сервер) розташований в центрі "квітки" і зв'язаний прямими лініями з усіма клієнтами, створюючи структуру, схожу на пелюстки навколо центра.

Сервер у клієнт-серверній архітектурі з топологією "квітка" є компонентом, який забезпечує ефективну і надійну роботу всієї системи. Він розташований у центрі структури, що нагадує квітку, і має пряме з'єднання з кожним клієнтським вузлом, що робить його центральною точкою обробки і керування запитами. Сервер виконує ключову роль у забезпеченні взаємодії між клієнтами, обробці даних і управлінні ресурсами системи.

Одна з основних функцій сервера полягає в обробці запитів, які надходять від клієнтів. Клієнти можуть надсилати різноманітні запити, включаючи запити на

отримання даних, виконання обчислень, управління ресурсами та інші операції. Сервер обробляє ці запити, використовуючи свої обчислювальні потужності і ресурси, і надсилає відповідні відповіді клієнтам. Цей процес забезпечує централізоване керування і обробку інформації, що підвищує ефективність і швидкість роботи системи.

Крім обробки запитів, сервер відповідає за управління ресурсами, які можуть включати бази даних, файлові системи, обчислювальні потужності та інші мережеві ресурси. Сервер забезпечує доступ клієнтів до цих ресурсів, здійснюючи контроль за їх використанням і оптимальним розподілом. Завдяки централізованому керуванню ресурсами, сервер може забезпечити балансування навантаження між різними клієнтами, що дозволяє підтримувати стабільну роботу системи навіть при збільшенні кількості підключень.

Безпека є важливою складовою роботи сервера в такій архітектурі. Сервер забезпечує аутентифікацію і авторизацію клієнтів, контролюючи доступ до системи і захищаючи її від несанкціонованого доступу. Шифрування даних і захист комунікацій також є ключовими аспектами, які забезпечують конфіденційність і цілісність інформації, що передається між клієнтами і сервером.

Масштабованість є ще одним важливим аспектом роботи сервера. У міру зростання потреб системи сервер може бути розширений, додаючи нові обчислювальні потужності, ресурси або навіть додаткові сервери, які працюватимуть разом у кластері. Це дозволяє системі ефективно обробляти зростаючий обсяг запитів і підтримувати високу продуктивність.

Сервер у клієнт-серверній архітектурі з топологією "квітка" також виконує роль координатора взаємодії між клієнтами. Клієнти можуть надсилати повідомлення один одному через сервер, який виступає посередником і забезпечує надійність і швидкість передачі даних. Це особливо важливо для систем, де клієнти повинні взаємодіяти в режимі реального часу.

Клієнти в клієнт-серверній архітектурі з топологією "квітка" є периферійними вузлами, які взаємодіють із центральним сервером для отримання послуг і ресурсів.

Розташовані на "пелюстках" квітки, клієнти мають пряме з'єднання з сервером, що дозволяє їм швидко і надійно надсилати запити та отримувати відповіді.

Основна роль клієнтів полягає у взаємодії з кінцевими користувачами та виконанні завдань, які делегуються серверу для обробки.

Клієнти відповідають за збір і відправку даних на сервер, що включає запити на отримання інформації, обробку даних та виконання команд.

Вони виступають інтерфейсом між користувачем і системою, забезпечуючи зручний доступ до функцій та послуг, що надаються сервером. Кожен клієнт працює автономно, але залежить від сервера для отримання та обробки даних.

Взаємодія клієнтів із сервером відбувається через мережеве з'єднання, де клієнти надсилають свої запити на сервер і отримують відповіді.

Це дозволяє забезпечити швидкий і надійний обмін інформацією, необхідною для виконання різних завдань. Клієнти можуть також відправляти запити, що вимагають тривалої обробки, та отримувати повідомлення від сервера про завершення цих запитів.

Клієнти грають важливу роль у масштабованості системи. У разі збільшення навантаження, до системи можна додавати нових клієнтів без значного впливу на продуктивність.

Кожен новий клієнт підключається до сервера, і його запити обробляються в порядку черги. Це дозволяє системі ефективно масштабуватися і підтримувати високу продуктивність навіть при збільшенні кількості користувачів.

У такій архітектурі безпека в роботі клієнтів є не менш важливим аспектом, ніж безпека сервера.

Клієнти повинні проходити аутентифікацію на сервері для отримання доступу до ресурсів системи.

Це забезпечує контроль доступу та захист від несанкціонованого використання системи.

Клієнти також використовують шифрування для захисту даних, що передаються між ними і сервером, забезпечуючи конфіденційність і цілісність інформації.

Клієнти можуть взаємодіяти не тільки із сервером, але й один з одним через сервер. Наприклад, вони можуть надсилати повідомлення, які сервер пересилає іншим клієнтам.

Це забезпечує координацію і співпрацю між клієнтами, дозволяючи реалізувати складніші сценарії взаємодії, такі як обмін повідомленнями, спільна робота над завданнями та інші форми співпраці.

Зв'язок між клієнтами і сервером в архітектурі з топологією "квітка" є ключовим елементом, що забезпечує ефективну взаємодію і обмін інформацією в системі.

Ця топологія передбачає пряме з'єднання кожного клієнта з центральним сервером, що знаходиться в центрі "квітки".

Така організація дозволяє забезпечити високу швидкість і надійність передачі даних, оскільки кожен клієнт може безпосередньо надсилати свої запити на сервер і отримувати відповіді.

Клієнти виступають ініціаторами зв'язку, надсилаючи запити на сервер. Ці запити можуть включати запити на отримання даних, виконання обчислень, доступ до ресурсів та інші операції. Сервер, отримавши запит, обробляє його, використовуючи свої обчислювальні потужності та ресурси, і відправляє відповідь назад клієнту.

Цей процес забезпечує централізоване керування інформацією і ресурсами, що підвищує ефективність роботи системи.

Комунікація між клієнтами і сервером відбувається через мережеві протоколи, які забезпечують надійну і безпечну передачу даних. Клієнти використовують мережеві сокети для встановлення з'єднання з сервером,

відправки запитів і отримання відповідей. Ця взаємодія відбувається в режимі реального часу, що дозволяє оперативно обробляти запити і надавати необхідні дані або послуги.

Зв'язок між клієнтами і сервером також підтримує координацію і співпрацю між різними клієнтами системи.

Сервер може виступати посередником у передачі повідомлень між клієнтами, забезпечуючи надійність і швидкість комунікації.

Це дозволяє клієнтам обмінюватися даними і співпрацювати в режимі реального часу, що є важливим для багатьох прикладних сценаріїв, таких як спільна робота над проєктами, обмін повідомленнями та інші форми взаємодії.

Надійність зв'язку між клієнтами і сервером забезпечується за рахунок використання протоколів передачі даних з підтвердженням доставки, які гарантують, що дані будуть доставлені адресату і отримані без помилок.

У разі виникнення проблем із зв'язком, сервер і клієнти можуть повторювати запити і відповіді, що забезпечує стійкість системи до збоїв і втрат даних.

Однією з найважливіших частин архітектури мультикомп’ютерної системи є проміжне програмне забезпечення.

Проміжне програмне забезпечення, або middleware, є програмним шаром, що забезпечує взаємодію та обмін даними між різними додатками та сервісами у розподілених обчислювальних середовищах.

Це програмне забезпечення виступає посередником між операційною системою та додатками, дозволяючи їм ефективно взаємодіяти один з одним, незалежно від платформи або протоколів, які вони використовують.

Проміжне програмне забезпечення відіграє ключову роль у створенні масштабованих, інтегрованих і надійних інформаційних систем.

Одним з основних типів проміжного програмного забезпечення є сервери додатків, які забезпечують середовище виконання для веб-додатків, обробляючи запити від клієнтів і забезпечуючи доступ до необхідних ресурсів, таких як бази

даних. Такі сервери підтримують різні мови програмування та фреймворки, забезпечуючи стандартизовані інтерфейси для розробників.

Іншим важливим видом є засоби інтеграції, такі як шини сервісно-орієнтованої архітектури (SOA) або брокери повідомлень.

Вони забезпечують надійну передачу повідомлень між різними системами і компонентами, підтримуючи асинхронний обмін даними та розподілену обробку.

Це дозволяє організаціям інтегрувати різні додатки та сервіси, незалежно від їхньої архітектури та технологій, використовуючи стандартизовані протоколи, такі як SOAP або REST.

Проміжне програмне забезпечення для управління транзакціями дозволяє забезпечити цілісність та узгодженість даних у розподілених системах.

Воно забезпечує координацію і контроль транзакцій, гарантує, що всі операції виконуються успішно або відміняються в разі виникнення помилки. Це критично важливо для фінансових систем, електронної комерції та інших областей, де необхідно забезпечити високий рівень надійності.

Засоби управління чергами дозволяють організувати обробку завдань у черзі, що забезпечує балансування навантаження та асинхронну обробку відповідних запитів.

Це корисно для систем з високим навантаженням, де необхідно забезпечити рівномірний розподіл ресурсів і уникнути перевантаження сервісів.

Проміжне програмне забезпечення для обробки подій забезпечує реєстрацію та обробку подій у реальному часі, дозволяючи додаткам реагувати на зміни в системі або навколишньому середовищі.

Це особливо важливо для систем моніторингу, управління процесами і додатків, що вимагають швидкої реакції на події.

У середовищах, де використовуються різні мови програмування і платформи, проміжне програмне забезпечення забезпечує сумісність і взаємодію між ними.

Це дозволяє розробникам створювати комплексні системи, які можуть використовуватися на різних платформах і пристроях, підтримуючи високу гнучкість і масштабованість.

2.2 Програмний засіб

Для успішного створення системи, що складається з певної кількості різноманітних пристроїв (не обов’язково ПК, це можуть бути і сервери, маршрутизатори, тощо), повинно бути належним чином здійснено проектування системи, а для цього в свою чергу необхіжно обрати програмний засіб, що найбільше підходить для виконання завдання.

Враховуючи сучасні тенденції, спектр вибору програм для реалізації мультикомп’ютерної системи є доволі широким.

Такі програмні засобизабезпечують ефективну комунікацію, управління ресурсами, обробку даних та координацію між різними вузлами системи.

Приклади сучасних програм, які використовуються для реалізації мультикомп’ютерних систем наведено в таблиці 2.1.

В таблиці наведено лише декілька прикладів, в той час, як цей список набагато ширший.

Проте в ній відображено програмні засоби, які більше спрямовані на розгортання, і роботу мультикомп’ютерних систем.

Далі буде розглянуто кілька прикладів засобів для моделювання систем.

Засоби моделювання систем відіграють важливу роль у розробці, тестуванні та оптимізації мультикомп'ютерних систем.

Вони дозволяють симулювати різні аспекти роботи системи, аналізувати її поведінку під різними умовами, виявляти потенційні проблеми та покращувати продуктивність.

Приклади сучасних засобів моделювання наведено в таблиці 2.2.

Таблиця 2.1 – Програмні засоби для реалізації мультикомп’ютерних систем

Назва	Тип засобу	Особливоті
Apache Hadoop	Фреймворк для розподіленого зберігання та обробки великих даних	Його основні компоненти включають HDFS (Hadoop Distributed File System) для зберігання даних і MapReduce для обробки даних. Hadoop також включає YARN (Yet Another Resource Negotiator) для управління ресурсами кластера.
Apache Spark	система для обробки великих даних у режимі реального часу	Spark забезпечує швидку обробку даних завдяки розподіленій обробці в пам'яті і підтримує різні моделі обробки, включаючи SQL, стрімінг, машинне навчання та обробку графів.

Продовження таблиці 2.1 – Програмні засоби для реалізації мультикомп’ютерних систем

Docker	платформа для розгортання, запуску і управління контейнеризованими додатками	Docker дозволяє створювати контейнеризовані середовища, які можуть легко масштабуватися і переміщуватися між різними середовищами.
Kubernetes	система оркестрації контейнерів, яка автоматизує розгортання, управління і масштабування контейнеризованих додатків	Kubernetes забезпечує високу доступність, відновлення після збоїв і автоматичне масштабування контейнерів.
Apache Kafka	розподілена стрімінгова платформа, яка дозволяє створювати системи обробки даних у реальному часі	Kafka використовується для побудови розподілених систем обміну повідомленнями і обробки подій.
RabbitMQ	система обміну повідомленнями, яка підтримує протокол AMQP (Advanced Message Queuing Protocol)	RabbitMQ забезпечує надійну передачу повідомлень між додатками, підтримує черги і маршрутизацію повідомлень.

Таблиця 2.2 – Програмні засоби моделювання мультикомп’ютерних систем

Назва засобу	Тип засобу	Особливості
SimGrid	інструментарій для моделювання великих розподілених систем, таких як хмарні обчислення, суперкомп'ютери та мережі	SimGrid дозволяє симулювати продуктивність додатків і інфраструктури, проводити експерименти з різними конфігураціями систем та аналізувати їх ефективність.
MATLAB/Simulink	середовище для моделювання, симуляції та аналізу динамічних систем	MATLAB/Simulink підтримує широкий спектр застосувань, включаючи моделювання комунікаційних систем, розподілених обчислень, управління ресурсами та інші аспекти мультикомп'ютерних систем.
CloudSim	фреймворк для моделювання та симуляції хмарних обчислень і сервісів	loudSim дозволяє моделювати різні аспекти хмарної інфраструктури, такі як управління ресурсами, планування завдань, енергоспоживання та продуктивність.

Є і інші засоби, проте згадувати всі є недоцільним, наведених прикладів цілком достатньо.

Окремо можна винести Cisco Packet Tracer.

Даний програмний засіб може бути використаний для створення моделі системи, а також дослідження її роботи.

Cisco доволі зручно використовувати, оскільки він має доволі широкий функціонал, і надає достатньо можливостей для комфортної роботи.

Cisco Packet Tracer — це потужний інструмент для моделювання мережевих систем, який використовується як для навчальних, так і для професійних споживчих цілей.

Він дозволяє створювати віртуальні мережі, симулювати їх роботу, тестувати конфігурації та аналізувати мережевий трафік.

До основних можливостей Cisco відносяться: створення мережевих топологій, конфігурація мережевих пристроїв, симуляція мережевої активності, навчальні ресурси, спільна робота.

Cisco Packet Tracer дозволяє створювати різноманітні мережеві топології, включаючи LAN, WAN, WLAN та інші.

Є можливість додавати різні типи пристроїв, такі як маршрутизатори, комутатори, ПК, сервери, бездротові пристрої та інші.

Інструмент надає можливість конфігурувати мережеві пристрої за допомогою командного рядка (CLI) або графічного інтерфейсу (GUI), налаштовувати IP-адреси, маршрутизаційні протоколи, безпекові політики та інші параметри.

Packet Tracer дозволяє симулювати роботу мережі, включаючи передачу даних, комутацію та маршрутизацію пакетів, роботу протоколів та сервісів, спостерігати за потоком трафіку в реальному часі та аналізувати його.

Проте, оскільки робота буде вестись з проміжним програмним забезпеченням, то для реалізації буде використано Microsoft Visual Studio. Microsoft Visual Studio — це інтегроване середовище розробки (IDE), яке

використовується програмістами для створення програмного забезпечення на різних мовах програмування, таких як C++, C#, Visual Basic, Python і багато інших. Visual Studio забезпечує широкий спектр інструментів і функцій, які сприяють підвищенню продуктивності розробників і полегшують процес створення, тестування та налагодження програм.

Серед основних можливостей Visual Studio варто виділити зручний редактор коду, який підтримує синтаксичне підсвічування, автодоповнення, рефакторинг та інші інструменти для ефективної роботи з кодом.

Інтегрована система управління версіями дозволяє легко працювати з популярними системами контролю версій, такими як Git, що полегшує командну розробку і відстеження змін у проєктах.

Крім того, Visual Studio надає розробникам потужні засоби для налагодження додатків, включаючи можливість встановлення точок зупину, огляд значень змінних у реальному часі, трасування стека викликів і аналізу вихідних даних.

Це дозволяє швидко ідентифікувати та виправляти помилки, забезпечуючи стабільну і надійну роботу програмного забезпечення.

Інтеграція з такими системами, як Microsoft Azure, дозволяє розробникам створювати, тестувати та розгортати хмарні додатки безпосередньо з середовища Visual Studio.

Visual Studio також підтримує широкий спектр технологій для веб-розробки, таких як ASP.NET, HTML, CSS, JavaScript і TypeScript.

Це робить можливим створення сучасних веб-додатків з використанням останніх стандартів і найкращих практик у веб-розробці. Інтегрований дизайнер інтерфейсу користувача дозволяє розробникам створювати привабливі і функціональні інтерфейси за допомогою простого перетягування елементів на форму, значно спрощуючи процес розробки графічних додатків.

Visual Studio пропонує вбудовані інструменти для роботи з базами даних, що дозволяє розробникам створювати, управляти і тестувати бази даних

безпосередньо з IDE. Це включає підтримку різних систем управління базами даних, таких як SQL Server, MySQL, PostgreSQL та інших.

За допомогою цих інструментів розробники можуть створювати складні запити, проводити аналіз даних і оптимізувати продуктивність використовуваних баз даних.

Розширювана архітектура Visual Studio дозволяє користувачам налаштовувати середовище під свої потреби за допомогою різноманітних розширень і плагінів, доступних у Visual Studio Marketplace.

Це включає інструменти для аналізу коду, інтеграції з іншими сервісами, додаткові засоби налагодження та багато іншого.

Завдяки цьому Visual Studio може бути адаптовано до будь-якого робочого процесу і задовольняти вимоги найвибагливіших розробників.

У підсумку, Microsoft Visual Studio є потужним і гнучким інструментом, який задовольняє потреби як початківців, так і досвідчених розробників, надаючи їм усі необхідні засоби для створення високоякісного використовуваного програмного забезпечення.

Завдяки своїй багатофункціональності, зручності використання та широким можливостям налаштування, Visual Studio залишається одним з найпопулярніших середовищ розробки у світі.

C++ є потужною мовою програмування, яка знайшла широке застосування в написанні проміжного програмного забезпечення (ПЗ).

Вона володіє великим набором функцій і можливостей, що дозволяють розробникам створювати ефективне і надійне ПЗ для різних застосувань. C++ володіє високою продуктивністю і може працювати безпосередньо з апаратним обладнанням, що робить її ідеальним вибором для розробки проміжного ПЗ, яке потребує оптимального використання ресурсів комп'ютера.

Мова C++ забезпечує можливість низькорівневого програмування, що дозволяє розробникам працювати з пам'яттю і обчислювальними ресурсами

машини на високому рівні контролю. Це особливо важливо для написання ПЗ, яке потребує ефективного управління ресурсами і оптимізації швидкодії.

Крім того, C++ підтримує об'єктно-орієнтований підхід до програмування, що дозволяє розробникам організовувати код в логічні блоки (об'єкти), що полегшує розробку і підтримку ПЗ великих систем.

За допомогою C++ можна ефективно реалізувати різноманітні функції, які вимагають високої швидкодії обробки даних, такі як алгоритми обробки сигналів, оптимізації ресурсів мережі, керування введенням-виведенням та інші.

Ця мова дозволяє розробникам працювати на рівні машини, що забезпечує високу продуктивність і надійність ПЗ.

Крім того, C++ має широку підтримку та активну спільноту розробників, що дозволяє отримувати швидку підтримку та розв'язання проблем.

Ця мова залишається однією з основних у великих і складних програмних проєктах, які вимагають високої ефективності і надійності в роботі.

2.3 Огляд та властивості топології

Топології квітка та зірка є двома важливими структурними підходами до організації комп'ютерних мереж, кожна з яких має свої унікальні характеристики, переваги та недоліки.

В обох випадках мова йде про підключення пристроїв таким чином, щоб забезпечити ефективний обмін даними, але вони мають суттєві відмінності в організації з'єднань між вузлами мережі.

Топологія квітка, хоч і виглядає складнішою та більш витонченою, насправді базується на принципах топології зірка.

В її основі також лежить центральний вузол, який виконує ключову роль у маршрутизації трафіку між різними пристроями в мережі.

Центральний вузол у топології квітка є тим самим критичним елементом, що і в топології зірка, забезпечуючи зв'язок між усіма периферійними вузлами.

Це означає, що вся структура топології квітка фактично побудована на фундаменті зірки, де центральний вузол є основним пунктом з'єднання.

Однак, на відміну від класичної зірки, топологія квітка включає додаткові з'єднання між периферійними вузлами, утворюючи більш складну і надійну мережу.

Ці додаткові з'єднання додають резервні шляхи передачі даних, що підвищує загальну надійність мережі і забезпечує стійкість до збоїв.

Але навіть з цими додатковими з'єднаннями, центральний вузол залишається ключовим елементом мережі, як і в топології зірка.

Таким чином, основа топології квітка залишається такою ж, як і в зірковій структурі, з центральним вузлом, що грає головну роль у з'єднанні та управлінні трафіком.

Цей факт підкреслює, що основні переваги топології зірка, такі як простота реалізації та управління, також присутні в топології квітка.

Центральний вузол забезпечує легкість налаштування та обслуговування мережі, дозволяючи адміністраторам ефективно контролювати і керувати мережею.

Отже, можна стверджувати, що топологія квітка використовує всі переваги топології зірка, збагачуючи їх додатковими з'єднаннями для підвищення надійності та стійкості мережі.

Це робить топологію квітка потужним варіантом для більш складних мережевих сценаріїв, одночасно зберігаючи основні принципи, на яких базується зіркова топологія.

Ще одним вагомим аргументом на користь топології зірка є її економічність.

Побудова мережі за цією топологією вимагає менше кабелів і обладнання в порівнянні з топологією квітка, де кожен вузол може бути підключений до кількох інших вузлів.

Це значно знижує витрати на матеріали та монтаж, що може бути критично важливим для невеликих організацій або проєктів з обмеженим бюджетом.

Крім того, зменшення кількості з'єднань знижує ризик фізичних пошкоджень мережевих кабелів, що підвищує загальну надійність мережі.

Топологія зірка також забезпечує простоту в обслуговуванні та моніторингу.

Центральний вузол виконує роль єдиної точки управління, що дозволяє адміністраторам легко відстежувати стан мережі, виявляти і усувати проблеми.

Це також полегшує процеси оновлення та модернізації мережі, оскільки зміни або додавання нових вузлів можуть бути здійснені без значного впливу на інші частини мережі.

Зручність моніторингу і управління особливо важлива у великих організаціях, де складність мережевої інфраструктури може ускладнити процеси підтримки.

Швидкість передачі даних у топології зірка є ще однією суттєвою перевагою.

Оскільки кожен вузол має своє власне з'єднання з центральним вузлом, немає необхідності ділити пропускну здатність мережі між кількома вузлами.

Це дозволяє забезпечити високу швидкість і ефективність передачі даних, що особливо важливо для застосунків, які вимагають швидкої обробки інформації або великої пропускної здатності.

У порівнянні з топологією квітка, де трафік може проходити через кілька вузлів, це забезпечує більш стабільну і передбачувану продуктивність.

Безпека мережі в топології зірка також може бути ефективніше забезпечена завдяки централізованому контролю.

Центральний вузол може використовувати різноманітні засоби захисту, такі як міжмережеві екрани, системи виявлення вторгнень і засоби шифрування, що дозволяє ефективно контролювати доступ і захищати дані.

Це створює єдину точку контролю, де можуть бути впроваджені всі необхідні заходи безпеки, спрощуючи управління захистом мережі.

З огляду на всі ці фактори, топологія зірка може бути доцільною в багатьох випадках, навіть якщо існують альтернативи з більшою надійністю, як-от топологія квітка.

Простота реалізації та управління, економічність, висока швидкість передачі даних і централізований контроль роблять топологію зірка привабливим варіантом для багатьох організацій, особливо для тих, які прагнуть до ефективного використання ресурсів та зручного управління мережею.

Схематичний вигляд мережі з топологією зірка наведено на рисунку 2.1.

А також на рисунку 2.2. зображено схематичний вигляд мережі з топологією квітка.

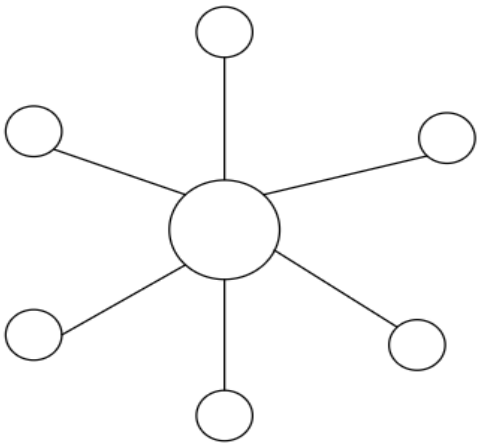


Рисунок 2.1. – схематичне зображення “зірки”

Порівнявши схематичні зображення топологій, з якими ведеться робота, в результаті маємо не велику, але достатньо помітну візуальну розбіжність.

В топології квітка порівняно із зіркою, як вже зазначалось раніше більша кількість зв’язків між вузлами “пелюстками”, та центральним вузлом.

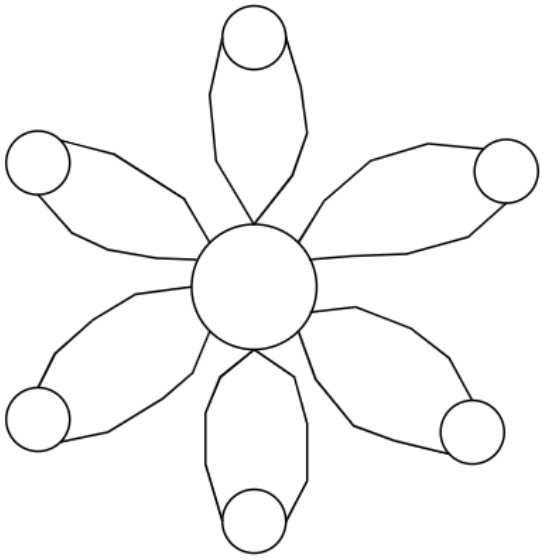


Рисунок 2.2. – схематичне зображення “квітки”

В даному випадку зображено по 2 з’єднання з кожним вузлом для найбільшого уподібнення вигляду схеми до квітки, яка існує в природі.

Проте це не означає, що 2 зв’язки це максимальна можлива кількість з’єднань, кількість зв’язків може бути змінена в одній системі відповідно до необхідності.

Крім того, розглядаючи саме квітку можна об’єднувати одразу декілька вузлів в одну “пелюстку”.

Таке можливо, коли, наприклад, мережа складається з декількох підмереж, кожна з яких виконує певні функції, відмінні від функцій інших підмереж (внутрішня мережа банку, тощо).

У випадку з мережею банку однією пелюсткою буде підмережа працівників банку, що працюють з клієнтами, іншою підмережа, де працівники виконують роботу не пов’язану з клієнтами, або підмережа, що об’єднує сервери, які виконують роль сховища даних.

В даному випадку квіткою мережу можна буде назвати лише умовно, оскільки при умові відсутності прямих зв’язків всіх пристроїв з центральним вузлом, а саме наявності додаткових точок з’єдання всіх пристроїв кожної пелюстки з умовним центром пелюстки, ми отримаємо іншу топологію, що називається “сніжинка”.

Сніжинка в порівнянні з квіткою чи зіркою має більш складну структуру, тому розглядати її в даному випадку немає сенсу.

А оскільки сама квітка також є ускладненням зірки, то найбільш ефективним варіантом використання топології квітка буде застосування топології зірка, як найпростішого варіанта квітки.

2.4 Висновки

У межах розділу 2 було здійснено детальний огляд архітектури в мультикомп'ютерних системах загалом, а також розглянуто різні види архітектур та їхні властивості.

Це дозволило отримати глибше розуміння структури і функціонування мультикомп'ютерних систем, їхніх переваг та недоліків у різних сценаріях використання.

Особливу увагу було приділено аналізу корисного та актуального програмного забезпечення, яке може бути використане для розробки і тестування проміжного програмного забезпечення.

Проведений аналіз включав порівняння особливостей різних програмних засобів, що дозволило визначити їхні сильні і слабкі сторони, а також сферу найбільш ефективного застосування.

Також у розділі було проведено огляд топологій, які використовуються в мультикомп'ютерних системах, із детальним розглядом їхніх характеристик і можливих способів заміни на більш зручні для роботи.

Це включало оцінку доцільності використання певних топологій у залежності від конкретних умов експлуатації і вимог до системи.

Зокрема, розглядалися можливості адаптації існуючих топологій для підвищення ефективності і надійності роботи мережі.

Таким чином, розділ 2 надав всебічний огляд архітектур мультикомп'ютерних систем та програмних засобів для їхньої підтримки, що створює базу для подальших досліджень і практичної реалізації проміжного програмного забезпечення.

Проведений аналіз дозволяє зробити обґрунтований вибір програмного забезпечення і топології для створення ефективної і надійної мультикомп'ютерної системи, що відповідає сучасним вимогам і стандартам.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОМІЖНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МУЛЬТИКОМП'ЮТЕРНОЇ СИСТЕМИ З ТОПОЛОГІЄЮ КВІТКА

3.1 Робота системи

Важливість алгоритмів полягає в тому, що вони визначають послідовність дій, які програма виконує для досягнення поставлених цілей.

Розгляд алгоритмів включає аналіз їх структури, логіки, а також методів оптимізації, що спрямовані на підвищення продуктивності та надійності програми.

Опис алгоритмів допоможе зрозуміти, як саме працює програмне забезпечення, які кроки виконуються на кожному етапі обробки даних, і як ці кроки взаємодіють між собою для досягнення бажаного результату.

Це дозволить більш детально оцінити ефективність розроблених рішень, виявити можливі проблеми та запропонувати шляхи їх усунення. Розуміння алгоритмів роботи програми є ключовим для подальшого вдосконалення програмного забезпечення та його адаптації до різних умов використання.

Розгляд почнемо із загальної структури мультикомп'ютерних систем. Загальна структура зображена на рисунку 3.1.

В даному випадку кожен вузол має своє ядро , яке в свою чергу містить моулі, що допомагають керувати локальними ресурсами, такими як локальний диск, пам'ять, локальний процесор, та інші.

Окремо виділимо локальний модуль, що відповідає за міжпроцесорну взаємодію.

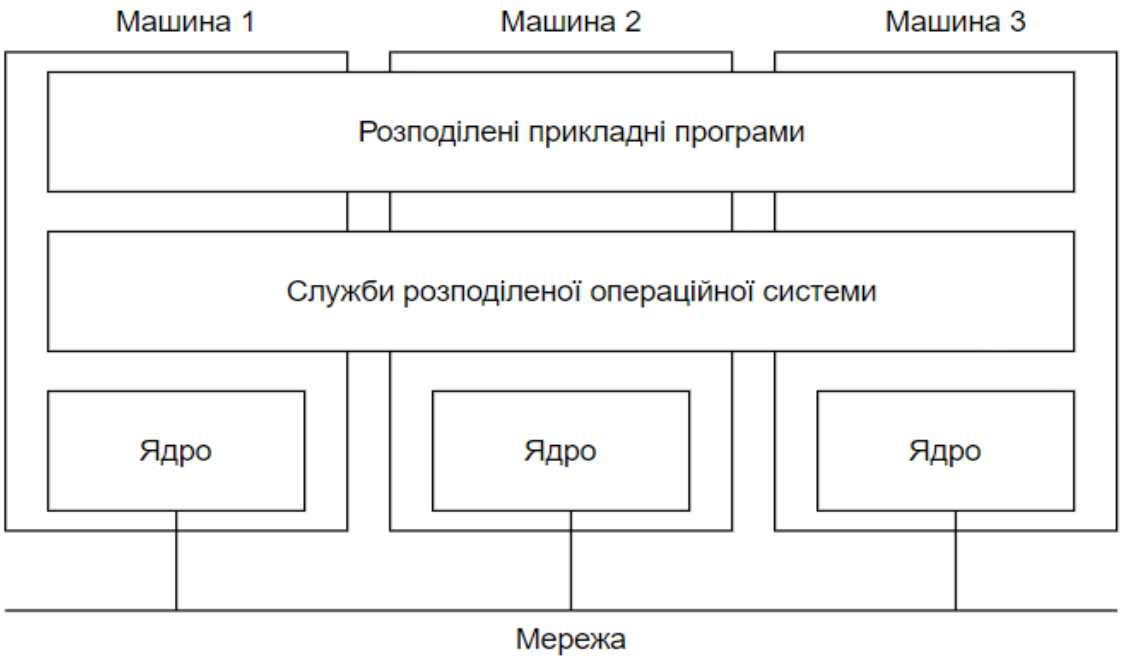


Рисунок 3.1. – загальна структура мультикомп’ютерної системи

Його основні функції - це відсилання повідомлень до інших вузлів, а також прийом повідомлень від них.

Власне ці передачі повідомлень і є єдиним можливим способом зв’язку між комп’ютерами в системі. Над кожним з ядер розміщено рівень певного програмного забезпечення, яке має загальне призначення, і виконує функцію реалізації операційної системи, яка забезпечує підтримку паралельної роботи із різними завданнями.

Для обміну повідомленнями можуть використовуватися системні сокети (unix sockets) у POSIX-системах або іменовані канали (named pipes).

Сокети – це програмні інтерфейси, які забезпечують обмін даними між програмами через мережу.

Вони використовуються для встановлення з'єднань і передачі даних між клієнтськими і серверними додатками.

Сокети працюють на основі мережевих протоколів, таких як TCP (Transmission Control Protocol) і UDP (User Datagram Protocol). Принцип роботи

сокетів можна описати шістьма етапами: створення сокета, прив’язка сокета, прослуховування, встановлення з’єднання, передача даних, закриття з’єднання.

При створенні сокета клієнт і сервер створюють сокет, вказуючи параметри такі, як домен (IPv4 або IPv6), тип (TCP або UDP) і протокол.

Під час прив’язки сокета сервер прив'язує сокет до конкретної IP-адреси і порту, які він буде використовувати для прийому вхідних з'єднань.

Для прослуховування сервер переводить сокет в режим прослуховування, щоб приймати вхідні з'єднання від клієнтів.

Щоб встановити з’єднання клієнт ініціює з'єднання з сервером, вказуючи IP-адресу і порт сервера.

Сервер приймає з'єднання і створює новий сокет для обробки цього з'єднання, залишаючи основний сокет у режимі прослуховування.

Передача даних відбувається наступним чином: після встановлення з'єднання клієнт і сервер можуть обмінюватися даними.

В TCP-з'єднанні дані передаються як потік байтів, що забезпечує надійність і порядок доставки. В UDP-з'єднанні дані передаються у вигляді окремих пакетів, що швидше, але менш надійно.

Закриття з’єднання відбувається після завершення обміну даними, клієнт і сервер закривають сокети, звільняючи ресурси.

Розглянемо приклади реалізації деяких топологій за допомогою C++.

```
#include <iostream>
#include <vector>
using namespace std;
class NetworkNode {
public:
    int id;
    NetworkNode(int id) : id(id) {}
    void sendData(NetworkNode* destination, const
string& data) {
```

```
            cout << "Node " << id << " sending data to Node
" << destination->id << ": " << data << endl;
        }
    };
```

Цей код створює просту модель мережевого вузла в мережевій системі на мові C++. Докладний опис того, що виконує кожен його компонент:

```
#include <iostream>
#include <vector>
using namespace std;
```

Ці рядки включають заголовкові файли для вводу/виводу та використання стандартного простору імен (namespace). iostream використовується для роботи з потоками вводу/виводу, а vector надає доступ до контейнера векторів з стандартної бібліотеки C++.

Оголошення класу NetworkNode:

```
class NetworkNode {
public:
int id;
NetworkNode(int id) : id(id) {}
void sendData(NetworkNode* destination, const string&
data) {
cout << "Node " << id << " sending data to Node " <<
destination->id << ": " << data << endl;
}
};
```

Це оголошення класу NetworkNode, який моделює вузол в мережі. Клас має два члени:

int id;

Ця змінна зберігає ідентифікатор вузла (ймовірно, унікальний для кожного вузла).

Конструктор NetworkNode(int id):

NetworkNode(int id) : id(id) {}

Це конструктор класу, який ініціалізує вузол з заданим ідентифікатором id.

Метод sendData:

void sendData(NetworkNode* destination, const string& data) {

cout << "Node " << id << " sending data to Node " << destination->id << ": " << data << endl;

}

Цей метод симулює відправку даних від одного вузла до іншого. Він приймає вказівник на об'єкт NetworkNode як пункт призначення (destination) і рядок data, який містить дані для відправки. Метод виводить повідомлення в консоль, вказуючи, який вузол відправляє дані, який вузол отримує дані, і що саме відправляється.

Загалом, цей код моделює базову функціональність вузла в мережі, який може відправляти дані іншим вузлам, і використовує консольний вивід для симуляції цього процесу.

```
class StarTopology {
public:
NetworkNode* centralNode;
vector<NetworkNode*> nodes;

    StarTopology(int numNodes) {
        centralNode = new NetworkNode(0);
        for (int i = 1; i <= numNodes; ++i) {
            nodes.push_back(new NetworkNode(i));
        }
    }
```

Цей код створює клас StarTopology, який моделює мережеву топологію "зірка". У цій топології є центральний вузол, до якого підключені всі інші вузли.

Конструктор StarTopology(int numNodes):

```
StarTopology(int numNodes) {
```

```
        centralNode = new NetworkNode(0);
        for (int i = 1; i <= numNodes; ++i) {
            nodes.push_back(new NetworkNode(i));
        }
    }
```

Це конструктор класу, який приймає ціле число numNodes, що визначає кількість вузлів у топології (крім центрального вузла). Конструктор виконує наступні дії:

Створює центральний вузол з ідентифікатором 0 і присвоює його змінній centralNode.

В циклі for від 1 до numNodes (включно) створює нові вузли з унікальними ідентифікаторами (від 1 до numNodes) і додає їх у вектор nodes.

Цей клас дозволяє створити мережеву топологію "зірка" з одним центральним вузлом та певною кількістю підключених до нього периферійних вузлів. Кожен вузол представлений об'єктом NetworkNode, а вектор nodes зберігає всі ці вузли для подальшого використання.

```
    ~StarTopology() {
            delete centralNode;
            for (auto node : nodes) {
                delete node;
            }
        }
```

Цей доданий фрагмент коду є деструктором класу StarTopology, який відповідає за правильне звільнення пам'яті, виділеної під час створення об'єктів класу.

```
    void removeNode(int id) {
        if (id > 0 && id <= nodes.size()) {
            delete nodes[id - 1];
            nodes.erase(nodes.begin() + (id - 1));
```

```
            cout << "Node " << id << " removed from the
network.\n";
            // Reassign IDs for remaining nodes
            for (int i = id - 1; i < nodes.size(); ++i) {
                nodes[i]->id = i + 1;
            }
        } else {
            cout << "Invalid node ID.\n";
        }
    }
```

В цій частині коду виконуються такі дії, як:

- Перевірка дійсності ID
- Видалення вузла
- Видалення вузла з вектора
- Повідомлення про видалення вузла
- Перепризначення ID для залишених вузлів
- Повідомлення про недійсний ID

Метод communicate:

```
void communicate(int from, int to, const string& data) {
        if (from > 0 && from <= nodes.size() && to > 0 &&
to <= nodes.size()) {
            nodes[from - 1]->sendData(centralNode, data);
            centralNode->sendData(nodes[to - 1], data);
        } else {
            cout << "Invalid node IDs for
communication.\n";
        }
    }
```

Перевіряє, чи є обидва from і to ID дійсними.

Відправляє дані від вузла from до центрального вузла.

Центральний вузол пересилає дані до вузла to.

Якщо будь-який з ID недійсний, виводиться відповідне повідомлення.

Метод displayNetwork

```
    void displayNetwork() const {
        cout << "Central Node ID: " << centralNode->id <<
endl;
        cout << "Connected Nodes: ";
        for (const auto& node : nodes) {
            cout << node->id << " ";
        }
        cout << endl;
```

Виводить ID центрального вузла.

Виводить ID всіх вузлів, підключених до центрального вузла.

```
    class RingTopology {
    public:
        vector<NetworkNode*> nodes;
        RingTopology(int numNodes) {
            for (int i = 0; i < numNodes; ++i) {
                nodes.push_back(new NetworkNode(i));
            }
      }
        void communicate(int from, int to, const string&
data) {
            cout << "Ring topology communication:\n";
            int currentNode = from;
            while (currentNode != to) {
                int nextNode = (currentNode + 1) %
nodes.size();
```

```
                nodes[currentNode]-
>sendData(nodes[nextNode], data);
                currentNode = nextNode;
            }
        }
        ~RingTopology() {
            for (auto node : nodes) {
                delete node;
            }
        }
    };
```

RingTopology: Клас, що представляє мережеву топологію "кільце".

nodes: Вектор вказівників на вузли в кільці.

RingTopology(int numNodes): Конструктор, що створює кільце з заданої кількості вузлів.

communicate(int from, int to, const string& data): Метод для передачі даних від одного вузла до іншого, пересилаючи дані через кожен проміжний вузол до досягнення кінцевого пункту призначення.

~RingTopology(): Деструктор, що видаляє всі вузли, звільняючи виділену пам'ять.

Робота методу communicate

communicate: Починає передачу даних від вузла from до вузла to.

Вузол from відправляє дані до наступного вузла в кільці.

Цей процес повторюється, поки дані не досягнуть вузла to.

Цей код забезпечує базову функціональність для створення та управління мережею з топологією "кільце" і передачі даних між вузлами в такій мережі.

Проте загальний алгоритм роботи програми буде мати дещо інший вигляд, відповідно до топології.

Блок-схеми алгоритму наведено на рисунках 3.2 та 3.3.

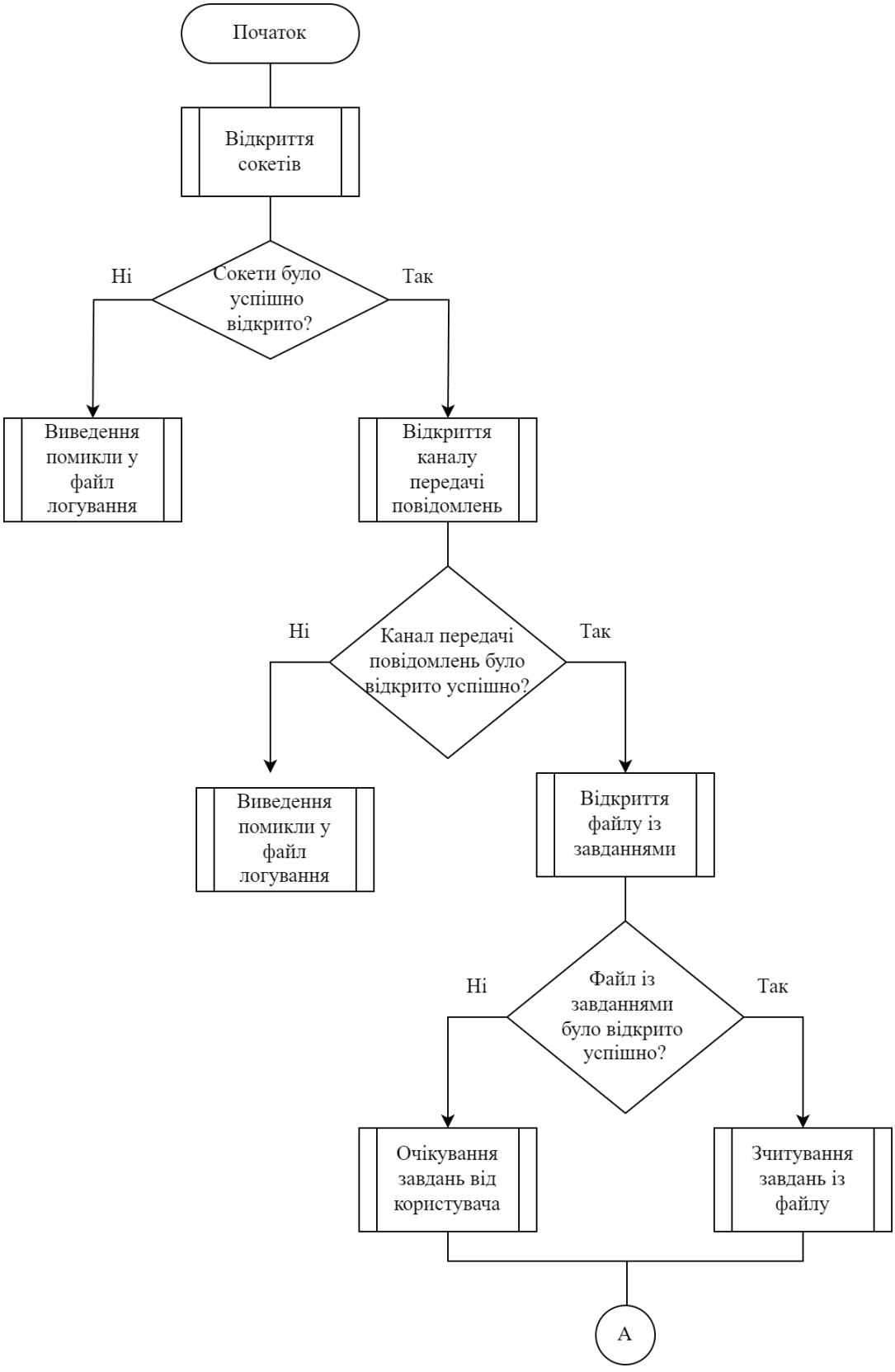


Рисунок 3.2.- Початок блок-схеми алгоритму роботи програми

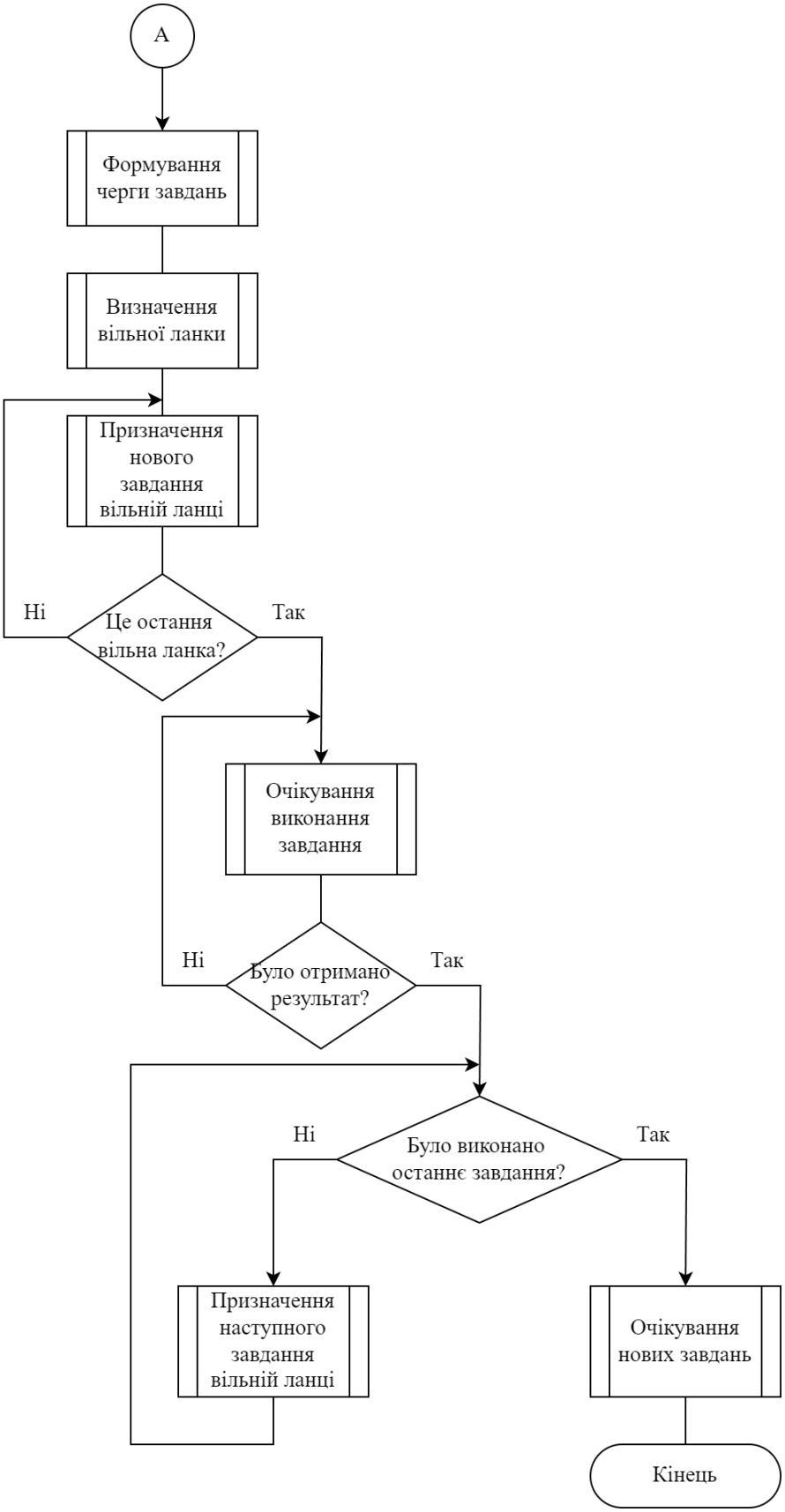


Рисунок 3.3 - кінець блок-схеми алгоритму роботи програми

3.2 Оцінка ефективності роботи системи з топологією зірка.

Ефективність функціонування комп'ютерних систем є критично важливим аспектом у сучасних обчислювальних середовищах, де швидкість і надійність виконання завдань безпосередньо впливають на продуктивність та якість обслуговування користувачів.

Традиційні сервери, що використовуються для обробки запитів, мають певні обмеження, зокрема в питаннях масштабованості та стійкості до відмов.

Натомість мультикомп'ютерні системи з топологіями квітка або зірка пропонують більш гнучкі і надійні рішення завдяки своїй архітектурі, яка сприяє ефективнішому розподілу навантаження і підвищенню загальної продуктивності системи.

Дослідження цих топологій та їх порівняння з традиційними серверами є важливим для розуміння шляхів оптимізації обробки запитів, покращення часу відгуку і забезпечення безперебійної роботи мережі навіть при високих навантаженнях.

Це підводить до необхідності детального порівняння ефективності роботи традиційного сервера і мультикомп'ютерної системи з топологією квітка або зірка, з метою визначення найбільш оптимальних рішень для різних сценаріїв використання.

Порівняємо можливості роботи нашої топології зі звичайним сервером. Кругові діаграми розподілу ефективності зображено на рисунках 3.4 та 3.5.

Рисунок. 3.4. – Діаграма ефективності роботи топології зірка

Рисунок 3.5. - Діаграма ефективності роботи звичайного сервера

Ці діаграми ілюструють, що топологія зірка має явні переваги в більшості ключових параметрів, хоча традиційний сервер може бути привабливішим з точки зору вартості.

Для ілюстрації порівняння кількості оброблених запитів також буде використано діаграму. Діаграма порівняння кількості запитів, а також середнього часу обробки одного запиту наведено на рисунку 3.6.

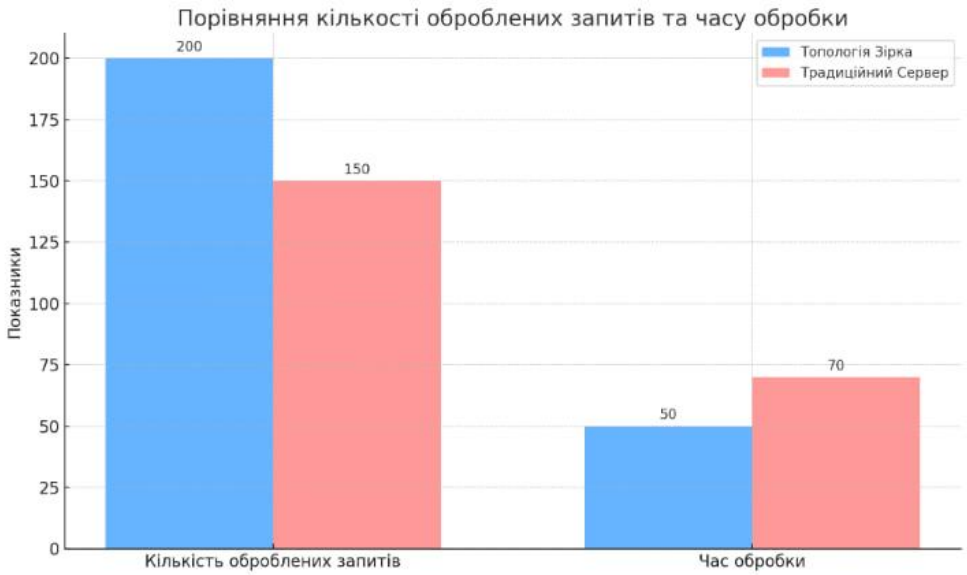


Рисунок 3.6 - Діаграма порівняння кількості оброблених запитів, та середнього часу обробки запиту

Для наочності доцільно провести кілька додаткових експериментів, їх результати зображено на рисунках 3.7 та 3.8.

В експериментах видно, що топологія зірка стабільно обробляє більше запитів порівняно з традиційним сервером:

Експеримент 1: 200 запитів для зірки проти 150 запитів для традиційного сервера.

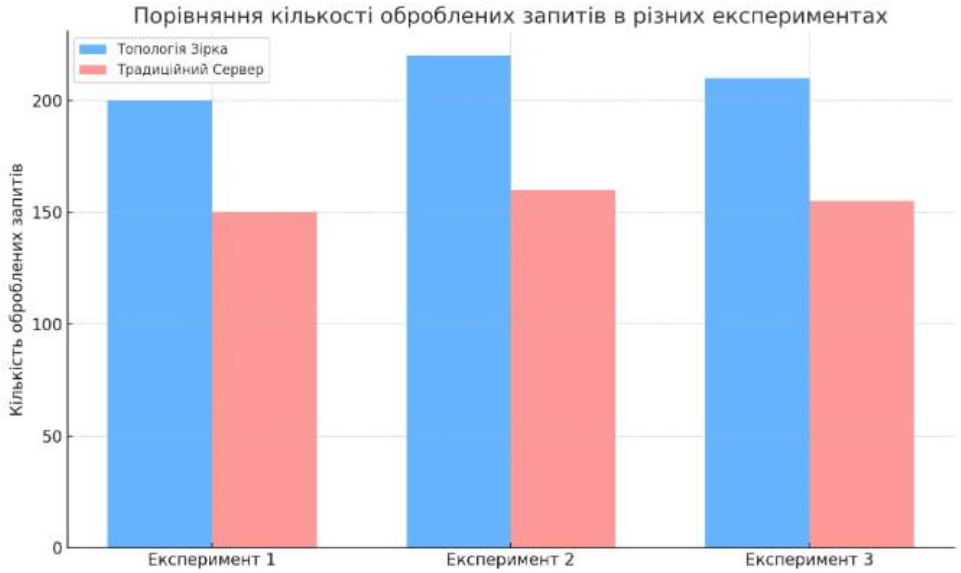


Рисунок 3.7. - Діаграма порівняння кількості оброблених запитів в кількох експериментах

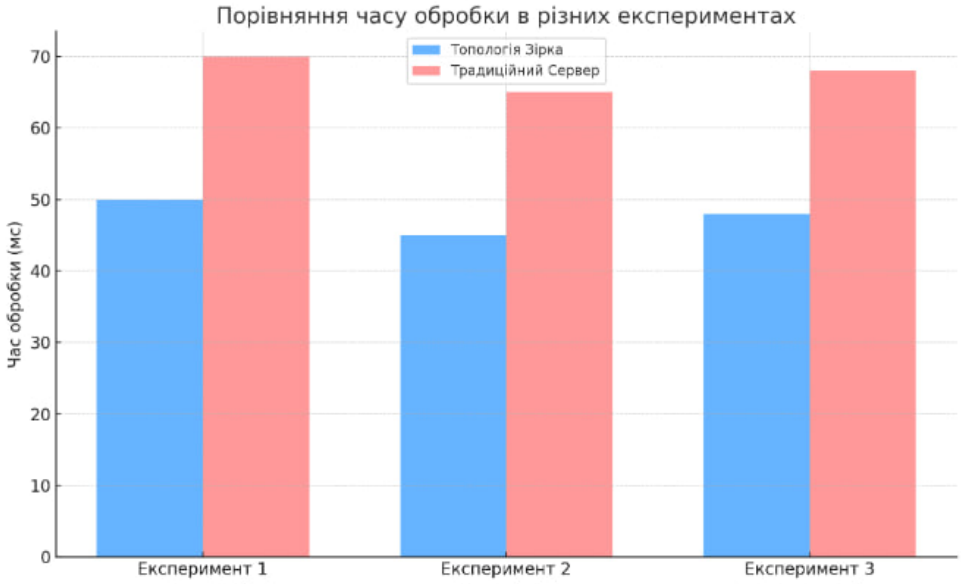


Рисунок 3.8. - Діаграма порівняння часу обробки в кількох експериментах

Експеримент 2: 220 запитів для зірки проти 160 запитів для традиційного сервера.

Експеримент 3: 210 запитів для зірки проти 155 запитів для традиційного сервера.

Також видно, що час обробки запитів для топології зірка є меншим у порівнянні з традиційним сервером:

Експеримент 1: 50 мс для зірки проти 70 мс для традиційного сервера

Експеримент 2: 45 мс для зірки проти 65 мс для традиційного сервера

Експеримент 3: 48 мс для зірки проти 68 мс для традиційного сервера

Висока продуктивність системи з топологією зірка (квітка) забезпечується завдяки концепції “бджілок”.

Концепція "бджілок" у контексті топології "квітка" є цікавою та інноваційною моделлю для управління запитами та розподілом навантаження в мережі.

У цій моделі "бджілки" виступають як автономні агенти, які подорожують між вузлами мережі, збираючи і передаючи інформацію, а також керуючи процесом обробки запитів. Основна ідея полягає в тому, що ці агенти, подібно до справжніх бджіл, збирають "нектар" – у цьому випадку дані про стан вузлів, їхнє навантаження, доступні ресурси та інші параметри.

Бджілки починають свою подорож від центрального вузла, де вони отримують початкову інформацію та завдання. Вони відправляються до периферійних вузлів, де збирають інформацію про поточне навантаження та обробку запитів.

Після збору даних бджілки повертаються до центрального вузла, передаючи зібрану інформацію.

Центральний вузол аналізує ці дані і приймає рішення про те, як найкраще розподілити запити серед вузлів, враховуючи поточну ситуацію.

Крім того, бджілки можуть безпосередньо перенаправляти запити до вузлів, які менш завантажені, на основі зібраної інформації. Такий механізм дозволяє забезпечити більш рівномірний розподіл навантаження та уникнути перевантаження окремих вузлів. Важливою характеристикою цієї моделі є її динамічність і адаптивність. Бджілки постійно оновлюють інформацію про стан

мережі, що дозволяє системі швидко реагувати на зміни і ефективно адаптуватися до нових умов.

Основні переваги використання бджілок у топології "квітка" включають підвищену ефективність обробки запитів, зниження затримок, оптимальне використання ресурсів і підвищення стійкості системи до навантажень.

Однак, така модель також має свої виклики, зокрема потребу в складних алгоритмах для координації дій бджілок та забезпечення їхньої безпеки і коректного функціонування.

Впровадження концепції бджілок у мультикомп'ютерні системи з топологією "квітка" демонструє, як можна використовувати біоінспіровані підходи для підвищення ефективності та продуктивності мережевих систем.

Це відкриває нові можливості для розробки більш складних і адаптивних мережевих інфраструктур, здатних ефективно обробляти великі обсяги запитів у різноманітних умовах.

В будь-якому випадку систему можна оптимізовувати, та покращувати різними способами.

В даному дослідженні неможливо отримати ресурси величезних масштабів, про те в загальній концепції мультикомп’ютерних систем завжди знайдеться можливість для покращення продуктивності, чи збільшення масштабованості системи.

Загалом покращення програмного забезпечення – це безперервний процес, який включає в себе різноманітні підходи і методики для підвищення його ефективності, надійності та зручності використання.

Одним із основних способів покращення є оптимізація коду, яка включає усунення неефективних ділянок, зменшення використання ресурсів і прискорення виконання завдань. Оптимізація може бути досягнута через рефакторинг коду, покращення алгоритмів та структур даних, а також використання сучасних технологій і бібліотек.

Іншим важливим аспектом є підвищення надійності програмного забезпечення через впровадження автоматизованих тестів і використання методологій безперервної інтеграції та розгортання (CI/CD). Це дозволяє швидко виявляти і виправляти помилки, зменшувати ризики збоїв і забезпечувати стабільну роботу програмних продуктів.

Покращення інтерфейсу користувача також відіграє ключову роль. Це включає в себе роботу над зручністю і доступністю програмного забезпечення, що робить його більш привабливим і легким у використанні для кінцевих користувачів.

Важливим є також зворотний зв'язок від користувачів, який допомагає виявляти недоліки і впроваджувати необхідні покращення.

Сучасні аналоги-гіганти, такі як Google, Microsoft і Amazon, значно просунутіші завдяки їхнім величезним ресурсам і досвіду. Вони використовують передові технології машинного навчання і штучного інтелекту для оптимізації своїх продуктів і сервісів.

Ці компанії мають можливість проводити масштабні дослідження і розробки, впроваджувати інновації і швидко адаптуватися до змін ринку.

Крім того, гіганти індустрії активно інвестують у хмарні технології, що дозволяє їм забезпечувати високий рівень доступності і масштабованості своїх сервісів. Вони також використовують складні системи моніторингу і управління, які дозволяють їм швидко реагувати на будь-які проблеми і підтримувати високу продуктивність.

Таким чином, хоча невеликі компанії і незалежні розробники можуть мати обмежені ресурси, вони можуть досягати значного прогресу через фокусування на інноваціях, оптимізації процесів і тісному взаємодії з користувачами. Водночас, сучасні аналоги-гіганти залишаються лідерами завдяки своїм потужним ресурсам, технологічним інноваціям і масштабним операціям, які дозволяють їм постійно вдосконалювати свої продукти і сервіси.

3.3 Висновки

В ході третього розділу було проведено опис алгоритму роботи програмного забезпечення, частково описано деякі моменти роботи коду, а також проведено порівняння ефективності розробленої топології з ефективністю роботи звичайного сервера.

В ході кількох експериментів було визначено, що система показує сеюе доволі продуктивною, в порівнянні зі звичайним сервером.

Результати експериментів відображено за допомогою діаграм, що відображають кількість оброблених запитів, та їх час обробки.

Також було запропоновано способи покращення ефективності роботи мультикомп’ютерних систем в загальному.

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень було розроблено програмне забезпечення, що працює відповідно до топології “квітка” в мультикомп’ютерній системі.

У першому розділі проведено аналіз теоретичної інформації. Враховуючи матеріали, наведені в першому розділі, можна зробити висновок, що мультикомп’ютерні системи сьогодення мають надзвичайно широкий спектр застосування і характеризуються великою варіативністю. Це пов’язано з їхньою здатністю забезпечувати високу продуктивність, гнучкість та надійність у різних галузях, таких як наука, бізнес, індустрія розваг і багато інших. Мультикомп’ютерні системи дозволяють ефективно використовувати ресурси, розподіляти навантаження і забезпечувати безперебійну роботу навіть у складних умовах. Особливої уваги заслуговують топології "квітка" та "зірка", які надають можливість оптимального з'єднання вузлів для забезпечення високої надійності та продуктивності мережі

У межах розділу 2 було здійснено детальний огляд архітектури в мультикомп'ютерних системах загалом, а також розглянуто різні види архітектур та їхні властивості. Це дозволило отримати глибше розуміння структури і функціонування мультикомп'ютерних систем, їхніх переваг та недоліків у різних сценаріях використання. Особливу увагу було приділено аналізу корисного та актуального програмного забезпечення, яке може бути використане для розробки і тестування проміжного програмного забезпечення. Проведений аналіз включав порівняння особливостей різних програмних засобів, що дозволило визначити їхні сильні і слабкі сторони, а також сферу найбільш ефективного застосування.

В ході третього розділу було проведено опис алгоритму роботи програмного забезпечення, частково описано деякі моменти роботи коду, а також проведено порівняння ефективності розробленої топології з ефективністю роботи звичайного сервера.

В ході кількох експериментів було визначено, що система показує сеюе доволі продуктивною, в порівнянні зі звичайним сервером.

Результати експериментів відображено за допомогою діаграм, що відображають кількість оброблених запитів, та їх час обробки.

Також було запропоновано способи покращення ефективності роботи мультикомп’ютерних систем в загальному.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Tang, X., Yao, Y., Yu, Z., & Liu, L. (2023). Multi-computer Communication Reliability Evaluation System Based on 8051 Microcontroller. *Advances in Computer, Signals and Systems*, 7(6), 36-44.

2. Silvano, C., Ielmini, D., Ferrandi, F., Fiorin, L., Curzel, S., Benini, L., ... & Birke, R. (2023). A survey on deep learning hardware accelerators for heterogeneous hpc platforms. *arXiv preprint arXiv*:2306.15552.

3. Lee, J. K., Jamieson, M., Brown, N., & Jesus, R. (2023). Test-driving RISC-V Vector hardware for HPC. *arXiv preprint arXiv*:2304.10319.

4. Mira, D., Pérez-Sánchez, E. J., Borrell, R., & Houzeaux, G. (2023). HPC-enabling technologies for high-fidelity combustion simulations. *Proceedings of the Combustion Institute*, 39(4), 5091-5125.

5. Enes, J., Expósito, R. R., Fuentes, J., Cacheiro, J. L., & Touriño, J. (2023). A pipeline architecture for feature-based unsupervised clustering using multivariate time series from HPC jobs. *Information Fusion*, 93, 1-20.

6. Rahul, P., & Kaarthick, B. (2023). An optimal cluster head and gateway node selection with fault tolerance. *Published in Intelligent Automation & Soft Computing*, 35(2), 1595-1609..

7. Lin, K., Fan, Z., Liu, B., Chen, Y., & Liu, Z. (2023, February). Design of gateway nodes for wireless sensor networks based on microservice architecture. *In Third International Symposium on Computer Engineering and Intelligent Communications (ISCEIC 2022)* (Vol. 12462, pp. 30-36). SPIE.

8. Panse, T., & Panse, P. (2023). An Efficient Gateway Node Selection Method for Clustering in Heterogeneous Mobile Ad-hoc Networks..

9. Bilgili, M., Canpolat, C., Pinar, E., & Sahin, B. (2023). Analysis of heating degree-days (HDD) data using machine learning and conventional time series methods. *Theoretical and Applied Climatology*, 1-20.

10. Muto, R., & Uchimura, Y. (2023). Controller design for HDD benchmark problem using RNN-based reinforcement learning. *IFAC-PapersOnLine*, 56(2), 4424-4429.

11. Seng[1], N. M., Al-Nahari[1], A., Ismail, N. A., & Ahmad, A. (2023). Check for updates Big Data Application on Prediction of HDD Manufacturing Process Performance. *Data Science and Emerging Technologies: Proceedings of DaSET 2022*, 165, 222.

12. Wang, L., Shi, W., Tang, Y., Liu, Z., He, X., Xiao, H., & Yang, Y. (2023). Transfer Learning-Based Lightweight SSD Model for Detection of Pests in Citrus. *Agronomy*, 13(7), 1710.

13. Huo, B., Li, C., Zhang, J., Xue, Y., & Lin, Z. (2023). SAFF-SSD: Self-Attention Combined Feature Fusion-Based SSD for Small Object Detection in Remote Sensing. *Remote Sensing*, 15(12), 3027.

14. Yang, H., Wang, W., Chen, M., Lin, B., He, T., Chen, H., ... & Ouyang, W. (2023). PVT-SSD: Single-Stage 3D Object Detector with Point-Voxel Transformer. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 13476-13487).

15. Cohen, J., Crispim-Junior, C., Chiappa, J. M., & Rodet, L. T. (2024). Industrial object detection with multi-modal SSD: closing the gap between synthetic and real images. *Multimedia Tools and Applications*, 83(4), 12111-12138.

16. Olson, K., Wampler, J., & Keller, E. (2023). Doomed to Repeat with IPv6? Characterization of NAT-centric Security in SOHO Routers. *ACM Computing Surveys*.

17. Nassar, R., Elhajj, I. H., Kayssi, A., & Salam, S. (2023, June). A Generalizable Machine Learning Model for NAT Detection. *In 2023 International Conference on Intelligent Computing, Communication, Networking and Services (ICCNS) (pp. 44-49). IEEE*.

18. Packard, N. (2023). The ARPANET into the Internet: A tale of two networks.

19. Packard, N. (2023). Internet prehistory: ARPANET chronology. *Cogent Social Sciences*, 9(2), 2245237.

20. Mohichehra, M. (2023). FROM ARPANET TO WWW, OR HISTORY OF THE SECRET NETWORK. *BEST SCIENTIFIC RESEARCH-2023*, 2(1), 162-171.

21. Kizi, T. S. G. (2023). Ethernet and Fast Ethernet network architecture. *Best Journal of Innovation in Science, Research and Development*, 175-179.

22. Loveless, A., Phan, L. T. X., Dreslinski, R., & Kasikci, B. (2023, May). PCSPOOF: Compromising the safety of time-triggered ethernet. *In 2023 IEEE Symposium on Security and Privacy (SP)* (pp. 3193-3208). IEEE.

23. Yang, Z., & Zhu, W. (2023). Improvement and Optimization of Vulnerability Detection Methods for Ethernet Smart Contracts. *IEEE Access*.

24. Wen, J., Yang, J., Wang, T., Li, Y., & Lv, Z. (2023). Energy-efficient task allocation for reliable parallel computation of cluster-based wireless sensor network in edge computing. *Digital Communications and Networks*, 9(2), 473-482.

25. Miao, X., Nie, X., Zhang, H., Zhao, T., & Cui, B. (2023). Hetu: A highly efficient automatic parallel distributed deep learning system.

26. Suleman, M. J. (2023). The Use of High-Performance Computing Services in University Settings: A Usability Case Study of the University of Cincinnati's High-Performance Computing Cluster (Doctoral dissertation, University of Cincinnati).

27. Aparecida Silva Camacho, T., Martins do Rosario, V., Oliveira Napoli, O., & Borin, E. (2023). PB3Opt: Profile-based biased Bayesian optimization to select computing clusters on the cloud. *Concurrency and Computation: Practice and Experience*, 35(18), e7540.

28. Grinstein, J. D. (2023). CZI to Build Massive GPU Cluster for Decoding Biology with AI: The Chan Zuckerberg Initiative announced that it will build one of the most powerful high-performance computing systems for non-profit life science research in the world and develop machine learning technology and tools to support scientists solving biology's greatest challenges. *GEN Edge*, 5(1), 629-632.

29. Zhang, D., & Dai, D. Optimizing Resource Management for Machine Learning Workloads in High-Performance Clusters.

30. Duerrwaechter, J., Kuhn, T., Meyer, F., Beck, A., & Munz, C. D. (2023). PoUnce: A framework for automatized uncertainty quantification simulations on high-performance clusters. *Journal of Open Source Software*, 8(82), 4683.

31. Wittig, A., & Wittig, M. (2023). Amazon Web Services in Action: An in-depth guide to AWS. Simon and Schuster.

32. Yang, S., Jin, W., Yu, Y., & Hashim, K. F. (2023). Optimized hadoop map reduce system for strong analytics of cloud big product data on amazon web service. *Information Processing & Management*, 60(3), 103271.

33. Boneder, S. (2023). Evaluation and comparison of the security offerings of the big three cloud service providers Amazon Web Services, Microsoft Azure and Google Cloud Platform (Doctoral dissertation, Technische Hochschule Ingolstadt).

34. Singh, S., Ramkumar, K. R., & Kukkar, A. (2023). Analysis and Implementation of Microsoft Azure Machine Learning Studio Services with Respect to Machine Learning Algorithms. *In Modern Electronics Devices and Communication Systems: Select Proceedings of MEDCOM 2021* (pp. 91-106). Singapore: Springer Nature Singapore.

35. MANZATO, A. Implementing the Zero Trust model through Microsoft Azure Technologies for Enterprise Security.

36. Wicaksono, A. B., & Munadi, R. (2023). Cloud server design for heavy workload gaming computing with Google cloud platform. *International Journal of Electrical & Computer Engineering* (2088-8708), 13(2).

37. Subashini, P., Dhivyaprabha, T. T., Krishnaveni, M., & Jennyfer Susan, M. B. (2023). Smart Intelligent System for Cervix Cancer Image Classification Using Google Cloud Platform. *In Enabling Technologies for Effective Planning and Management in Sustainable Smart Cities* (pp. 245-281). Cham: Springer International Publishing.

38. Gupta, U., & Sharma, R. (2024). Apache Hadoop framework for big data analytics using AI. *In Artificial Intelligence and Blockchain in Industry 4.0* (pp. 130-140). CRC Press.

39. Ibtisum, S., Bazgir, E., Rahman, S. A., & Hossain, S. S. (2023). A comparative analysis of big data processing paradigms: Mapreduce vs. apache spark. *World Journal of Advanced Research and Reviews*, 20(1), 1089-1098.

40. Truong, T. X., Nhu, V. H., Phuong, D. T. N., Nghi, L. T., Hung, N. N., Hoa, P. V., & Bui, D. T. (2023). A New Approach Based on TensorFlow Deep Neural Networks with ADAM Optimizer and GIS for Spatial Prediction of Forest Fire Danger in Tropical Areas. *Remote Sensing*, 15(14), 3458.

41. Pattanayak, S. (2023). Introduction to deep-learning concepts and TensorFlow. *In Pro Deep Learning with TensorFlow 2.0: A Mathematical Approach to Advanced Artificial Intelligence in Python* (pp. 109-197). Berkeley, CA: Apress.

42. Kunduru, A. R. (2023). Blockchain Technology for ERP Systems: A Review. *American Journal of Engineering, Mechanics and Architecture* (2993-2637), 1(7), 56-63.

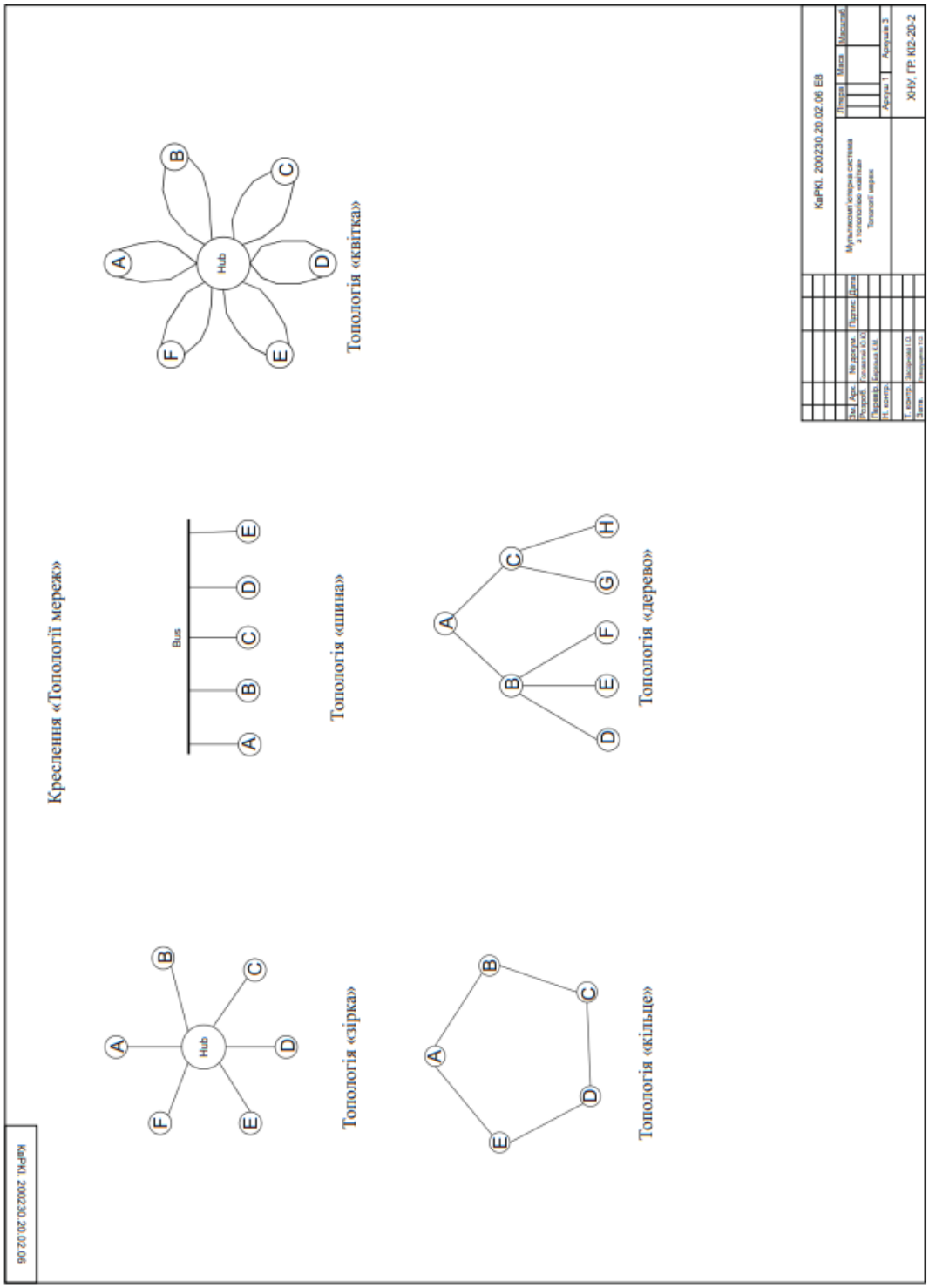
43. Bandara, F., Jayawickrama, U., Subasinghage, M., Olan, F., Alamoudi, H., & Alharthi, M. (2023). Enhancing ERP responsiveness through big data technologies: an empirical investigation. *Information Systems Frontiers*, 1-25.

44. Panjaitan, R. A. W. N., & Purba, M. J. (2023). FORENSIC NETWORK ANALYSIS AND IMPLEMENTATION OF SECURITY ATTACKS ON VIRTUAL PRIVATE SERVERS. *Jurnal Sistem Informasi dan Ilmu Komputer Prima* (JUSIKOM PRIMA), 6(2), 28-34.

Додаток А

(обов’язковий)

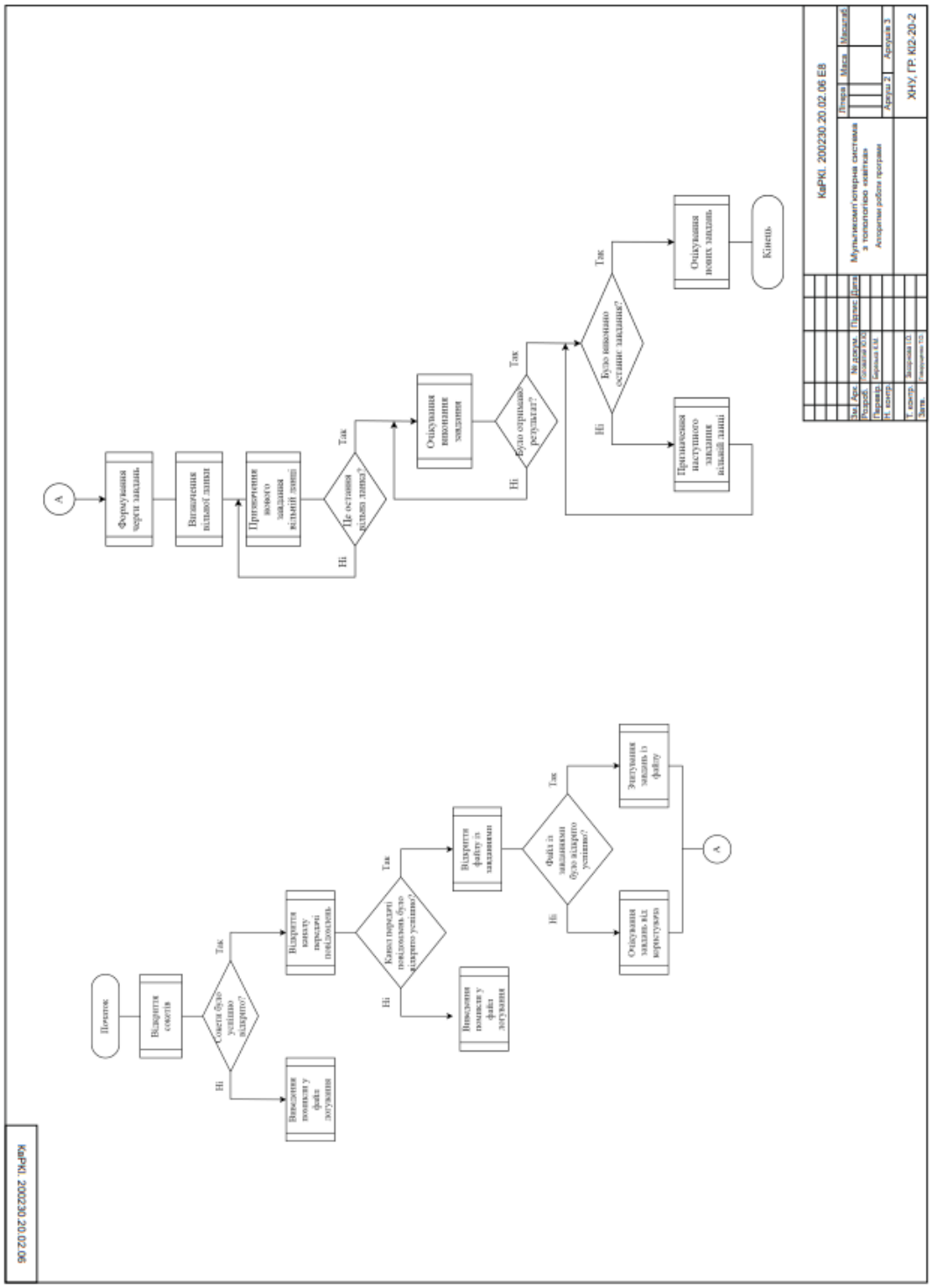
Копія креслення «Топологія мереж»



Додаток Б

(обов’язковий)

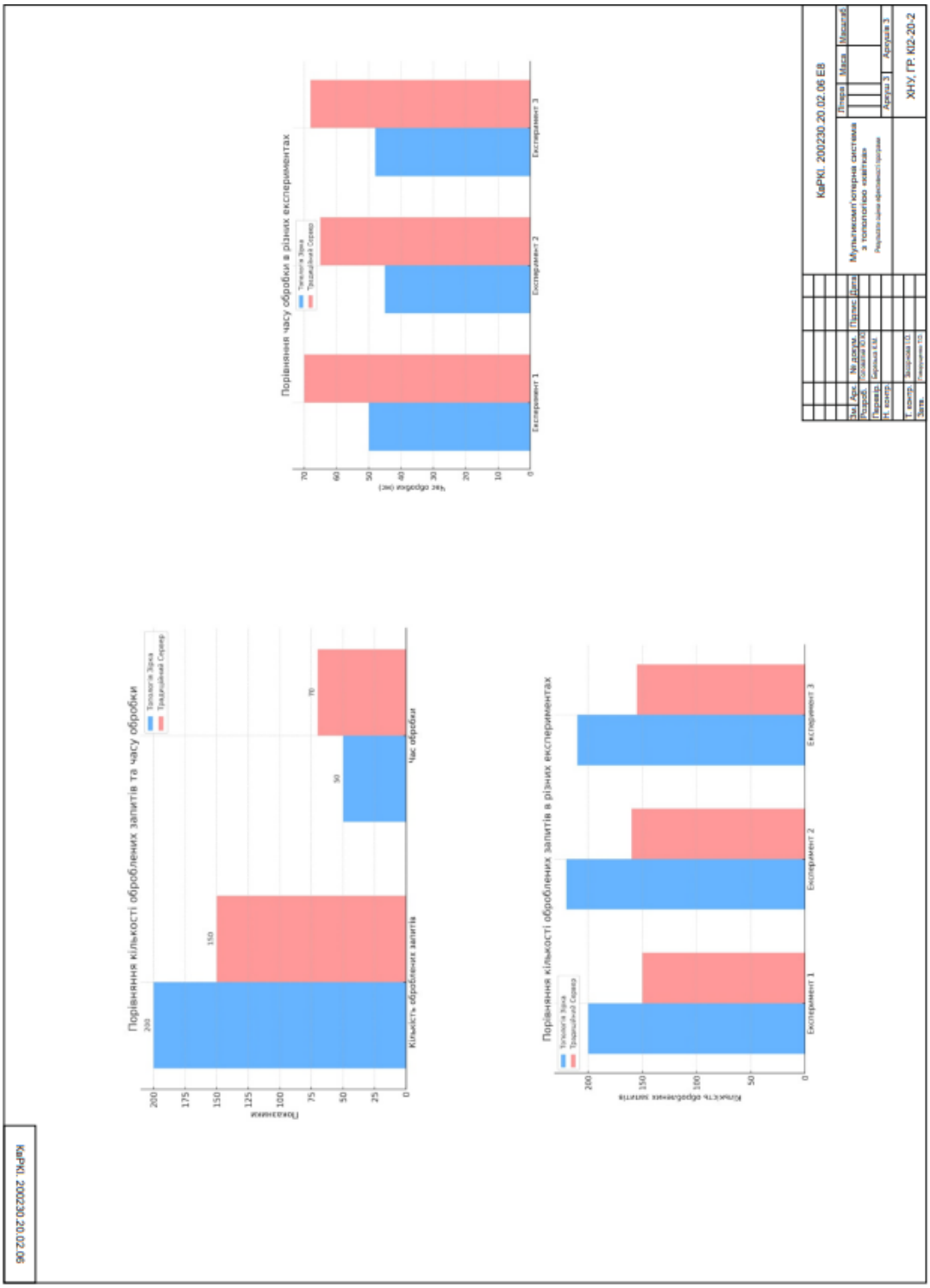
Копія креслення «Алгоритм роботи програми»

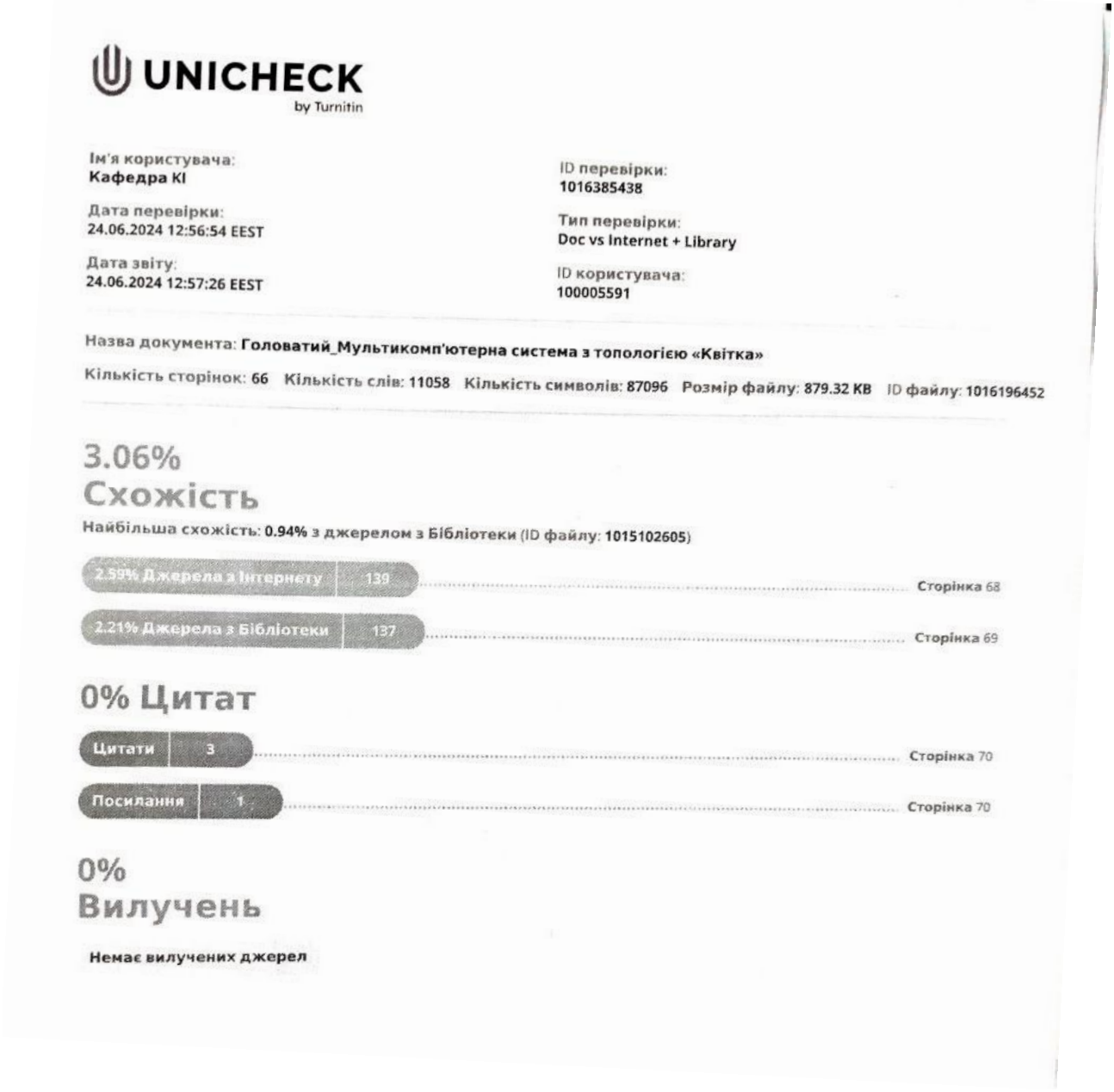


Додаток В

(обов’язковий)

Копія креслення «Оцінка результатів ефективності програми»





UNICHECK
by Turnitin

Ім'я користувача:
Кафедра КІ

Дата перевірки:
24.06.2024 12:56:54 EEST

Дата звіту:
24.06.2024 12:57:26 EEST

ID перевірки:
1016385438

Тип перевірки:
Doc vs Internet + Library

ID користувача:
100005591

Назва документа: **Головатий_Мультикомп'ютерна система з топологією «Квітка»**

Кількість сторінок: **66** Кількість слів: **11058** Кількість символів: **87096** Розмір файлу: **879.32 KB** ID файлу: **1016196452**

3.06%
Схожість

Найбільша схожість: **0.94%** з джерелом з Бібліотеки (ID файлу: **1015102605**)

2.59% Джерела з Інтернету 139 ..... Сторінка 68

2.21% Джерела з Бібліотеки 137 ..... Сторінка 69

0% Цитат

Цитати 3 ..... Сторінка 70

Посилання 1 ..... Сторінка 70

0%
Вилучень

Немає вилучених джерел

Mon Jun 24 11:57:04 EEST 2024, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 6.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 9%**

	Документ		Сумарний збіг по Базі Даних	
ID: 132391 Назва: БКР Мультикомп'ютерна система з топологією «Квітка» Додано в БД: 2024-06-24 Автора: Ю. Ю. Головатий Керівники: К. М. Березька Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	76797	608	5578 (7%)	64 (11%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Головатий Юрій

Тема: Мультикомп'ютерна система з топологією квітка».

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3____ Кількість сторінок записки ___57___

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є проектування мультикомп'ютерної системи з топологією квітка.

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: У першому розділі проведено аналіз теоретичної інформації. У межах розділу 2 було здійснено детальний огляд архітектури в мультикомп'ютерних системах загалом, а також розглянуто різні види архітектур та їхні властивості. Це дозволило отримати глибше розуміння структури і функціонування мультикомп'ютерних систем, їхніх переваг та недоліків у різних сценаріях використання. В ході третього розділу було проведено опис алгоритму роботи програмного забезпечення, частково описано деякі моменти роботи коду, а також проведено порівняння ефективності розробленої топології з ефективністю роботи звичайного сервера. В ході кількох експериментів було визначено, що система показує себе доволі продуктивною, в порівнянні зі звичайним сервером.

Результати експериментів відображено за допомогою діаграм, що відображають кількість оброблених запитів, та їх час обробки.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: недостатня увага моделюванню інших топологія

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) _____

Завідувач кафедри автоматизації, комп'ютерно-інтегрованих технологій та робототехніки, професор, д.т.н. Мартинюк В.В.

"21" 06 2024 р. _____(підпис)

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

Головатого Юрія Юрійовича
ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи КІ2-20-2

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.

8 травня 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: Мультикомп'ютерна система з топологією квітка

Автор: Головатий Юрій Юрійович

Спеціальність: 123– Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Березька Катерина Миколаївна, д.т.н, професор

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, про що свідчить посилання системи на збіг з 139 джерелами на один фрагмент речення;
4) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 3.06% і адресується до 139 першоджерела, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи К. М. Березька

Гарант ОП С.М. Лисенко

Завідувач кафедри КІІС Т. О. Говорущенко