Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи управління розподіленими гетерогенними хмарними системами

(тема)

Виконав:

студент II курсу, групи СПм-22-6

Паронікян П.А.
(прізвище, ініціали)

Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: проф. Волк М.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Паронікяну Папіну Арменовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Методи управління розподіленими гетерогенними хмарними системами

затверджена наказом по університету від “ 01 ” квітня 2024 р. № 257 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 15 червня 2024 р.

3. Вхідні дані до роботи

Моделі хмарних обчислень.

Постачальники хмарних послуг (AWS, GCP, Azure).

Методи оцінки продуктивності хмарних платформ.

4. Перелік питань, що потрібно опрацювати у роботі

Аналіз предметної області.

Моделі та методи управління розподіленими гетерогенними хмарними системами.

Інтеграція методу в хмарне середовище та проведення експериментів.

Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Презентація 12 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області	02.04.24 – 04.04.24	
2	Розробка моделей	05.04.24 – 15.04.24	
3	Реалізація алгоритмів	16.04.24 – 28.04.24	
4	Розробка структури програмних засобів	29.04.24 – 15.05.24	
5	Розробка програмних модулів	16.05.24 – 25.05.24	
6	Оформлення матеріалів кваліфікаційної роботи	26.05.24 – 05.06.24	
7	Подання кваліфікаційної роботи керівникові та	07.06.24 – 12.06.24	
	її попередній захист		
8	Подання кваліфікаційної роботи на рецензу-	13.06.24 – 15.06.24	
	вання		

Дата видачі завдання 01 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Волк М.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 57 с., 7 рис., 3 табл., 1 дод., 60 джерел.

ХМАРНІ СИСТЕМИ, ГЕТЕРОГЕННІ СИСТЕМИ, МАШТАБОВАНІСТЬ, ГНУЧКІСТЬ, МОДЕЛЮВАННЯ, ІНТЕРНЕТ РЕЧЕЙ.

Інтернет речей (IoTs) – це технологія, яка підключає сенсорні пристрої до Інтернету для забезпечення розумнішого та інтелектуальнішого управління віддаленими системами. Сьогодні багато галузей промисловості використовують різноманітні пристрої IoT для створення розумних та інтелектуальних середовищ. Як правило, таки системи об'єднують різні апаратні та програмні платформи та формують гетерогенні інформаційні системи. Однак раптове зростання попиту створило серйозну проблему для підключень IoT, відому як масштабованість. Масштабованість означає збільшення та розширення кількості підключених до Інтернету пристроїв для певної програми. Щоб вирішити цю проблему, пропонується імітація горизонтального масштабування для досягнення швидшого та ефективнішого масштабування для пристроїв IoT. Було досліджено різні методи горизонтального масштабування та запропонована модифікація ланцюга Маркова для моделювання масштабування. Дані, отримані під час моделювання використані для оптимізації масштабування, візуалізованого процесом ланцюга Маркова.

Метою роботи було забезпечення гнучкого характеру горизонтальної масштабованості для підключення різних пристроїв і ресурсів IoT за потребою. Запропонована оптимізацію горизонтальної масштабованості з вертикальною масштабованістю, яка має вбудовану функцію еластичності з урахуванням вартості, швидкість обслуговування та передачі даних.

ABSTRACT

Master’s thesis: 57 pages, 7 figures, 3 tables, 1 appendice, 60 sources.

CLOUD SYSTEMS, HETEROGENEOUS SYSTEMS, SCALING, FLEXIBILITY, SIMULATION, INTERNET OF THINGS.

The Internet of Things (IoTs) is a technology that connects sensor devices to the Internet to enable smarter and more intelligent control of remote systems. Today, many industries are using a variety of IoT devices to create smart and intelligent environments. As a rule, such systems combine various hardware and software platforms and form heterogeneous information systems. However, the sudden increase in demand has created a major challenge for IoT connections known as scalability. Scalability refers to increasing and expanding the number of Internet-connected devices for a given application. To solve this problem, horizontal scaling simulation is proposed to achieve faster and more efficient scaling for IoT devices. Different methods of horizontal scaling were investigated and a modified Markov chain was proposed for modeling scaling. The data obtained during the simulation are used to optimize the scaling visualized by the Markov chain process.

The goal of the work was to provide the flexible nature of horizontal scalability to connect different IoT devices and resources as needed. Horizontal scalability optimization with vertical scalability is proposed, which has a built-in elasticity function considering cost, service speed and data transfer.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ІТ – інформаційні технології

ПЗ – програмне забезпечення

ПК – персональний комп'ютер

ЦП – центральний процесор

API – інтерфейс програмування додатків (англ., Application Programming Interface)

AWS – Amazon Web Services

CREIM – Coordinate-based efficient indexing mechanism

HAN – домашня мережа (англ., Home Area Network)

HPC – високопродуктивні обчислення (англ., High Performance Computing)

GA – генетичний алгоритм (англ., Genetic algorithm)

IaaS – інфраструктура як послуга (англ., Infrastructure as a Service)

IoT – інтернет речей (англ., Internet of Things)

HTTP – протокол передачі гіпертексту (англ., HyperText Transfer Protocol)

NFV – віртуалізація функцій мережі (англ., Network Function Virtualization)

PSO – метод рою часток (англ., Particle Swarm Optimization)

PaaS – платформа як послуга (англ., Platform as a service)

SaaS – програмне забезпечення як послуга (англ., Software as a service)

WAN – глобальна комп’ютерна мережа (англ., Wide Area Network)

WLAN – безпровідна локальна мережа (англ., Wireless Local Area Network)

WSN – бездротова сенсорна мережа (англ., Wireless Sensor Networks)

ВСТУП

Інтернет речей (IoT) – це технологія, яка швидко розвивається, інтегрується з різними типами технологій, такими як бездротові сенсорні мережі, вбудовані пристрої та сканери RFID, для виконання програм у режимі реального часу, які покращують комфорт і полегшують життя людини. Інтернет речей стає все більш популярним у різних галузях, від сільського господарства до промислових застосувань. Він включає широкий спектр сенсорних пристроїв, які можна підключити до реального світу, щоб приймати розумніші та розумніші рішення. Згідно з останніми оцінками, до 2025 року кількість підключених і взаємодіючих пристроїв досягне близько 75 мільярдів [1,2]. Технологія IoT використовується для вдосконалення різних сфер, таких як розумні міста, розумний дім, автомобільна автоматизація, керування здоров'єм, розумне сільське господарство та розумне управління трафіком. Метою розвитку сучасних інформаційних технгологій є намагання зробити швидкий розвиток технологій доступним для кожного куточка суспільства.

Реалізація додатків IoT значною мірою залежить від сенсорних пристроїв для збору та передачі даних. Ці сенсорні пристрої підключаються до об'єктів реального світу, збирають дані та надсилають їх у хмару через комунікаційні мережі. У хмарі дані обробляються, а результати роботи програм у реальному часі відображаються кінцевому користувачеві для подальшого аналізу та прийняття рішень [3–5]. Оскільки додатки IoT продовжують розширюватися, кількість пристроїв, ресурсів та пов'язаної з ними інфраструктури збільшується, що створює проблеми з масштабованістю. Одним із рішень цієї проблеми є використання периферійних обчислень, де шлюзи використовуються для зв'язку з пристроями IoT через дротові або бездротові з'єднання з метою зберігання, доступу та обробки даних.

Метою роботи Метою кваліфікаційної роботи є підвищення ефективності управління розподіленими хмарними системами за рахунок розробки методу керування масштабованістю.

Для досягнення цієї мети треба вирішити наступні задачі:

- дослідити методи та алгоритми горизонтальної та вертикальної масштабованості;
- обрати механізми керування масштабованістю та критерії оцінювання її ефективності;
- розробити метод управління масштабованістю хмарної системи;
- цінити ефективність запропонованих рішень.

Об'єктом досліджень є процес управління розподіленими гетерогенними хмарними системами.

Предмет досліджень: метод керування масштабованістю хмарних систем.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Вертикальна та горизонтальна масштабованість хмарних систем

Масштабованість додатків IoT у хмарі можна розділити на дві категорії: вертикальна масштабованість і горизонтальна масштабованість [6] (рисунок 1.1).

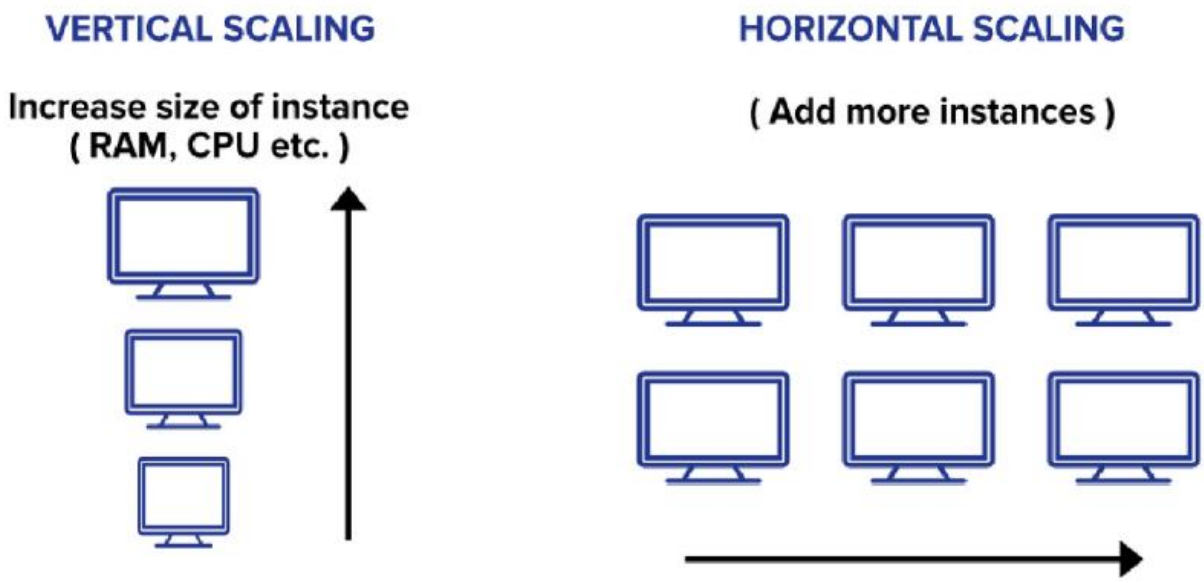

Рисунок 1.1 – Вертикальна та горизонтальна масштабованість

Вертикальна масштабованість означає додавання більше серверів до хмари. Однак недоліком цього підходу є те, що кількість сенсорних пристроїв, шлюзів, серверів, обчислювальної потужності та пристроїв зберігання даних, які можна додати до програми IoTs, обмежена. Якщо ці пристрої буде додано за межі цього ліміту, пакети данних від них будуть відкидатися, що призведе до збою ресурсів та зниження продуктивності, а також збільшить витрати.

В роботі досліджєно оптимізацію горизонтальної масштабованості в хмарних обчисленнях для програм IoT. Із зростаючою кількістю пристроїв IoT і збільшенням обсягу даних, які вони генерують, виникає потреба в ефективному управлінні ресурсами та продуктивністю додатків. Горизонтальна масштабованість — це рішення, яке передбачає додавання додаткових серверів для обробки збільшеного робочого навантаження. У той же час моделювання хмарних систем є алгоритмом оптимізації, який може допомогти знайти оптимальну конфігурацію ресурсів для заданого робочого навантаження. Щоб повністю зрозуміти важливість нашої дослідницької проблеми та запропонованого рішення, важливо надати більш детальну основу щодо концепцій IoT та масштабованості.

1.2 Технології масштабування в контексті IoT

Технологія масштабування, яка швидко розвивається, інтегрована з різними типами технологій, такими як бездротові сенсорні мережі, вбудовані пристрої та сканери RFID, для виконання додатків у реальному часі, які покращують комфорт і полегшують життя людини. Масштабованість є ключовою проблемою, оскільки вони потребують ефективного управління ресурсами, щоб впоратися зі збільшеним навантаженням.

В роботі акцент зроблено на горизонтальній масштабованості, яка передбачає додавання додаткових серверів для обробки збільшеного навантаження. Ми надаємо вичерпний огляд концепції та пояснюємо, як вона може допомогти вирішити проблеми управління ресурсами та продуктивності додатків у системах IoT. Ми також представляємо алгоритм оптимізації та моделювання і пояснюємо, як їх можна використовувати для пошуку оптимальної конфігурації ресурсів для даного робочого навантаження.

Горизонтальна масштабованість у хмарних обчисленнях означає здатність системи справлятися зі зростаючим робочим навантаженням,

додаючи більше ресурсів, наприклад серверів, а не оновлюючи один сервер. Це спосіб масштабувати кількість серверів або пристроїв у розподіленій системі, а не збільшувати ресурси окремих серверів. Цей підхід дозволяє збільшити потужність і продуктивність шляхом додавання додаткових ресурсів до системи. Це більш гнучкий підхід до додавання ресурсів і адаптації системи, оскільки він дозволяє додавати нові ресурси та пристрої за потреби. Це досягається в хмарних обчисленнях шляхом додавання більшої кількості віртуальних машин або екземплярів служби. Це можна зробити за допомогою балансувальників навантаження та груп автоматичного масштабування.

Прикладом горизонтальної масштабованості в хмарних обчисленнях може бути веб-додаток, що працює на кластері віртуальних машин. Коли кількість користувачів, які отримують доступ до програми, збільшується, до кластера можна додати більше віртуальних машин для обробки збільшеного трафіку. Це дозволяє програмі продовжувати безперебійно працювати без перерв і без необхідності вносити зміни в саму програму.

Горизонтальна масштабованість є важливим аспектом хмарних обчислень, оскільки вона дозволяє системі обробляти зростаючий обсяг трафіку та робоче навантаження без перерв. Це може бути економічно ефективним рішенням для компаній, оскільки воно дозволяє їм платити за необхідні ресурси в той момент, коли вони їм потрібні, замість того, щоб надавати ресурси надмірно.

Однією з головних причин вибору горизонтальної масштабованості перед вертикальною є те, що вона забезпечує кращу відмовостійкість. У горизонтальній масштабованості, якщо один сервер або пристрій виходить з ладу, навантаження можна розподілити між іншими серверами або пристроями у системі, що означає, що система в цілому не вимкнеться. Це дозволяє створювати більш надійні та надійні додатки Інтернету речей, які можуть працювати зі збільшеною кількістю пристроїв і витонченіше справлятися зі збоями.

Крім того, горизонтальна масштабованість також дозволяє легше розвивати систему, оскільки додаванням нових ресурсів і пристроїв набагато легше керувати, ніж у випадку вертикального масштабування. Це також може сприяти кращому використанню ресурсів, оскільки система може обробляти більшу кількість запитів з тими самими ресурсами, і її можна підсумувати таким чином:

- відстеження використання ресурсів системи та визначення, коли поточні ресурси наближаються до ліміту;

- додавання нових ресурсів (наприклад, додаткових серверів або екземплярів) до системи, щоб збільшити її потужність;

- налаштування балансування навантаження для рівномірного розподілу вхідних запитів між новими та наявними ресурсами;

- за потреби повторення згаданих висче кроків, щоб продовжити горизонтальне масштабування системи зі збільшенням робочого навантаження.

У запропонованій нами методології ми зосереджуємось на горизонтальній масштабованості, оскільки вона пропонує переваги з точки зору збільшення кількості пристроїв IoT, шлюзів, серверів та інших ресурсів. Це збільшує гнучкість адаптації до майбутніх вимог і покращує якість додатків IoT. Горизонтальна масштабованість фокусується на балансуванні навантаження мережі для рівномірного розподілу даних між різними вузлами [3,7].

Однією з головних переваг горизонтальної масштабованості є те, що вона стійка до відмов. Якщо деякі вузли або ресурси виходять з ладу, вся система не вимикається. Натомість інші ресурси можуть ефективно керувати всіма підключеними пристроями в мережі IoT [8,9]. Це дозволяє створювати більш надійні додатки Інтернету речей, які можуть працювати з великою кількістю пристроїв та більш надійно.

Існує кілька методів досягнення горизонтальної масштабованості в хмарних обчисленнях:

- балансування навантаження: цей метод розподіляє вхідні запити між декількома серверами, щоб гарантувати, що жоден сервер не буде перевантажений; це можна зробити за допомогою програмних або апаратних балансувальників навантаження;

- автоматичне масштабування: цей метод автоматично додає або видаляє ресурси (наприклад, сервери чи екземпляри) із системи на основі поточного робочого навантаження, що гарантує системі постійне забезпечення ресурсами, які необхідні для обробки вхідних запитів;

- кластеризація: цей метод об'єднує кілька серверів разом, щоб діяти як єдине ціле, що дозволяє системі обробляти збільшений трафік шляхом додавання додаткових серверів до кластера;

- контейнерізація та оркестровка: цей метод використовує контейнери та інструменти оркестровки контейнерів, такі як Kubernetes, Docker Swarm або Mesos, для розгортання та масштабування програм у хмарному середовищі;

- безсерверні обчислення: цей метод дозволяє розробникам створювати та запускати програми та служби без необхідності керувати основною інфраструктурою, що робиться шляхом абстрагування основної інфраструктури та надання можливості хмарному провайдеру керувати масштабуванням і наданням ресурсів;

- мікросервіси: цей метод розбиває монолітну програму на менші незалежні служби, які можна розгортати, масштабувати та керувати ними незалежно.

У нашій пропозиції ми зосередимося на кількох типах підходів до кластеризації, які доступні в хмарних обчисленнях, таких як активна-пасивна кластеризація, активна-активна кластеризація та N+1 кластеризація.

Спочатку ми відповідним чином моделюємо нашу проблему, а потім застосовуємо техніку оптимізації на основі SA, щоб архівувати горизонтальне масштабування.

Активно-пасивна кластеризація: у цьому підході є основний вузол, який обробляє всі вхідні запити, і вторинний вузол, який стоїть як резервний. Якщо основний вузол виходить з ладу, вторинний вузол бере на себе роботу.

Активно-активна кластеризація: у цьому підході декілька вузлів обробляють вхідні запити одночасно, забезпечуючи балансування навантаження та можливості відновлення після відмови.

Кластеризація N+1: у цьому підході система розроблена з більшою кількістю вузлів, ніж потрібно для обробки поточного робочого навантаження, з додатковими вузлами, які діють як резервні на випадок збою.

1.3 Аналіз наукових праць в галузі масштабування хмарних систем

Данна робота реалізує експериментальний підхід до горизонтальної масштабованості в реальному часі з вбудованою функцією еластичності для додавання більшої кількості пристроїв на основі IoT відповідно до розширення для конкретної мережі. У вертикальній масштабованості буде обмеження на додавання пристроїв IoTs і мережевих ресурсів для розробки конкретної програми IoTs. Якщо розробка є дуже великою та в майбутньому вимагає додавання більше пристроїв, у такому випадку краще масштабувати за допомогою горизонтальної масштабованості. Проаналізуємо література з досліджень віртуальної масштабованості та горизонтальної масштабованості.

Раззак, Мірза Абдур та ін. [10] зосереджуються на огляді характеру масштабованості розробки додатків IoT в університетському містечку. Середній час роботи сервера IoTs збільшився, коли робоче навантаження дуже високе. У певний момент середній час виконання скорочується при використанні вертикальної масштабованості. Робоче навантаження на віртуальний сервер збільшується, у той же час середній час виконання віртуального сервера IoTs також збільшиться, і він стане вузьким місцем.

У той час як за допомогою горизонтального масштабування існує багато віртуальних серверів IoT для ефективного керування балансуванням робочого навантаження різних завдань. У цій статті автори описують важливість горизонтального масштабування, але не використовують експериментальний підхід до виділення функції горизонтального масштабування. Фахмі, Хафіжуддін Зул та ін. [11] автори запропонували вертикальну масштабованість на основі NFV (віртуалізація мережевих функцій) відповідно до навантаження ЦП для фрагментів IoT. Автори оцінюють продуктивність вертикальної масштабованості, порівнюючи з двома системами, віртуальну масштабованість, за якою слідує горизонтальна масштабованість. Інша система реалізована лише за допомогою горизонтальної масштабованості. Метою авторів є експериментальний підхід до віртуальної масштабованості з наступною горизонтальною масштабованістю для отримання нижчого коефіцієнту використання ЦП, нижчого енергоспоживання, достатнього часу відгуку, а також пропускної здатності. Відповідно до внеску авторів у своїй статті, можна зробити висновок про неефективність віртуальної масштабованості, якщо порівнювати з горизонтальною масштабованістю щодо використання процесора та пропускної здатності для додатків IoT.

Система, розроблена за допомогою вертикальної масштабованості разом із горизонтальною масштабованістю, може продемонструвати можливість нижчого використання процесора та нижчого енергоспоживання. Це показує, що за допомогою функції лише горизонтальної масштабованості можна знизити використання ЦП і енергоспоживання, а не лише віртуальну масштабованість.

Гусєв, Мар'ян та ін. [12] представили масштабовані обчислення для забезпечення пристроїв, а також показали, що менше обчислень дозволяє виконувати функції поблизу пристроїв, і вони будуть знаходитися ближче до користувача, що забезпечує продажи низькорівневої інфраструктури та містить різнорідні пристрої IoT.

Безсерверні обчислення – це сучасне технологічне рішення, де хмарний провайдер керує ресурсами всієї обчислювальної інфраструктури, а користувач виконує роль утилізатора, має обробляти та використовувати ресурси. Автори вводять горизонтальну масштабованість для розподіленого виконання, а централізований спосіб включає вертикальну масштабованість для курування вниз. Утиліта горизонтальної масштабованості виконує розподіл різноманітних функцій смарт-пристроїв для призначення інтелектуальним пристроям. Тоді як утиліта централізованої масштабованості виконується головним вузлом, який контролює спеціальну мережу (тобто не покладається на існуючу інфраструктуру) розумних пристроїв для використання функцій обробки.

Підхід без пристроїв є ідеєю для розумних пристроїв на основі IoT. Автори націлені на масштабованість для безапаратного підходу без пристроїв, а також моделі без речей, щоб створити сучасну платформу, де провайдери керують усіма ресурсами для надання кращих послуг користувачам, тоді як користувачі можуть ефективно використовувати ці ресурси. Автори описують модель підходу без пристроїв за допомогою горизонтальної масштабованості, але не пропонують жодного експериментального підходу.

Брокельсбі, Вільям і Рудра Дутта та ін. [13] запропонували метод безпеки в мережі кампусу шляхом аналізу мережі кампусу та трафіку даних, пропонуючи різні архітектурні вдосконалення для цих моделей апаратного забезпечення та поєднання апаратного та програмного забезпечення для ефективного забезпечення безпеки. Автори визначають і розробляють шаблони безпеки для потоків трафіку між хостами та схід-захід у мережах університетського містечка в різних будівлях для розробки гібридної архітектури для покращення кіберінфраструктури та використання горизонтального підходу для контролю зловмисного програмного забезпечення та підвищення мінімальних накладних витрат на продуктивність за різних типи схем руху для забезпечення безпеки кампусу.

Тан, Сонгтао та ін. [14] запропонували скоординований механізм ефективного індексування (CREIM) для систем Інтернету речей (IoT) у гетерогенному периферійному обчислювальному середовищі. Ці периферійні вузли розгортають сусідні пристрої Інтернету речей (IoT), які мають незбалансовані ресурси як обчислення, так і пам’ять. Запропонований CREIM використовує віртуальний 2- вимірний простір (2D простір) для відображення крайових вузлів. CREIM досягає балансування навантаження на крайових вузлах, маючи різнорідні функції.

Ці методи індексування працюють із горизонтальним масштабуванням крайових вузлів. Включно з CREIM із швидкістю одного стрибка накладання для забезпечення отримання даних із низькою затримкою для периферійних програм. CREIM адаптує гетерогенні крайові вузли та зменшує вартість збільшення та зменшення крайових вузлів. Автори досягли балансування навантаження та розподілу ресурсів, а також покращили адаптацію динамічної природи вузлів для зберігання та пошуку даних. Han, Weixia, Sun Li та ін. [15] запропонував методику розподілу ресурсів на основі IoT для освітнього містечка. IoT використовується для налагодження комунікації для спільного використання ресурсів коледжу. Алгоритм MCTS використовується для пошуку освітніх ресурсів коледжу. Освітні ресурси коледжів, університетів алгоритм MCTS використовував для встановлення зв’язку відображення між ресурсами та розподілом обчислювальних вузлів і додавання вузлів за допомогою горизонтального масштабування в кампусі.

Альтвоян, Вафа та Ібрагім С. Алсукайті. та ін. [16] запропонували нову архітектуру IoT для розгляду ефективної інтегрованої системи Інтернету речей (IoT) в університетському містечку. Ця система архітектури IoTs покращує безпеку, швидкість реагування та підключення до IoT. Запропонована архітектура забезпечує модульну та масштабовану конструкцію з шістьма рівнями архітектури для забезпечення підвищення безпеки цієї архітектури. Запропонована архітектура IoT, що має високу надійність даних можливість керування, високу безпека, швидкість

реагування та особливості горизонтальної масштабованості для додавання більшої кількості пристроїв відповідно до зростання застосування IoT в університетському містечку.

1.4 IoT як представник гетерогенних хмарних систем

IoT – це широкосмугова мережа, яка використовує стандартні протоколи зв’язку. Основним поняттям IoT є широке існування об’єктів, які можна вимірювати та робити висновки, а також здатність змінювати ситуацію у відповідь на отримані дані. Відповідно, IoT розширюється за рахунок розширення кількох речей і комунікаційного обладнання.

Речі в IoT включають розумне обладнання, таке як мобільні телефони та інші засоби, розумні будинки, датчики температури, інші прилади, які можуть співпрацювати для досягнення спільної мети.

Головною характеристикою IoT є його вплив на життя споживачів. У концепції IoT, оскільки вартість кабелю для мільйонів датчиків є високою, зв’язок між датчиками має бути бездротовим. Стандартний зв'язок малої потужності підходить для з'єднання багатьох пристроїв. Відповідно до розташування та відстані покриття деякі мережі представлені таким чином:

- домашні мережі (HAN), які використовують стандарти малого радіусу дії, такі як ZigBee, Dash7 і Wi-Fi; усі компоненти моніторингу та керування в будинку з’єднані через мережу;
- глобальні мережі (WAN) забезпечують зв’язок між клієнтами та розподільними службами, які вимагають значно ширшого покриття, ніж HAN, і для впровадження потребують оптоволоконного кабелю або широкосмугового бездротового зв’язку, наприклад 5G і LTE;
- польові мережі, які використовуються для зв'язку між клієнтами та підстанціями.

В IoT виконуються дві задачі, включаючи визначення й обробку даних, але вони не об’єднані з точки зору бездротової сенсорної мережі (WSN).

Уніфікованими рішеннями є Speakthing та iOBridge. Speakthing – це аналітична платформа IoT для збору, візуалізації та аналізу даних у хмарі, і подальшого аналізу даних за допомогою, наприклад, кодування MATLAB. Навпаки, iOBridge має власні апаратні модулі, підключені до хмари, до яких можна отримати доступ через веб-інтерфейси, а зібрані дані можна агрегувати в інші веб-сервіси. Примітно, що хмара дуже важлива в розумних містах для зберігання та обробки даних. Розглянемо деякі технологія, пов’язані з IoT.

Радіочастотна ідентифікація (RFID) RFID, включаючи зчитувачі та мітки, має життєво важливим компонентом IoT. Застосовуючи технології для кожної пов’язаної речі, здійснюючи їх автоматичну ідентифікацію та присвоєння єдиної цифрової ідентичності будь-якій із речей, можливо включити мережу, пов’язану з цифровою інформацією та послугами. RFID надає деякі програми в інтелектуальних мережах, включаючи відстеження та локалізацію об’єктів, додатки для охорони здоров’я, паркування та управління активами. Кожна мітка може бути датчиком, оскільки вони містять не лише дані, які записуються вручну, але й фіксують дані, наприклад інформацію про навколишнє середовище.

Комунікація ближнього поля (NFC) використовується для двонаправленого зв’язку на короткій відстані, особливо в смартфонах. Цей діапазон зазвичай включає сантиметровий діапазон. Застосування NFC у смартфонах дозволяє використовувати його також у розумних містах. Одна з його програм включає використання смартфонів із NFC як гаманця, що дозволяє нам використовувати смартфони як особисті картки, такі як банківська картка, ідентифікаційна картка, картка громадського транспорту, картка контролю доступу. Крім того, оскільки NFC є двонаправленим, його можна використовувати для обміну даними між пристроями, мультимедіа та документів. Розмістивши NFC у стратегічному місці вдома та забезпечивши інтерфейс із центральним контролером, можна змінювати статус об’єктів,

перевіряючи розташування, наприклад, увімкнути Wi-Fi, коли користувач повертається додому.

Низька швидкість бездротової персональної мережі (LWPAN) відноситься до радіотехнологій малого радіусу дії, які охоплюють великі відстані до 10–15 км. Енергоспоживання цієї технології надзвичайно низьке, а термін служби батареї становить близько 10 років. Відповідно до стандарту IEEE 802.15.4, він забезпечує низьку вартість і низьку швидкість зв'язку для сенсорних мереж. Він має два найнижчі рівні протоколів, включаючи фізичний і середній рівень доступу, окрім протоколів верхнього рівня, включаючи 6LoWPAN і ZigBee [14].

У сенсорних вузлах ZigBee використовується як малопотужна та недорога технологія бездротового зв'язку. Вони засновані на стандарті IEEE 802.15.4 і підходить для створення бездротових персональних мереж (WPAN), таких як домашня автоматизація, медичні пристрої та інші малопотужні та малосмугові мережі. Деякі з його застосувань включають бездротові вимикачі світла, електролічильники та системи керування дорожнім рухом. ZigBee підходить для обмежених діапазонів, охоплення регіону міста та підтримки мільярдів пристроїв. У мережі на основі ZigBee визначено механізм передачі пакетів IPv6. Для застосування ZigBee зазвичай потрібне додаткове обладнання, яке включає координатор, маршрутизатор і кінцеві пристрої ZigBee.

Розглянемо деякі особливості впровадження розумних міст на основі IoT.

Коли вся інформація збирається й оцінюється на одній платформі IoT, система може протистояти численним атакам, таким як міжсайтовий скриптинг. Крім того, система може бути протистояти значним уразливостям. Багатокористувацький доступ також може призвести до проблем безпеки та призвести до витоку даних. Тому міста повинні вжити серйозних заходів для забезпечення конфіденційності та безпеки даних громадян. Без цієї гарантії громадяни не можуть довіряти уряду, і збір

інформації буде складним. Усі системи мають бути стійкими до кібератак, особливо критична інфраструктура, як-от розумні лічильники. Як наслідок, для успішного впровадження IoT міста повинні поставити конфіденційність і безпеку як головний пріоритет.

Одним з основних боків системи IoT є гетерогенність. Системи IoT зазвичай розробляються з конкретними та визначеними рішеннями, у яких кожен елемент системи приєднаний до особливого прикладного контексту. На цій основі органи влади мають вивчити свої цільові сценарії, визначити необхідне апаратне/програмне забезпечення, а потім об’єднати ці гетерогенні підсистеми. Забезпечення таких субструктур і закупівля належної схеми співпраці між ними справді є серйозною місією для систем IoT.

Системи на основі IoT викликають певні проблеми з надійністю. Наприклад, через мобільність автомобілів взаємозв'язок між ними не дуже надійний. Більше того, участь величезної кількості інтелектуальних технологій призведе до певних проблем із надійністю, зокрема щодо їх відмови.

Кілька визначених сценаріїв потребують взаємодії між величезною кількістю розподілених пристроїв, які, ймовірно, будуть вбудовані в глобальне середовище. Система IoT забезпечує належну платформу, яка здатна аналізувати та збирати інформацію, отриману з різних пристроїв. Однак ці великомасштабні дані потребують належного зберігання та обчислювальних можливостей, оскільки вони збираються з високою швидкістю, що призводить до звичайних проблем, з якими складніше впоратися. Крім того, розподіл пристроїв IoT може впливати на дії моніторингу, оскільки пристрої мають мати справу із затримками, пов’язаними з динамікою та підключенням.

Система IoT, ймовірно, є послугою відповідно до наданих користувачами даних. Для таких умов постачальники послуг повинні базуватися на різних місцевих і міжнародних правилах. Подібним чином

заявники стикаються з достатніми стимулами для відвідування певного сценарію та збору даних.

Враховуючи близько 50 мільярдів пристроїв, безперечно важливо звернути увагу на передачу, зберігання та відкликання даних, а також проаналізувати великий обсяг інформації, створеної ними.

Зрозуміло, що підструктури IoT будуть одними з цих важливих джерел великих даних. У проблемах з великими даними виділено три основні характеристики, що складаються з числа, швидкості та дисперсії. Отже, інформація інтелектуального лічильника отримується відповідно до цих специфікацій.

Сенсорні мережі можна розглядати як чудову технологію для забезпечення IoT [8]. Вони можуть формувати світ, надаючи можливості вимірювання, висновків і розуміння екологічних індексів. Сучасний розвиток і вдосконалення технологій забезпечили ефективні та дешеві пристрої для застосування у великомасштабному дистанційному зондуванні. Крім того, смартфони містять різні типи датчиків і, як наслідок, вони дають змогу використовувати різні види мобільних пристроїв у різних сферах IoT. Для цього головною складною дією може бути те, як обробити великомасштабну інформацію датчиків щодо енергетичних і мережевих обмежень і різних типів невизначеності [82].

IoT здатний сприяти сприянню сприйнятливого попиту в системі. Існують різні перешкоди, які можуть обмежити участь у програмах. Ці бар’єри можна розділити на три ключові групи, а саме бар’єри клієнта, бар’єри постачальників і структурні бар’єри, які всебічно вивчені.

2 МОДЕЛІ ТА МЕТОДИ УПРАВЛІННЯ РОЗПОДІЛЕНИМИ ГЕТЕРОГЕННИМИ ХМАРНИМИ СИСТЕМАМИ

У будь якому місті застосування Інтернету речей (IoT) і сенсорних пристроїв роблять місто розумним і інтелектуальним. Різні сенсори, мікроконтролерні системи та програмне забезпечення, такі як температура та вологість, система управління енергією, забруднення повітря, система збору сміття, все це елементи нової технології для створення розумного та інтелектуального міста.

Щоб продемонструвати ефективність запропонованого нами алгоритму для оптимізації горизонтальної масштабованості в хмарних обчисленнях, ми провели порівняльний аналіз із кількома існуючими методами. Зокрема, ми порівняли наш алгоритм з двома іншими методами оптимізації, а саме генетичним алгоритмом (GA) і оптимізацією роєм частинок (PSO), які зазвичай використовуються для вирішення подібних проблем.

Ми реалізували кожен алгоритм, використовуючи той самий набір вхідних параметрів, і провели серію експериментів, щоб виміряти їх продуктивність з точки зору часу виконання, використання віддалених ресурсів і масштабованості.

Наші результати показують, що запропонований нами алгоритм стабільно перевершує як GA, так і PSO з точки зору якості рішення та масштабованості. Зокрема, наш алгоритм здатний досягти вищого ступеня горизонтальної масштабованості та кращого використання ресурсів порівняно з іншими методами, а також є більш ефективним з точки зору організації розподілених обчислень.

Ці результати є переконливими доказами переваг запропонованого нами алгоритму та підтверджують наше твердження, що він перевершує існуючі на даний момент методи.

2.1 Масштабованість програми IoT для розумної інформаційної системи міста

Масштабованість є однією зі складних особливостей програми IoTs, яка забезпечує розширення пристроїв відповідно до вдосконалення програм IoT. Різні програми IoT моделюються за допомогою наборів даних у реальному часі [17,18]. Ці набори даних складаються з даних про максимальні траєкторії за допомогою системи глобального позиціонування (GPS), яка містить інформацію про довготу, широту та висоту) у форматі CSV (значення, розділені комами). Сервер Інтернету речей (IoT) обробляє різні віртуальні пакети, щоб завершити один цикл обробки, який займе час (t) у мілісекундах (мс). Коефіцієнт затримки використовується для розрахунку цих отриманих віртуальних пакетів. Верхня і нижня межа затримки - 15 мс. Рівняння для розрахунку коефіцієнта затримки таке:

$$Ptime = rand(15,len), \tag{2.1}$$

де *len* – довжина пакету даних.

У наведеному вище рівнянні час обробки пакета довжиною *len*, обчислюється за допомогою функції *rand*(), яка повертає випадкове значення в залежності від довжини пакету з обмеженням мінімуму часу затримки в 15мс. Буфер віртуальної черги, що має можливість обмежувати набір значень для максимум 100 пакетів. Віртуальні шини завантажено в буфер відправника, який надсилає дані зі швидкістю, визначеною за допомогою атрибута екземпляра шини для імітації сервера обробки навантаження. Віртуальний сервер використовує отримання даних відповідно до черги отримання, а також обробку пакетів відповідно до коефіцієнта затримки кожного пакета в цій черзі. Затримка циклу однієї обробки розраховується за такою формулою:

$$Cycletime = \sum_{n=1}^{N} Ptime_n , \qquad (2.2)$$

де N – кількість пакетів.

За один цикл обробки, N показує кількість пакетів в системі. Додаючи більшу кількість шин b ми зменшуємо час циклу, але збільшуємо вартість обробки пакетів. Отже, загальне навантаження обробки, представлене за допомогою формули

$$Processtime = \frac{\sum_{n=1}^{N} Ptime_n}{b} \cdot Cyclecount \qquad (2.3)$$

де *Cyclecount* – кількість циклів обробки даних.

У наведеному вище рівнянні шини, які використовуються для даних, використовуються для надсилання даних на сервер на основі технології Інтернету речей (IoT). Для розрахунку розраховується середня затримка сервером. У наведеному вище рівнянні N використовується для загальної кількості пакетів у черзі буфера наприкінці циклу обробки. Якщо наш додаток IoTs стає повільним і не може впоратися із поточним збільшенням кількості пристроїв навіть за умови оптимізації, необхідно масштабувати цю систему, застосовуючи вертикальне масштабування та горизонтальне масштабування для належного масштабування додатків на основі IoT в університетських містечках.

2.2 Запропонований метод

Щоб додати більше пристроїв для застосування IoT, важливо перевірити коефіцієнт масштабування вертикальної масштабованості та

горизонтальної масштабованості для кращого використання застосування IoTs у місті.

2.2.1 Вертикальне масштабування

Блок схема алгоритму, що реалізує вертикальне масштабування представлено на рисунку 2.1.

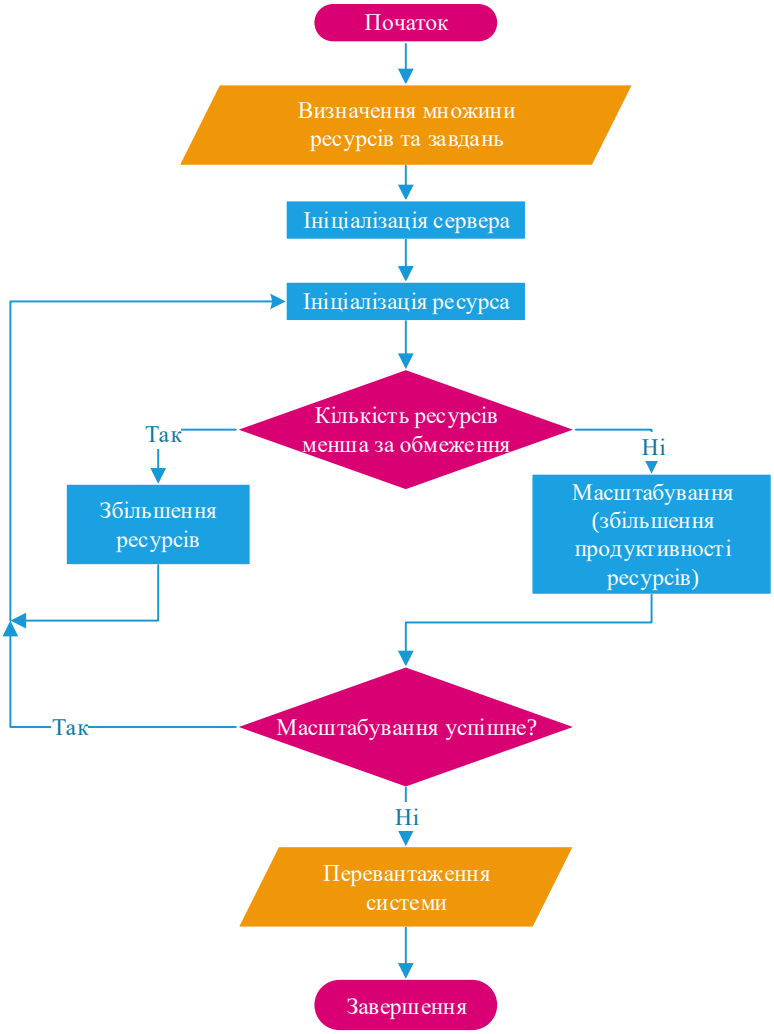

Рисунок 2.1 – Граф схема вертикального масштабування

Вертикальне масштабування (рисунок 2.2) виконується наступними кроками:

Крок 1. Відправник передає віртуальні дані на сервер віртуального Інтернету речей.

Крок 2. Віртуальний IoT-сервер обробляє дані, отримані від віртуального відправника. Ці отримані дані менші за максимальний ліміт потужності сервера.

Крок 3. Крок 2 повторюється, збільшуючи кількість пристроїв від відправника віртуальних даних, щоб збільшити навантаження на сервер, поки сервер не перевантажиться і не почне відкидати отримані пакети. Вертикальне масштабування коштує дорожче, ніж горизонтальне, і може вимагати вимкнення вашої машини на мить, коли відбувається більший процес (тобто додавання додаткових ресурсів, які збільшують потужність сервера).

2.2.2 Горизонтальне масштабування

Блок схема алгоритму, що реалізує вертикальне масштабування представлено на рисунку 2.2.

Горизонтальне масштабування виконується наступними кроками:

Крок 1. Відправник передає дані на сервер Horizontal IoT.

Крок 2. Горизонтальний сервер IoT обробляє дані, отримані від горизонтального відправника. Ці отримані дані менші за максимальний ліміт потужності сервера.

Крок 3. Крок 2 повторюється, збільшуючи кількість пристроїв від горизонтального відправника даних, щоб збільшити навантаження на сервер.

Горизонтальна масштабованість із функцією еластичності для додавання додаткових ресурсів для застосування IoT.

Змодельований алгоритм базується на процесі ланцюга Маркова, який є стохастичним процесом, що рухається через послідовність станів. У контексті моделювання хмарних обчислень, ланцюг Маркова використовується для дослідження простору можливих рішень проблеми. На кожній ітерації алгоритму нове рішення генерується шляхом внесення

невеликих змін до поточного рішення, і якість нового рішення порівнюється з якістю поточного рішення.

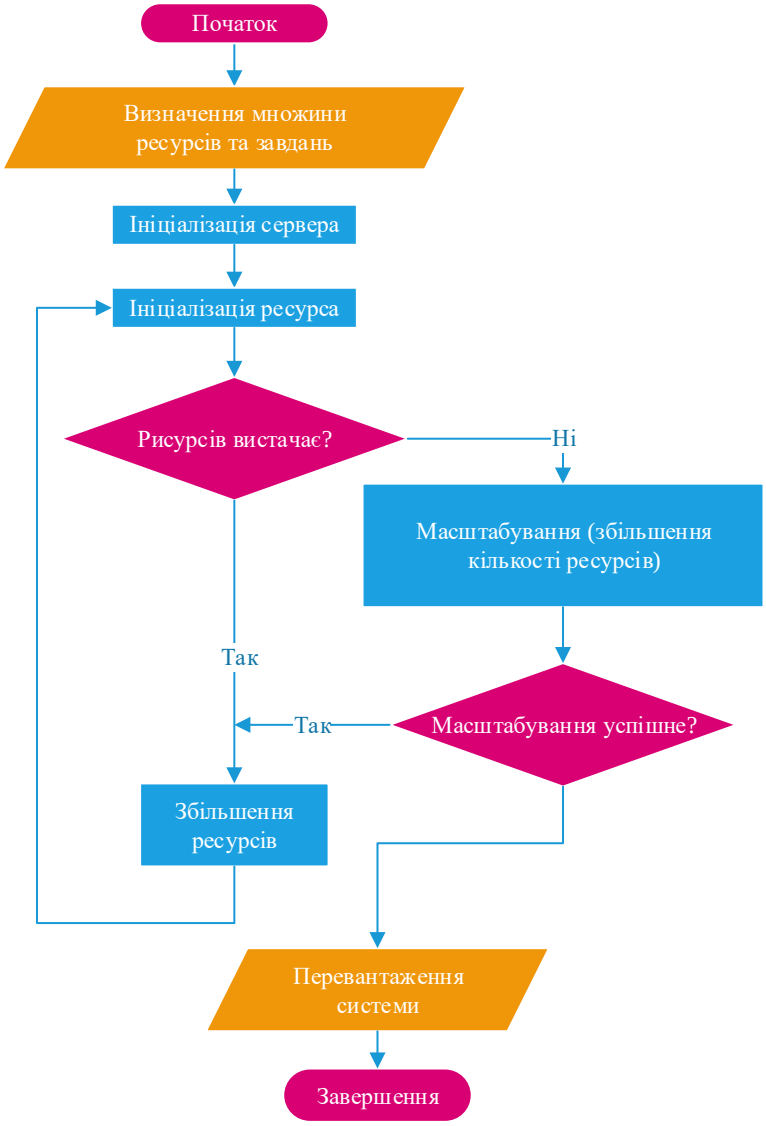

Рисунок 2.2 – Граф схема горизонтального масштабування

Якщо нове рішення краще, воно є прийнято як нове поточне рішення. Якщо нове рішення гірше, можливо все ще буде прийнято з певною ймовірністю, яка зменшується в міру просування алгоритму. Ця ймовірність базується на температурному параметрі, який контролює ступінь випадковості в процесі пошуку. У міру просування алгоритму температурний параметр поступово зменшується, що призводить до зближення алгоритму до

локального оптимального рішення. Поєднуючи процес ланцюга Маркова з алгоритмом симуляції відпалу, ми можемо ефективно здійснювати пошук у великому просторі можливих рішень і знаходити оптимальне рішення для заданої проблеми.

Запропонований нами алгоритм відпалу базується на процесі Маркова, та є стохастичним випадковим процесом, який має дискретну послідовність станів (рисунок 2.3).

```
1:  function SIMULATED_ANNEALING (initial_config)
2:      current_config ← initial_config
3:      current_cost ← cost(current_config)
4:      T ← initial_temperature while T > final_temperature do
5:       new_config ← generate_neighbor(current_config)
6:      new_cost ← cost(new_config)
7:      delta_cost ← new_cost − current_cost if delta_cost < 0 or random(0, 1) < probability
    (delta_cost, T) then
8:       current_config ← new_config
9:      current_cost ← new_cost
10:
11:      T ← update_temperature(T)
12:
13:      returncurrent_config??
14:  end function2
```

Рисунок 2.3 – Розроблений алгоритм розпаралелювання

Визначимо $X=x_1,x_2,\ldots,x_N$ – множина можливих рішень задачі, і нехай $f(x)$ функція витрат, яка вимірює якість рішення $x \in X$. На кожній ітерації алгоритму нове рішення y створюється шляхом внесення невеликих змін до поточного рішення відповідно до розподілу ймовірності переходу $p(x|y)$. Ймовірність прийняття нового рішення y визначається ймовірністю прийняття Метрополіса-Гастингса:

$$\alpha(x,y) = min\left\{1, exp(-\frac{f(y) - f(x)}{T})\right\}, \quad (2.4)$$

де $X=x_1,x_2,\ldots,x_N$ – це набір можливих рішень проблеми, $f(x)$ є функцією витрат, яка вимірює якість рішення $x \in X$, і $p(y|x)$ є розподілом ймовірності переходу, який генерує нове рішення y вносячи невеликі зміни в поточне рішення x. Температурний параметр T контролює ступінь випадковості в процесі пошуку. У міру просування алгоритму температурний параметр поступово зменшується відповідно до графіка охолодження, що призводить до зближення алгоритму до локального оптимального рішення. Поєднуючи процес ланцюга Маркова з алгоритмом симуляції відпалу, ми можемо ефективно здійснювати пошук у великому просторі можливих рішень і знаходити оптимальне рішення для заданої проблеми. При цьому температурний параметр постійно контролює випадковість рішення в пошуковому процесі. У той час, коли здійснюється просування алгоритму, проходить зменшення температурного параметру згідно з законом охолодження. Таким чином рішення наближається до локального оптимального рішення. Це дозволило нам поєднати Марківський процес з алгоритмом симуляції відпалу та здійснювати ефективний пошук.

2.3 Параметри оцінювання

У цьому підрозділі ми спробуємо отримати допомогу щодо кількох концепцій хмарних обчислень за допомогою стандартних рівнянь, які стосуються горизонтальної масштабованості. У хмарних обчисленнях кількісна оцінка характеристик продуктивності розподілених систем із збільшенням кількості вузлів або користувачів може допомогти підвищити продуктивність. Розуміючи ці концепції, ми можемо проектувати та оптимізувати розподілені системи для досягнення максимальної продуктивності та масштабованості. Закон Амдала допомагає описати

теоретичне максимальне прискорення, якого може досягти програма зі збільшенням кількості процесорів. Він передбачає, що деяка частина програми не може бути розпаралелена і завжди працюватиме на одному процесорі. Закон Амдала часто використовується для оцінки потенційних переваг розпаралелювання програми. Рівняння для розрахунку ефективності паралельної програми з використанням закону Амдала:

$$E = \frac{1}{(1-p) + \frac{p}{n_p}}, \tag{2.5}$$

де E – це ефективність, p – є паралельною частиною програми, і n_p – кількість процесорів, які паралельно виконують частини програми p.

Закон Густафсона допомагає описати масштабованість програми зі збільшенням кількості процесорів. На відміну від закону Амдала, закон Густафсона припускає, що розмір розв'язуваної проблеми збільшується зі збільшенням кількості процесорів, тому частина програми, яку можна розпаралелити, також зростає. Рівняння для розрахунку масштабованості паралельної програми за допомогою закону Густафсона:

$$S = p + (1 - p)N, \tag{2.6}$$

де S – це прискорення, p – паралельна частина програми, N – це кількість процесорів. Час передачі використовується для оцінки часу, необхідного для переміщення даних з одного місця в інше через мережу. Він враховує розмір даних, що передаються, доступну пропускну здатність і затримку мережі. Рівняння для розрахунку часу, необхідного для передачі даних через мережу:

$$T = \frac{D}{B} + L,, \tag{2.7}$$

де T – час передачі, D – це розмір даних, B – доступна пропускна здатність, L – це затримка (латентність).

Універсальний закон масштабованості використовується для опису максимальній кількості користувачів, яку може підтримувати розподілена система, зберігаючи прийнятну продуктивність. Він враховує використання системи та коефіцієнт конкуренції, який представляє ступінь конкуренції за спільні ресурси. Рівняння для розрахунку масштабованості розподіленої системи з використанням універсальної масштабованості:

$$N = \frac{U}{1-(U-1)C+\sqrt{(U-1)^2C^2+4(U-1)C}}, \qquad (2.8)$$

де N – кількість користувачів, U – це коефіцієнт використання системи, C – є фактором суперечок. Максимальна пропускна здатність використовується для оцінки максимальної швидкості, з якою дані можуть оброблятися розподіленою системою без конкуренції. Він припускає, що обробка системи та потужність мережі є обмежуючими факторами. Рівняння для розрахунку максимальної пропускної здатності розподіленої системи без конкуренції:

$$V = \frac{1}{\frac{1}{Tntwk}+\frac{1}{Tprcsr}}, \qquad (2.9)$$

де V – максимальна пропускна здатність, T_{ntwk} – час, необхідний для передачі даних по мережі, T_{prcsr} – процесорний час, необхідний для обробки пакету.

Оцінка нашого методу включала кілька параметрів, включаючи початкову температуру, швидкість охолодження та кількість ітерацій. Ми

вибрали ці параметри на основі попередніх досліджень симуляції відпалу та наших власних експериментів із проблемою.

Зокрема, ми провели серію пілотних експериментів, щоб визначити прийнятний діапазон значень для кожного параметра, а потім виконали пошук у сітці, щоб визначити комбінацію значень параметрів, яка призвела до найкращої продуктивності наших тестових програм. Ми вибрали ці параметри на основі їх здатності збалансувати компроміс між дослідженням і використанням пошукового простору та уникнути застрягання в локальних оптимумах.

Зокрема, ми вибрали початкову температуру, яка була досить високою, щоб забезпечити достатнє дослідження простору пошуку, швидкість охолодження, яка була досить повільною, щоб уникнути передчасної конвергенції, і достатню кількість ітерацій, щоб забезпечити конвергенцію до хорошого рішення. Ретельно вибравши та налаштувавши ці параметри, ми змогли продемонструвати ефективність нашого методу оптимізації горизонтальної масштабованості в хмарних обчисленнях для Інтернету речей.

3 ІНТЕГРАЦІЯ МЕТОДУ В ХМАРНЕ СЕРЕДОВИЩЕ ТА ПРОВЕДЕННЯ ЕКСПЕРИМЕНТІВ

3.1 Опис експериментальної частини дослідження

Дослідницька схема для цього дослідження полягала в поєднанні експериментальних і спостережних методів. Спочатку ми визначили набір тестових програм для використання в моделюванні, а потім провели експерименти, щоб виміряти продуктивність цих програм на різних хмарних архітектурах. Ми використали моделювання відпалу для оптимізації горизонтальної масштабованості хмарних архітектур з метою досягнення найкращої продуктивності для еталонних програм. Дані для цього дослідження було зібрано шляхом моделювання за допомогою інструментарію CloudSim. Ми використали метод стратифікованої випадкової вибірки для вибору вхідних параметрів для моделюваного алгоритму відпалу. Щоб проаналізувати дані, ми використали такі статистичні методи, як перевірка гіпотез і довірчі інтервали, щоб визначити значущість наших результатів.

Моделювання відбувалось на 16 (A10) віртуальних машинах, які працювали на платформі Azure (фірми Microsoft) з 128 ядрами. Таблиця 3.1 містить конфігурацію кожної віртуальної машини у кластері.

Таблиця 3.1 – Апаратна конфігурація для віртуальної машини

Хмарна платформа	Тип	Ядра	Процесор	Об'єм пам'яті	Тип пам'яті	Віртуалізація	Мережа
Azure	A10	8	IntelXeon E5-2670@ 2.6 GHz	64 GB	DDR3	Hyper-V	10 gigabit ethernet

Для проведення експериментальної частини була обрана операційна система CentOS 6.5. Оцінка масштабованості, продуктивності та впливу використання розроблених методу та алгоритму проводилася з використанням гібридних кодів.

Навантаження на хмарну систему перевіряли ефективне використовування розробленого програмного забезпечення з урахуванням вартості проведення обчислень. Реалізовані навантаження на систему обслуговувалися за допомогою віртуальних машин A10 VM.

Для оцінки навантаження використовувались наступні характеристики: швидкість і MOPS. MOPS надає результати на рівні бенчмарку. Також було аналізоване прискорення на базі співвідношення часу, що потрібен для послідовного виконання, та часу паралельного виконання.

Оцінка результатів проводилася шляхом повторних випробувань для збільшення точності оцінки.

3.2 Результати тестів на масштабованість

На графіку (рисунок 3.1) показано результати тесту на масштабованість, проведеного на розподіленій системі для оцінки її продуктивності за різних робочих навантажень. Універсальний закон масштабованості використовувався для розрахунку максимальної кількості користувачів, яку може підтримувати система, зберігаючи прийнятну продуктивність. Тест проводився шляхом поступового збільшення кількості користувачів і вимірювання часу відгуку системи. Результати показують, що система могла обслуговувати максимум 500 користувачів, перш ніж час її відгуку перевищив прийнятний поріг. Після досягнення порогу треба використовувати методи масштабування. Цю інформацію можна використовувати для прийняття рішень щодо планування потужності системи та розподілу ресурсів для забезпечення оптимальної продуктивності за очікуваних робочих навантажень (рисунок 3.1).

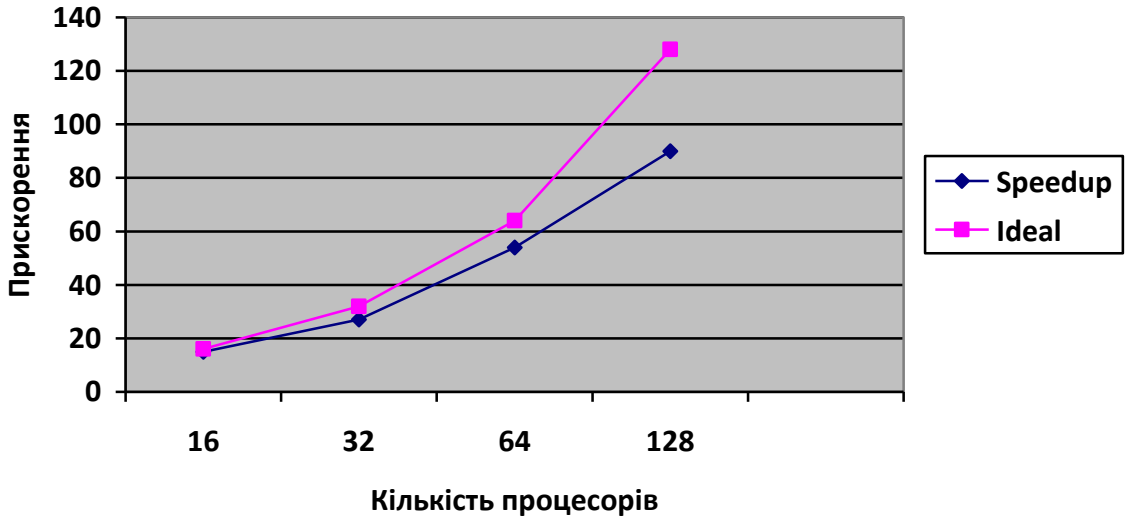

Рисунок 3.1 – Залежність прискорення (speedup) від кількості процесорів

На рисунку 3.2 показана залежність часу обробки від кількості пакетів даних. По осі X вказано кількість запитів, а по осі Y – час, витрачений на обробку запиту. На малюнку показано порівняння вертикальної та горизонтальної масштабованості максимального часу для запиту за використаний час.

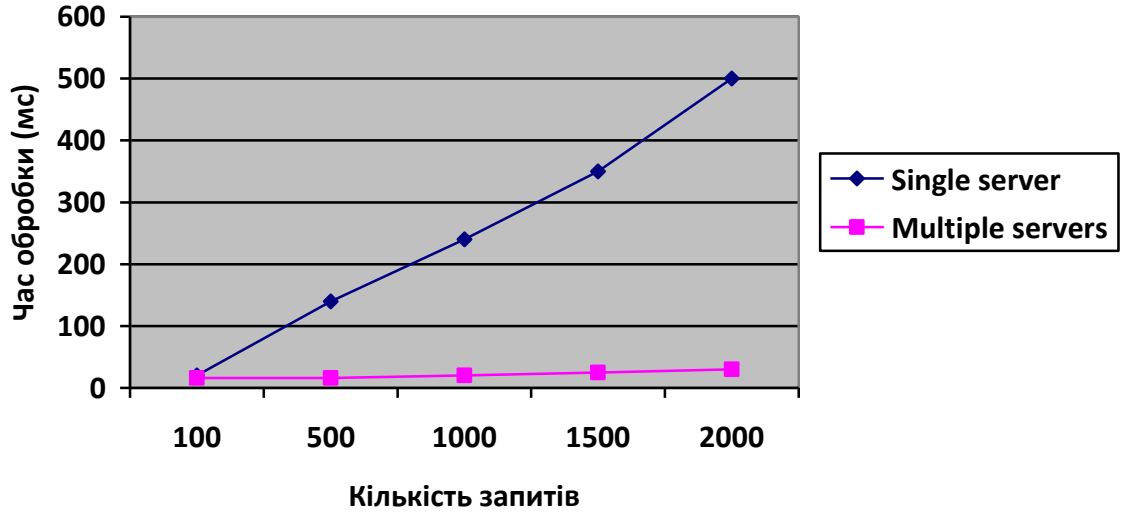

Рисунок 3.2 – Залежність часу обробки від кількості запитів за вертикальною та горизонтальною масштабованостями

Горизонтальне масштабування забезпечує швидшу доставку, ніж вертикальне масштабування, завдяки меншому середньому часу запиту. З точки зору оперативності, горизонтальна масштабованість забезпечує кращу продуктивність завдяки розподілу навантаження на сервер між доступними вузлами. Горизонтальна масштабованість запобігає виникненню дефіциту ресурсів. Горизонтальна масштабованість економічно ефективніша, ніж вертикальна. Рисунок 3.2 показує кількість запитів по осі X і часові витрати по осі Y.

Результати, представлені на рисунку можна використовувати для порівняння вертикальної та горизонтальної масштабованості з точки зору оцінки вартості на місяць. Горизонтальна масштабованість забезпечує швидшу доставку, ніж вертикальна масштабованість через менший середній час запиту. З точки зору швидкості доставки, горизонтальна масштабованість надає ліпшу продуктивність, тому що розподіл навантаження виконується між усіма доступними вузлами. У випадку горизонтальної масштабованості немає виникнення дефіциту обчислювальних ресурсів.

3.3 Оцінка часу обслуговування

Рисунок 3.3 показує конфігурацію машини (об'єм пам'яті) по осі X і витрачений час по осі Y. На малюнку показано порівняння вертикальної та горизонтальної масштабованості максимального часу для запиту за використаний час. Горизонтальна масштабованість забезпечує швидшу доставку, ніж вертикальна масштабованість через менший середній час запиту. Горизонтальна масштабованість забезпечує кращу продуктивність ніж вертикальна, тому що усі запити на обробку при збільшенні об'ємів даних система розподіляє між доступними процесорними вузлами. Кожен вузол має не тільки свій процесор для обчислення, але й свою оперативну та зовнішню пам'ять, що є ще одним фактором, який впливає на збільшення швидкості обробки даних системою у цілому.

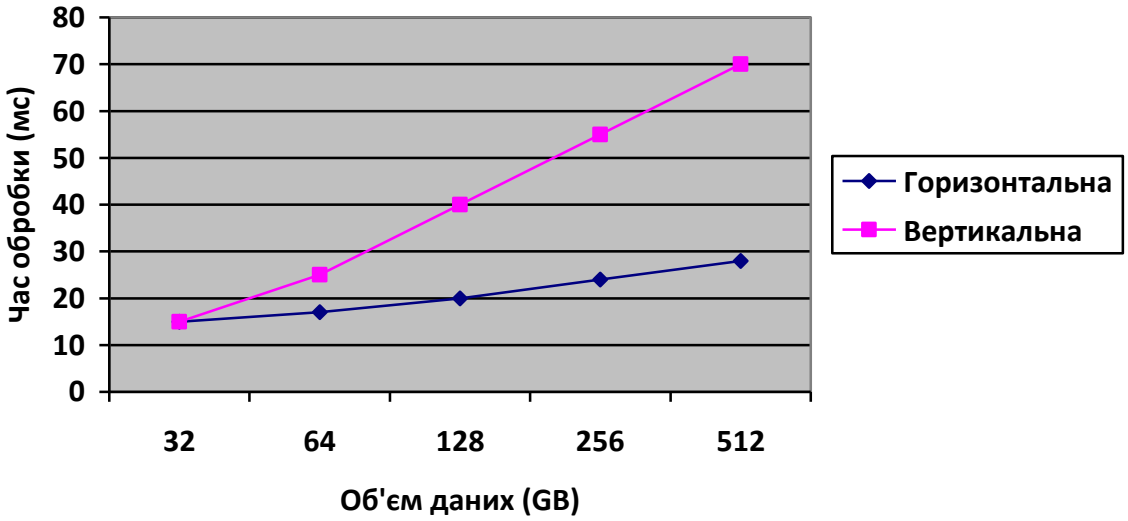

Рисунок 3.3 – Залежність часу обробки від об'єму даних для обробки запитів за вертикальною та горизонтальною масштабованостями

Наступний експеримент показує залежність часу відгуку системи при зростанні кількості користувачів при використанні вертикальній та горизонтальній масштабованості (рисунок 3.4). Аналіз графіку показує, що вертикальна масштабованість дає максимальний ефект на невеликому прирості кількості клієнтів.

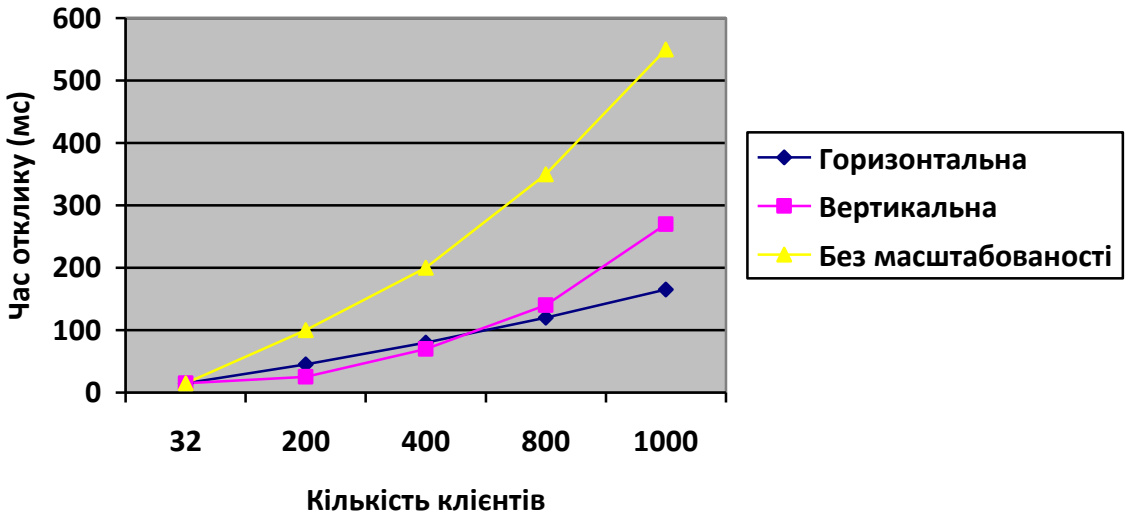

Рисунок 3.4 – Залежність часу відгуку при зростанні кількості клієнтів за вертикальною та горизонтальною масштабованостями

Однак, при подальшому зростанні кількості клієнтів, спостерігається більш швидке зростання часу відгуку при вертикальному масштабуванні. Цей факт можна пояснити тим, що був досягнута межа можливостей збільшення продуктивності за рахунок наявного обладнання.

У випадку горизонтального масштабування, при необмежному додаванні ресурсів, зростання часу відгуку значно менша. Наявність зростання в загалі, при горизонтальному масштабуванні, пояснюється витратами на керування розподіленими обчисленнями, розподіл даних, та обробку результатів обчислень.

Графік результатів показує розрахункову максимальну кількість користувачів, яку може підтримувати розподілена система, зберігаючи прийнятну продуктивність, як передбачено формулою універсального закону масштабованості. На графіку показано, що зі збільшенням коефіцієнта використання та конкуренції максимальна кількість користувачів, яку може підтримувати система, зменшується. Це пов’язано з тим, що в міру того, як система стає більш завантаженою та фактором конкуренції, зростає конкуренція за спільні ресурси, що може призвести до зниження продуктивності та в кінцевому підсумку вплинути на кількість користувачів, яких може підтримувати система.

Графік також показує, що крива стає крутішою, оскільки коефіцієнт використання та конкуренції також зростає, що вказує на те, що продуктивність деградує швидше, оскільки система стає більш інтенсивною. Загалом, цей графік дає цінну інформацію про масштабованість розподіленої системи та може інформувати про рішення, пов’язані з плануванням потужності та розподілом ресурсів.

Результати, які відображено на рисунках, представлено у вигляді двох таблиць, окремо для вертикальної (таблиця 3.2) та горизонтальної (таблиця 3.3) масштабованості.

Незважаючи на обнадійливі результати, отримані за допомогою запропонованого нами методу для оптимізації горизонтальної

масштабованості в хмарних обчисленнях для програм IoT, є деякі обмеження нашого дослідження, які слід зазначити.

Таблиця 3.2 – Вертикальна масштабованість

Кільк. процес-орів	Час обслуг. (с)	Передача даних (Кб/с)	Мінім. час обслугову-вання (мс)	Макс. час обслугову-вання(мс)	Час відгуку (мс)	Оцінка вартості (USD)
1	65	155	45	64343	7	0.7
2	55	200	40	40399	5	2.1
4	37	250	42	30545	4	3.3
8	34	280	43	25789	3	5.8
16	30	340	55	24998	2	7.1

Таблиця 3.3 – Горизонтальна масштабованість

Кільк. процес-орів	Час обслуг. (с)	Передача даних (Кб/с)	Мінім. час обслугову-вання (мс)	Макс. час обслугову-вання(мс)	Час відгуку (мс)	Оцінка вартості (USD)
1	65	155	45	64343	7	0.7
2	45	210	45	35123	4	1.5
4	34	290	42	28646	3	2.4
8	30	310	44	23434	2	3.3
16	26	450	38	20345	2	5.4

Одним з обмежень є те, що наше дослідження було зосереджено лише на застосуванні імітації відпалу як алгоритму оптимізації та не досліджував використання інших методів машинного навчання або алгоритмів оптимізації.

Крім того, хоча наші результати були багатообіцяючими, вони обмежуються певним набором показників ефективності, і подальші дослідження необхідні для перевірки наших висновків у різних сценаріях і

використанні різних критеріїв оцінювання. Є кілька напрямків на майбутні дослідження, які могли б допомогти усунути обмеження нашого дослідження та розширити запропоноване нами рішення.

Однією з областей для майбутніх досліджень може бути дослідження використання інших методів машинного навчання та алгоритмів оптимізації для оптимізації горизонтальної масштабованості в хмарній системі для програм IoT.

Крім того, необхідні подальші дослідження, щоб оцінити ефективність запропонованого нами рішення з використанням різних показників продуктивності та в різних сценаріях.

Ще одна сфера майбутнього дослідження може полягати в дослідженні застосування горизонтальної масштабованості і змодельованого алгоритму відпалу для інших областей хмарних обчислень за межами IoT програм. Наприклад, можна зробити дослідження потенційних переваг горизонтальної масштабованості для інших типів програм із залученням штучного інтелекту або машинного навчання.

Нарешті, пропонуємо дослідити вплив різного робочого навантаження типів на ефективність запропонованого нами рішення. Дослідження може включати моделювання різних видів навантажень та оцінки ефективності нашого рішення за різними сценаріями.

ВИСНОВКИ

В кваліфікаційній роботі реалізовано експериментальний підхід до горизонтальної масштабованості в реальному часі з вбудованою функцією еластичності для додавання більшої кількості пристроїв на базі IoT відповідно до розширення застосування IoT для конкретної мережі в місті. У вертикальній масштабованості встановлено обмеження для додавання пристроїв IoTs і мережевих ресурсів для розробки конкретної програми IoTs. Якщо розробка програми IoTs дуже велика, і в майбутньому потрібно додати більше пристроїв IoTs, у такому випадку краще розширити програму IoTs за допомогою горизонтальної масштабованості.

Експериментальні результати показують, що порівняно з вертикальною масштабованістю горизонтальна масштабованість показує кращі результати для додавання ресурсів відповідно до зростання програми IoT і балансування робочого навантаження між різними вузлами в програмі IoT. У роботі ми пропонуємо експериментальний підхід у режимі реального часу щодо важливості горизонтальної масштабованості для міста у випадку розробки конкретного програмного забезпечення IoT.

Вертикальна масштабованість додатка IoT застосовується, доки не буде зменшено певний ліміт для середнього часу відповіді.

Горизонтальна масштабованість має властивість еластичності, тобто кількість пристроїв IoT, шлюзів, серверів, балансувальників навантаження або будь-яких ресурсів, які можна додати відповідно до вдосконалення застосування IoT у будь-якому місті. Горизонтальна масштабованість зосереджена на балансуванні навантаження мереж для рівномірного розподілу даних між різними пристроями в мережі.

При горизонтальній масштабованості, якщо деякі вузли та ресурси вийдуть з ладу, вся система не вийде з ладу. Горизонтальна масштабованість звичайно реалізується із сітчастою топологією всіх підключених пристроїв у

мережі IoT, щоб інші ресурси могли керувати без погіршення поточної системи. Ця функція демонструє можливість відмовостійкості горизонтальної масштабованості для кращої продуктивності, властивої еластичності, додаючи необмежені ресурси для майбутнього вдосконалення програми IoTs, балансування навантаження, відмовостійкість для повноцінної розробки програми IoTs для будь-якого міста.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. P. Ray. A survey on internet of things architectures, Journal of King Saud University-Computer and Information Sciences. Vol.30 (3).2018. P.291–319. https://doi.org/10.1016/j.jksuci.2016.10.003

2. B. Safaei, A. Monazzah, M. Bafroei, A. Ejlali. Reliability side-effects in internet of things application layer protocols, in: 2nd International Conference on System Reliability and Safety (ICSRS).2017. P. 207–212. DOI: 10.1109/ICSRS.2017.8272822

3. S. Talari. A review of smart cities based on the internet of things concept, Energies 10 (4). 2017. P.1-23. https://doi.org/10.3390/en10040421

4. S. Noor. Energy demand side management within micro-grid networks enhanced by blockchain. Applied Energy. Vol. 228. 2018. P. 1385–1398. https://doi.org/10.1016/j.apenergy.2018.07.012

5. E. Borgia. The internet of things vision: key features, applications and open issues. Computer Communication. Vol. 54. P. 1–31. https://doi.org/10.1016/j.comcom.2014.09.008

6. A. Gupta, R. Christie, R. Manjula. Scalability on Internet of Things: Features, Techniques, and Research Challenges. International Journal of Computational Intelligence Research. Vol. 13. 2017. P. 1617-1627

7. A. Javed, Scalable iot platform for heterogeneous devices in smart environments, IEEE Access Vol 8. 2020. P. 211973–211985. DOI: 10.1109/ACCESS.2020.3039368

8. M. R. Palattella. Internet of things in the 5g era: enablers, architecture, and business models. IEEE Journal on Selected Areas in Communications. Vol.34, Issue 3. 2016. P. 510–527. DOI: 10.1109/JSAC.2016.2525418

9. T. Bakhshi. State of the Art and Recent Research Advances in Software Defined Networking. Wireless Communications and Mobile Computing. Article ID 7191647. 2017. P.1-35. https://doi.org/10.1155/2017/7191647

10. M. A. Razzaq. Simulation and assessment of vertical scaling for a smart campus environment using the internet of things, IEEE Access 10 96322–96330.

11. H. Z. Fahmi, F. J. Lin, Nfv-enabled vertical scalability for iot slices, in: 22nd Asia- Pacific Network Operations and Management Symposium. IEEE access, vol. 10. pp. 96322-96330. 2022. doi: 10.1109/ACCESS.2022.3204042

12. M. Gusev. Scalable dew computing, Appl. Sci. Vol.12 (19) 9510. 2022. P.1-10. https://doi.org/10.3390/app12199510

13. W. Brockelsby, R. Dutta. Traffic analysis in support of hybrid sdn campus architectures for enhanced cybersecurity. 2021 24th Conference on Innovation in Clouds, Internet and Networks and Workshops. Paris, France. 2021. P. 41-48. DOI: 10.1109/ICIN51074.2021.9385530.

14. S. Tang. Coordinate-based efficient indexing mechanism for intelligent iot systems in heterogeneous edge computing, J. Parallel Distr. Comput. Vol.166.2022. P. 45–56. https://doi.org/10.1016/j.jpdc.2022.04.012

15. W. Han, S. Li, The method of allocating resources for ideological and political education in universities based on iot technology. Journal of Information & Knowledge ManagementVol. 21, No. Supp02, 2240011 (2022). https://doi.org/10.1142/S0219649222400111

16. W. Altwoyan, I. S. Alsukayti, A novel iot architecture for seamless iot integration into university systems, Int. J. Adv. Comput. Sci. Appl. 13 (4).

17. Y. Zheng, Q. Li, Y. Chen, X. Xie, W. Ma. Understanding mobility based on gps data. 10th International Conference on Ubiquitous Computing. 2008 P.312–321. https://doi.org/10.1145/1409635.1409677

18. Y. Zheng, X. Xie, W. Ma. Geolife: a collaborative social networking service among user, location and trajectory. IEEE Data Eng Bull. Vol. 33(2). 2010. P. 32–39.

19. M. A. Vouk. Cloud computing : Issues, research and implementations. In Information Technology Interfaces, 2018. ITI 2018. 30th International Conference on, , June 2018. P. 31–40.

20. L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner. A break

in the clouds: Toward a cloud definition. ACM SIGCOMM Computer Communication Review, 39(1). 2019. P.50–55.

21. L. Johnson, A. Levine, and R. Smith. The 2019 horizon report, 2019. Austin, Texas, The New Media Consortium.

22. Foster, Y. Zhao, I. Raicu and S. Lu, “Cloud Computing and Grid Computing 360-Degree Compared,” Grid Computing Environments Workshop (GCE '08), 2008.

23. R. Buyya, R. Ranjan and R. N. Calheiros, Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. Proceedings of the Conference on High Performance Computing and Simulation, Leipzig, Germany. IEEE Press: New York, U.S.A., 21–24 June 2019. pp.1–11.

24. L. Vaquero, L. Rodero-Merino , J. Caceres and M. Lindner, A break in the clouds: towards a cloud definition. SIGCOMM Computer Communication Review, Volume 39, Number 1, January 2019. pp.50-55.

25. NIST definition of cloud computing, http://csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf

26. L. Youseff, M. Butrico and D. Da Silva, Toward a Unified Ontology of Cloud Computing, Grid Computing Environments Workshop (GCE '08), 2008.

27. Six benefits of cloud computing. Available at http://web2.sys-con.com/node/640237.

28. B. Hayes. Cloud computing. Communications of the ACM, 51(7). July 2018. pp.9–11.

29. B. P. Rimal, E. Choi and I. Lumb, A Taxonomy and Survey of Cloud Computing Systems. Fifth International Joint Conference on INC, IMS and IDC. 2019, pp 44–51.

30. Gathering clouds of XaaS! https://www.ibm.com/developerworks/mydeveloperworks/blogs/sbose/entry/gathering_clouds_of_xaas?lang=en

31. B. Sonisky, chapitre 1, cloud computing bible,wiley publishing inc 2011

32. Q. Zhang, L. Cheng and R. Boutaba, Cloud computing : state of the art and research challenges. Journal of Internet Services and Applications, vol. 1, 2020 pp. 7–18.

33. C.N. Hoefer and G. Karagiannis, Taxonomy of cloud computing services. In: Proceedings of the 14th IEEE workshop on enabling the future service-oriented Internet (EFSOI'20), Workshop of IEEE GLOBECOM 2020, pp. 1345–1350.

34. C. N. Hofer and G. Karagiannis, Cloud computing services: taxonomy and comparison, Journal of Internet Services and Applications, Springer, Vol. 2, September 2011, No. 2. pp. 69-79.

35. A. Lenk, M. Klems, J. Nimis, S. Tai and T. Sandholm. What's Inside the Cloud? An Architectural Map of the Cloud Landscape. In ICSE Workshop on Software Engineering Challenges of Cloud Computing, May 2019.

36. L. Wang and G. von Laszewski, Scientific Cloud Computing: Early Definition and Experience," in 20th IEEE International Conference on High Performance Computing and Communications, HPCC'18. 2-18. pp.825-830.

37. Q. Zhang, L. Cheng and R. Boutaba, Cloud computing : state of the art and research challenges. Journal of Internet Services and Applications, vol. 1. 2020. pp. 7–18.

38. Chiang, M. and Zhang, T., 2016. Fog and IoT: An overview of research opportunities. IEEE Internet of Things Journal, 3(6), pp.854-864.

39. Chen, M., Ma, Y., Song, J., Lai, C.F. and Hu, B.,. Smart clothing: Connecting human with clouds and big data for sustainable health monitoring. Mobile Networks and Applications, 21(5), 2016, pp.825-845.

40. Kumar, V., Laghari, A.A., Karim, S., Shakir, M. and Brohi, A.A., 2019. Comparison of Fog Computing & Cloud Computing. International Journal of Mathematical Sciences and Computing (IJMSC), 5(1), pp.31-41.

41. Jayaraman, P.P., Perera, C., Georgakopoulos, D., Dustdar, S., Thakker, D. and Ranjan, R., 2017. Analytics-as-a-service in a multi-cloud environment through semantically-enabled hierarchical data processing. Software: Practice and

Experience, 47(8), pp.1139-1156.

42. Laghari, A.A., He, H., Karim, S., Shah, H.A. and Karn, N.K., 2017. Quality of experience assessment of video quality in social clouds. Wireless Communications and Mobile Computing, 2017.

43. Power, B. and Weinman, J., 2018. Revenue Growth is the Primary Benefit of the Cloud. IEEE Cloud Computing, 5(4), pp.89-94.

44. Benlian, A., Kettinger, W.J., Sunyaev, A., Winkler, T.J. and GUEST EDITORS, 2018. The transformative value of cloud computing: a decoupling, platformization, and recombination theoretical framework. Journal of management information systems, 35(3), pp.719-739.

45. Akherfi, K., Gerndt, M. and Harroud, H., 2018. Mobile cloud computing for computation offloading: Issues and challenges. Applied computing and informatics, 14(1), pp.1-16.

46. Mitra, A., O'Regan, N. and Sarpong, D., 2018. Cloud resource adaptation: A resource-based perspective on value creation for corporate growth. Technological Forecasting and Social Change, 130, pp.28-38.

47. Díaz, M., Martín, C. and Rubio, B., 2016. State-of-the-art, challenges, and open issues in the integration of the Internet of things and cloud computing. Journal of Network and Computer Applications, 67, pp.99-117.

48. Vrable, M., Savage, S. and Voelker, G.M., 2012, February. BlueSky: A cloud-backed file system for the enterprise. In Proceedings of the 10th USENIX conference on File and Storage Technologies (pp. 19-19). USENIX Association.

49. Coppolino, L., D'Antonio, S., Mazzeo, G. and Romano, L., 2017. Cloud security: Emerging threats and current solutions. Computers & Electrical Engineering, 59, pp.126-140.

50. Marinescu, D.C., 2017. Cloud computing: theory and practice. Morgan Kaufmann.

51. Sultan, N., 2015. The implications of cloud computing for global enterprise management. Global enterprise management (pp. 39-56). Palgrave Macmillan, New York.

52. Galloway, S., 2017. The four: the hidden DNA of Amazon, Apple, Facebook and Google. Random House.

53. Gajbhiye, A. and Shrivastava, K.M.P., 2014, September. Cloud computing: Need, enabling technology, architecture, advantages and challenges. In 2014 5th International Conference-Confluence The Next Generation Information Technology Summit (Confluence) (pp. 1-7). IEEE.

54. Aruna Irani, A.R., Manjula, D. and Sugumaran, V., 2019. Task scheduling techniques in cloud computing: A literature survey. Future Generation Computer Systems, 91, pp.407-415.

55. Verma, A., Pedrosa, L., Korupolu, M., Oppenheimer, D., Tune, E. and Wilkes, J., 2015, April. Large-scale cluster management at Google with Borg. In Proceedings of the Tenth European Conference on Computer Systems (p. 18). ACM.

56. Varghese, B. and Buyya, R., 2018. Next-generation cloud computing: New trends and research directions. Future Generation Computer Systems, 79, pp.849-861.

57. Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J. and Ghalsasi, A., 2011. Cloud computing—The business perspective. Decision support systems, 51(1), pp.176-189.

58. Laghari, A.A., He, H., Shafiq, M. and Khan, A., 2017, May. Impact of storage of mobile on quality of experience (QoE) at the user level accessing the cloud. In 2017 IEEE 9th international conference on communication software and networks (ICCSN) pp. 1402-1409.

59. Aljamal, R., El-Mousa, A. and Jubair, F., 2018, April. A comparative review of high-performance computing major cloud service providers. In 2018 9th International Conference on Information and Communication Systems (ICICS) (pp. 181-186). IEEE.

60. М.О. Волк, В.С Курочкін, А.П. Запорожченко, П.А. Паронікян. Гібридний метод розподілу ресурсів в хмарних системах. Системи управління, навігації та зв'язку. № 2 (76). 2024. С. 70-73. Фахове видання.