Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____

Кафедра _____Програмної інженерії_____

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти – другий (магістерський)

Дослідження методів управління доступом до сховищ даних з використанням туманної архітектури

Виконав: студент 2 курсу, групи ПЗСм-18-1___

_____Жажкий І.І._____
(прізвище, ініціали)

спеціальності 121- Інженерія програмного забезпечення
(код і повна назва спеціальності)

Освітньо-професійної програми_____
(тип програми)

Програмне забезпечення систем _____
(повна назва освітньої програми)

Керівник _____к.т.н. доц. Лановий О.Ф. _____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри, проф. _____ З.В.Дудар

20__р.

Харківський національний університет радіоелектроніки

Факультет_Комп'ютерних наук_____

Кафедра_Програмної інженерії_____

Рівень вищої освіти - другий (магістерський)

Спеціальність _121-Інженерія програмного забезпечення_____
(код і повна назва)

Тип програми освітньо-професійна програма_____

Освітня програма_ Програмне забезпечення систем_____

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові _Жажкому Ігорю Ігоровичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи _Дослідження методів управління доступом до сховищ даних з використанням туманної архітектури_____
затверджена наказом університету від "___" _____20 ___ р № _____
2. Термін подання студентом роботи до екзаменаційної комісії
_____
3. Вихідні дані до роботи архітектура туманного обчислення, архітектура хмарного обчислення, пояснювальна записка.
4. Перелік питань, що потрібно опрацювати в роботі мета роботи, аналіз проблемної галузі і постановка задачі, аналіз методів застосування архітектури туманного обчислення при роботі з великими об'ємами даних, методи доступу та управління сховищами даних у туманній архітектурі, варіанти моделей побудови систем з використанням туманного обчислення

Наступний аркуш завдання

5 Консультанти розділів роботи

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка*
1.	Аналіз предметної галузі	15 вересня 2019р.	
2.	Огляд варіантів систем, що використовують туманне обчислення	10 жовтня 2019р.	
3.	Моделювання архітектури туманного обчислення	17 жовтня 2019р.	
4.	Підготовка пояснювальної записки	23 жовтня 2019р.	
5.	Спецчастина	13 листопада 2019р.	
6.	Підготовка презентації та доповіді	04 грудня 2019р.	
7.	Попередній захист	07 грудня 2019р.	
8.	Нормоконтроль, рецензування	09 грудня 2019р.	
9.	Занесення диплома в електронний архів	17 грудня 2019р.	
10.	Допуск до захисту у зав. кафедри	18 грудня 2019р.	

Дата видачі завдання _____ _____2019 р.

Студент _____
(підпис)

Керівник роботи _____ к.т.н. доц. Лановий О.Ф.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до атестаційної роботи магістра, 59 с., 17 рисунків, 8 таблиць, 20 джерел.

АТЕСТАЦІЙНА РОБОТА, ТУМАННЕ ОБЧИСЛЕННЯ, ХМАРНИЙ СЕРВЕР, ТУМАННІ ВУЗЛИ, ГРАНИЧНА МЕРЕЖА, ШЛЮЗ, ВІРТУАЛЬНИЙ КОНТЕЙНЕР, КЛАСТЕР.

Об’єктом дослідження є туманне обчислення. Предметом дослідження є методи моделювання туманної архітектури.

Метою роботи є дослідження методів доступу та управління сховищами даних у туманній архітектурі.

У результаті роботи було здійснено вимірювання ефективності використання туманної архітектури при роботі з високо навантаженими сховищами даних. Було проведене моделювання та порівняння туманної та хмарної архітектури у системі з великим навантаженням та з великим об’ємом даних.

ATTESTATION WORK, FOG COMPUTING, CLOUD SERVER, FOG NODES, EDGE NETWORK, GATEWAY, VIRTUAL CONTAINER, CLUSTER.

The object of the study is fog computing. The subject of the study are the fog architecture modeling methods.

The purpose of the study is to investigate methods of data storages accessing and management in fog architecture.

As a result, the efficiency of fog architecture usage for high-load data storages was measured. Fog and cloud architectures were modeled compared for high-load system with big amount of data.

ЗМІСТ

Вступ.....6
1 Аналіз предметної галузі.....7
1.1 Особливості реалізації туманної архітектури.....7
1.2 Виявлення проблем та актуалізація рішень.....15
1.3 Автентифікація у туманній архітектурі.....21
1.4 Безпека даних у туманному шарі.....22
1.5 Постановка задачі.....25
2 Опис теоретичних досліджень.....27
2.1 Порівняння існуючих видів архітектури.....27
2.2. Модель архітектури з використанням туманного обчислення..... 30
2.3. Опис теоретичних досліджень..... 33
3. Результати теоретичних досліджень..... 35
3.1. Опис дослідження моделей архітектури..... 35
3.2. Порівняння отриманих результатів..... 49
4. Впровадження результатів дослідження у наукову та практичну діяльність.....53
4.1. Опис можливості використання отриманих результатів у науковій та практичній діяльності.....53
4.2. Відомі проблеми туманних обчислень.....55
4.3. Питання подальшого дослідження..... 56
Висновки.....58
Перелік джерел посилання..... 59
Додаток А Слайди презентації..... 61
Додаток Б Апробація результатів роботи..... 71
Додаток В Рецензії..... 79
Додаток Г Відгук..... 82

ВСТУП

Наразі кількість електронних пристроїв у світі зростає з величезною швидкістю і разом з тим зростає і кількість одночасно передаваємих у мережі Інтернет даних. Саме через це все складніше стало створювати системи та сервери, архітектура яких допомогла б працювати з такою величезною кількістю даних та одночасно підключених клієнтів.

Незважаючи на широке використання хмарних сервісів та хмарного обчислення, воно наразі не здатне в повній мірі задовольнити всі потреби та вимоги індустрії інформаційних технологій. Хмарна архітектура не є ідеальним рішенням при роботі з великою кількістю даних через притаманні їй проблеми, а саме: занадто висока затримка при передачі даних через мережу, відсутність підтримки мобільності та розпізнавання локації обчислення, а також низький рівень захищеності важливих даних, що надходять з обчислювальних пристроїв.

Саме через неможливість хмарної архітектури в повній мірі задовольнити всі потреби сучасної індустрії інформаційних технологій і була створена туманна архітектура та туманне обчислення. Концепція туманного обчислення спрямована на зміну сценаріїв використання хмарного обчислення, а також на те, щоб зробити хмарні сервіси, орієнтовані на дані, децентралізованими та локалізованими.

Туманне обчислення це архітектура, яка направлена на зміну тих сценаріїв, що за останні роки були створені за допомогою хмарного обчислення, а також на те, щоб децентралізувати та локалізувати хмарні центри даних.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Особливості реалізації туманної архітектури

Останні декілька років хмарна архітектура була і залишається дуже поширеною у сфері інформаційних технологій завдяки своїм функціям керування мережевою інфраструктурою, обчислення, обробки та зберігання даних. Усі ці функції виконуються в централізованих центрах обробки даних, хмарних сервісах. Свою популярність серед користувачів та організацій хмарна архітектура завоювала через те, що це економічно вигідна послуга для вирішення основних бізнес-задач.

Популярність хмарного обчислення також призводить до того, що хмарні сервери дуже поширені по всьому світу і всі вони дуже ефективно глобально масштабуються. Це також підвищило продуктивність організації за рахунок зменшення кількості робіт, необхідних для виконання IT-командам. Послуги хмарного обчислення визнаються стеками [1], що представляють собою інфраструктуру як послугу (Infrastructure as a Service – IaaS), платформу як послугу (Platform as a Service – PaaS) та програмне забезпечення як послугу (Software as a Service – SaaS) та інші:

- програмне забезпечення як послуга (SaaS) – Ця модель надає можливість користувачеві платити за будь-яку програму чи послугу, доступ до котрих він може отримати з будь-якого місця в будь-який час через Інтернет з’єднання. Усі програми постачаються та керуються через стороннього постачальника, чий інтерфейс доступний для користувача на клієнтській стороні;
- платформа як послуга (PaaS) – Ця модель надає можливість користувачеві платити за доступ до платформ, дозволяє їм розгортати власні програми та програмне забезпечення в хмарі;
- інфраструктура як послуга (IaaS) – Ця модель надає користувачеві можливість керувати та вести контроль над системами, як над додатками, операційними

системами, сховищами даних. Але модель не потребує контролю хмарної інфраструктури;

- робочий стіл як послуга (Desktop as a Service) – хмарний сервіс, де провайдер розміщує функціонал для роботи з віртуальною інфраструктурою через робочий стіл. У цій моделі провайдер послуг копіює особисті дані замовника на віртуальний робочий стіл і з нього під час процесу входу та виходу з системи. Хоча постачальник хмарних послуг сплачує всі витрати на інфраструктуру та технічне обслуговування, підприємство клієнта все ще несе відповідальність за управління програмами на робочому столі та за безпеку системи;
- мобільний бекенд як послуга (Mobile backend as a service) – також відомий як "бекенд як послуга" (BaaS), являє собою модель надання розробникам веб-додатків та мобільних додатків способів підключення до сховища у хмарному сервері або до інтерфейсу програмних застосунків (API). Також ця модель надає такий функціонал, як управління користувачами, відправлення сповіщень та інтеграція зі службами соціальних мереж;
- кероване програмне забезпечення як послуга (Managed software as a service) – відрізняється від традиційного SaaS тим, що включає до своїх послуг оператора програмного забезпечення. Це дозволяє користуватися усіма перевагами сервісу та не витрачати час на освоєння його програмного забезпечення. Користувачі можуть слідкувати за всіма процесами, можуть затверджувати рішення, підтверджувати машинне навчання та доступ до аналітики та звітування. З часом, якщо це їм підходить, користувачі можуть навчитися користуватися програмним забезпеченням відповідно до власного розкладу та потреб. Розгортанням та виконанням займаються партнери з програмного забезпечення.

Однак, незважаючи на високу ефективність у сфері інформаційних технологій, наразі існують багато складнощів та проблем для хмарного обчислення. За останні кілька років кількість пристроїв, підключених до

глобальної мережі, зросла до приголомшливої позначки, що у свою чергу призвело до високого попиту на Інтернет речей у всьому світі.

Централізовані хмарні центри обробки даних часто не здатні працювати з мільярдами географічно розподілених пристроїв IoT. Ці центри хмарного обчислення не здатні надавати послуги у режимі реального часу і це часто призводить до перевантаженості мережі та високої затримки під час передачі даних. Оскільки хмарні центри даних розташовані біля основної мережі, то всі програми та служби, що розташовані на далекій відстані від цих центрів, будуть зазнавати неприйнятної затримки у часі під час передачі інформації, тому що передача буде здійснюватися через декілька шлюзів [2].

У архітектурі хмарного обчислення вся обробка та обчислення даних, зібраних певним вузлом, виконується на центральному сервері та зберігається у центральному сховищі даних (див. рис. 1.1).

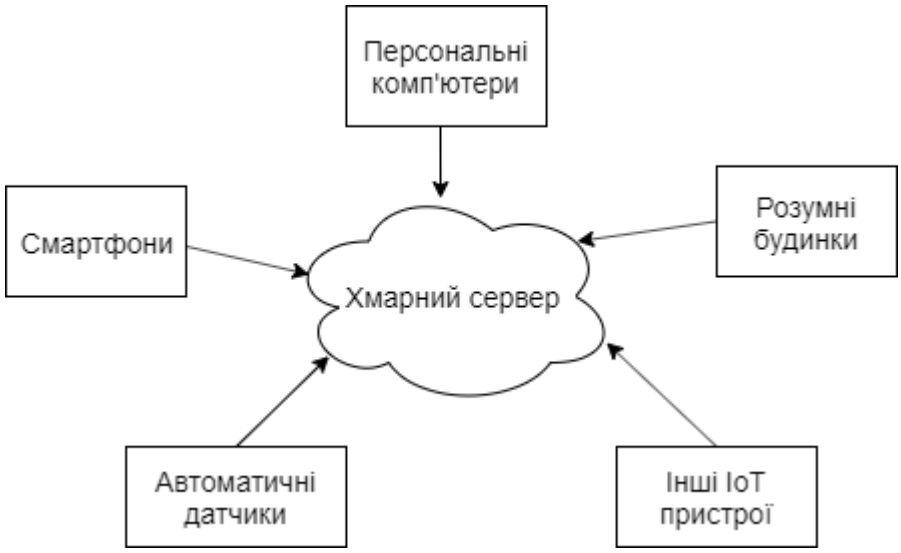

Рисунок 1.1– Використання хмарного сервісу для передачі даних з IoT-пристроїв

Це вимагає великої кількості часу, оскільки дані повинні бути передані з вузла на центральний сервер, перш ніж обробка даних буде здійснена на сервері. Крім того, не практично передавати терабайти даних з вузла мережі на хмару і

назад. Також одним із основних недоліків хмарного обчислення є затримка, викликана взаємодією між пристроєм IoT та самим хмарним середовищем.

Для того, щоб мінімізувати недоліки хмарного обчислення при роботі з великою кількістю даних, була створена технологія для розширення хмарної архітектури під назвою туманне обчислення (Fog computing).

Туманне обчислення – це архітектура обробки даних, що складається з групи розподілених серверів-вузлів та баз даних, клієнтами для яких є декілька підключених різнорідних пристроїв (граничних пристроїв) [3].

Різницею між хмарним та туманним обчисленням є те, що перше, як правило, проводяться в сервісному центрі обробки даних, при цьому дані розподіляються з більш-менш централізованих ресурсів (що виконують обчислення або зберігання) до споживачів на границях мережі, а друге висуває операції над даними близько до джерел, з яких ці дані надходять (від користувачів, або IoT пристроїв) [4].

Туманне обчислення це розподілена обчислювальна парадигма, яка дозволяє використовувати функціонал, що надає хмарна архітектура, на віддалених вузлах мережі (edge nodes). Це полегшує обчислення, зберігання та швидкість передачі даних між пристроями на кінцях мережі та традиційними хмарними серверами. Замість того, щоб запускати програми лише на хмарі, створюється додатковий шар у архітектурі системи, на якому і відбуваються основні обчислювальні операції перед тим як відправити результати обчислення до основного хмарного серверу для зберігання інформації. Цей додатковий шар туманного обчислення являє собою набір обчислювальних машин, зв’язаних в один мережевий вузол у глобальній мережі, який отримує всю інформацію від граничних пристроїв (edge devices). Туманне обчислення поліпшує процес передачі та обробки даних саме тому, що знаходяться переважно близько до граничних пристроїв і затримка при передачі даних незначна. Також туманна архітектура використовує хмарні ресурси лиже за потребою, що дозволяє

фільтрувати всю непотрібну інформацію і відправляти на хмарний сервер значно меншу кількість даних, зменшуючи трафік [5].

У туманному обчисленні і управління і зберігання даних здійснюються розподілено, бо зберігання всієї кількості даних у хмарі не є реалістичним через високу затримку, високі навантаження на трафік між граничним пристроєм та хмарою та високі навантаження на зберігання такої кількості даних. Крім того, важливим параметром, що розглядається, є мобільність граничних пристроїв, яка вимагає мобільності туманних серверів для ефективної роботи запитів. Розподіляючи сховища на туманному шарі існує певна залежність від різних параметрів, таких як географічне розташування граничних пристроїв та споживачів. Це рішення все ж вимагає центрального управління від хмарного серверу у поєднанні з розподіленими сховищами даних у туманних вузлах. Зокрема, з часом можна налаштовувати та змінювати методи балансування навантаження, базуючись на метриках, що були зібрані після початку розподілення серверів та сховищ по різним туманним вузлам, а також на певних потребах користувачів цих туманних сервісів.

При використанні туманної архітектури, обробка даних повністю здійснюється у вузлі, якщо для даних не потрібна більша обчислювальна потужність і виконується частково, якщо для даних потрібна велика обчислювальна потужність. Після чого дані передаються на центральний сервер для решти обчислень або ж безпосередньо для зберігання у хмарному сховищі. Це значно скорочує час обробки даних і підвищує ефективність обчислень у декілька разів, оскільки центральний сервер не перевантажений. Така архітектура досить корисна у географічно розділених районах, де зв’язок може бути нестабільним.

Туманне обчислення пропонує використовувати так звані туманні вузли (Fog nodes), які у собі містять компактно розподілені точки збору даних [6]. Ці вузли також містять інформацію про своє місцезнаходження, що є однією з основних відмінностей туманної архітектури від хмарної.

Таке розподілення ресурсів дозволяє чітко направляти певний вузол на вирішення локальних задач та сфокусувати на це всі наявні обчислювальні ресурси. Замість використання масштабних дата центрів для зберігання інформації, кожен з мережевих вузлів туманної архітектури може містити локальне сховище даних, для тимчасового зберігання інформації. Операції над даними, що потрапляють у вузол із пристроїв, проводяться з мінімізованою мережевою затримкою, а результат обчислення, або фільтрації зберігається в цих локальних сховищах. Після того, як певний цикл роботи над даними завершується, дані з цих сховищ можуть бути передані на головний сервер для зберігання в глобальному централізованому сховищі (див. рис. 1.2).

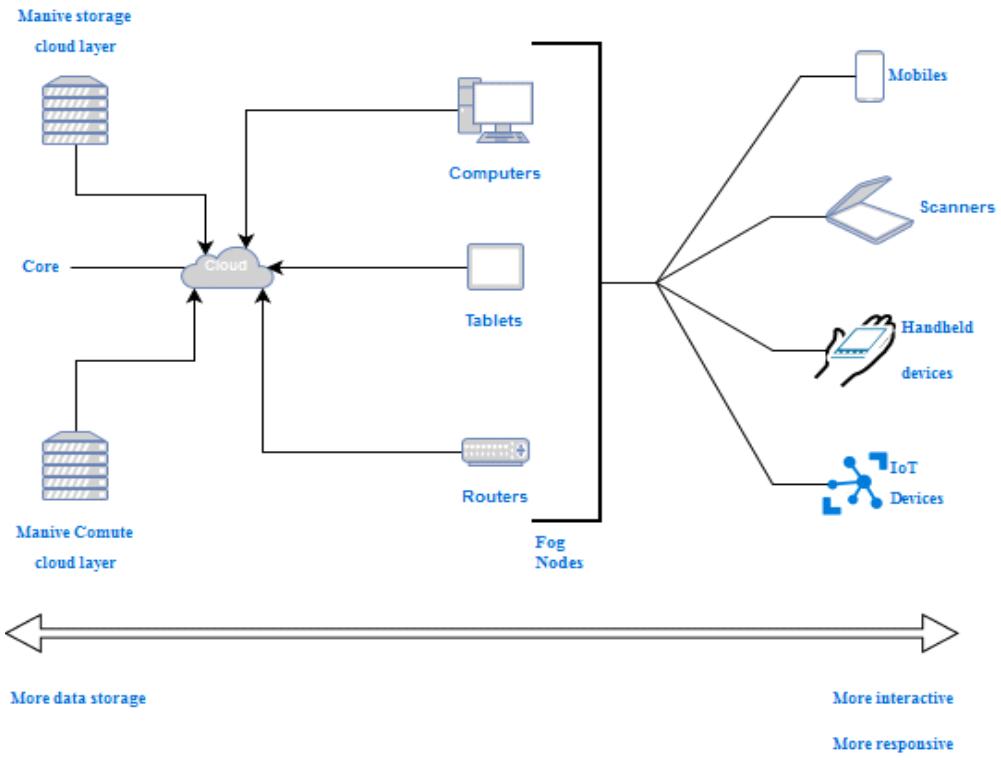

Рисунок 1.2. Схема туманної архітектури

Таке рішення мінімізує вірогідність потрапляння на хмарний сервер не важливих даних, передача яких зайняла б велику кількість часу, якщо б шар туманного обчислення був відсутній у архітектурі.

Пристрої та комп’ютери, що задіяні в туманному шарі, зберігають у собі лише ту логіку, яка необхідна у даному локальному вузлі [7]. А локальні сховища після фільтрації всієї кількості даних, отримують у результаті лише ту частину інформації, яка є важливою та найчастіше використовується. Об’єм цих даних значно менший, ніж той, що в результаті буде зібраний у центральному хмарному сховищі. Це допомагає туманному шару не зберігати ті дані, які використовуються не часто, а в будь який момент зробити запит на отримання певних даних від основного сховища.

Мотивація використовувати такий підхід у тому, щоб зробити потік даних у різних доменах максимально ефективним та безперебійним. Туманне обчислення можна розглядати як проміжний шар між кінцевими користувачами та хмарним сервером, який легко з'єднує цих користувачів і тим самим зменшує затримку при наданні послуг.

Ролі різних об'єктів, що беруть участь у туманному обчисленні, описані нижче:

– користувач – знаходиться на найнижчому рівні туманної архітектури. Користувач отримує дані із найближчого децентралізованим вузла шару туманного обчислення, який здатен надавати лише локалізовану інформацію. Але, якщо потрібні дані недоступні на туманному сервері, то такі дані мають бути передані із центрального хмарного сервера. Треба зауважити, що дані, які передаються, можуть бути не конфіденційні. Але архітектура повинна мати додатковий криптографічний механізм для захисту даних на рівні туманного вузла. Замість цієї ролі в IoT архітектурі присутні граничні пристрої, або автоматизовані контролери, які призначені для збору певних метрик та даних і подальшого їх відправлення для аналізу;
– туманний сервер – це проміжний рівень між користувачем та шаром хмарних сервісів. Він призначений в основному для обчислювальних цілей. Зазвичай він складається з мережевих туманних вузлів у вигляді маршрутизаторів, проксі-серверів, персональних обчислювальних машин тощо. Тут туманні

вузли отримують дані від користувачів або граничних пристроїв для обробки, фільтрування, криптоопераций або інших маніпуляцій бізнес-логіки. Після цього дані або зберігаються у локальній базі даних на тимчасовий період, або ж напряму відправляються до хмарного серверу. Цей рівень також необхідний для передачі інформації від основного хмарного серверу до користувачів, якщо ім необхідна інформація, якої в даний момент немає у базі даних цього туманного серверу;

– хмарний сервер – є найвищим рівнем типової архітектури туманного обчислення. Він являє собою один або групу дата центрів, які мають можливість зберігати всі важливі дані, зібрані з хмарних вузлів (серверів). Хмарні бази даних на цьому рівні повинні мати велику ємність для зберігання великого об'єму інформації;

Основними перевагами використання туманної архітектури можна назвати:

– зберігання необхідної інформації поруч з граничними користувачами. Для того, щоб мінімізувати затримки при передачі через мережу великої кількості даних від або до хмарного серверу, туманний сервер дозволяє зберігати інформацію, яка знадобиться через короткий час, у локальній базі даних;
– широке географічне поширення. Туманне обчислення створює граничні мережі, які дозволяють розширювати площу впровадження послуг хмарними серверами. Такий вид інфраструктури допомагає швидше обробляти та аналізувати велику кількість даних паралельно у через декілька потоків глобальної мережі;
– висока ефективність обробки даних. Через величезну кількість інформації, яка надходить з автоматизованих контролерів або з пристроїв клієнтів, можливість розподіляти навантаження на мережу та на обчислювальні ресурси є дуже значущою. Туманні системи здатні аналізувати, обробляти та фільтрувати інформацію з граничних пристроїв у багатопоточному режимі, знаходячись на невеликій відстані від джерела передачі. Це дозволяє значно знизити навантаження на мережу під час передачі результатів до хмарних серверів,

оскільки передаються лише важливі дані через бажані інтервали, у необхідні проміжки часу;

– зменшення об’єму зберігаємої інформації у хмарному сховищі. Шар туманної архітектури дозволяє попередньо фільтрувати та обробляти значний об’єм даних, перед тим, як відправити ці дані до хмари.

1.2 Виявлення проблем та актуалізація рішень

Якщо говорити про переваги використання туманної архітектури перед хмарною, то однією з основних буде можливість використовувати розподілені сховища даних у спеціально відведених вузлах, замість використання одного головного сховища даних для всіх клієнтів. Саме можливість розподіляти навантаження на обробку та зберігання інформацію і допомагає підвищувати продуктивність при роботі з великою кількістю одночасно підключених пристроїв.

Кожний окремий вузол містить в собі сервер для обчислення і базу даних для зберігання інформації, що збирається у конкретній місцевості, для якої цей вузол був відділений. Але створення та підтримка такої розподіленої архітектури серверів та сховищ є досить складним завданням на даний час.

Незважаючи на те, що туманна архітектура є найбільш підходящим рішенням для розподілення навантаження при роботі з великою кількістю клієнтів одночасно та для обчислення даних лише за необхідністю, на даний час існує лише декілька можливих варіантів моделі розподілу навантаження [8]. Трьома основними підходами для створення такого виду архітектури є: модель розвантаження, модель агрегації та peer-to-peer модель.

– у моделі розвантаження (offloading model) дані, що згенеровані з граничних пристроїв, завантажуються в найближчий туманний вузол, а потім у хмарну

базу даних (розвантаження вгору) і у зворотному порядку від хмарного шару до граничних пристроїв (розвантаження вниз);

- у моделі агрегації (aggregation model) потоки даних, згенеровані декількома граничними пристроями, зберігаються та, можливо, обробляються у найближчому туманному вузлі перед завантаженням у хмарний центр обробки даних;
- у peer-to-peer моделі туманні вузли, що знаходяться на невеликій відстані від граничних пристроїв, поєднують їх обчислювальні можливості зі своїми та представляють локальні сховища даних для зберігання тимчасової інформації після обробки. Це значно прискорює роботу з даними та дозволяє використовувати хмарне сховище лише тоді, коли це необхідно, відправляючи запити на збереження даних періодично.

Ще однією проблемою є розміщення групи туманних серверів таким чином, щоб вони надавали максимально широкий спектр послуг пристроям у певній місцевості та були максимально продуктивно-ефективними [9]. Потрібно завчасно спланувати: як зменшити затримки під час передачі та агрегування даних, як не допустити перевитрати обчислювальних ресурсів та з який об’єм даних зберігати на кожному вузлі і коли передавати ці дані до головного хмарного сховища.

Сервери, що виконуватимуть роль туманних вузлів, можуть бути розгорнуті в будь-якому середовищі з мережевим з'єднанням. Шар туманного обчислення має додаткові сховища для зберігання даних на границях мережі для більш продуктивної роботи з клієнтами. Отже, туманному серверу потрібно адаптувати свої послуги для найбільш ефективної роботу у своєму вузлі, що вимагає додаткових витрат на управління та обслуговування. Слід ознайомитися з існуючими варіантами створення моделі з великою кількістю вузлів для обчислення даних з клієнтських пристроїв та з варіантами зниження рівня навантаження на ці вузли при обробці даних.

Для порівняння існуючих рішень та виявлення проблем щодо використання туманного обчислення, наведено приклади систем, що базуються на цій

архітектурі. Електронний ресурс [10] посилається на статтю, у якій наведено декілька прикладів систем, що побудовані з використанням туманного обчислення.

Першою з описаних у статті систем [10] є Energy Lattices – система для підвищення ефективності використання енергії та зменшення рівня забрудненості навколишнього середовища. Міста, що мають декілька джерел енергоресурсів, матимуть можливість вибору найбільш ефективного з них. Рисунок 1.3 демонструє архітектуру, на якій побудована система Energy Lattices.

Рисунок 1.3 – Архітектура системи Energy Lattices

Пристрої, такі як лічильники електроенергії, автоматично переходять на ефективний енергоресурс на основі доступності та найнижчої ціни. Сервери туманного шару зберігають дані, що генеруються підключеними датчиками, і на основі отриманих даних вони відправляють запити на вибір конкретного енергетичного ресурсу.

Дані з датчиків фільтруються та обробляються на обчислювальних машинах туманного вузла, перед тим, як відправити на зберігання до хмарного сховища. Ці

дані передаються кінцевим користувачем через Wi-Fi/Інтернет. Внутрішні дані, які використовуються для вибору відповідних енергоресурсів, зберігаються як приватні дані і належним чином захищені за допомогою криптографічних механізмів.

Наступною системою, що описана в представленому ресурсі [10], є MediFog – система з туманною архітектурою у сфері медицини. У цій сфері час обробки та передачі інформації займає найголовнішу роль, тому що від цього залежить життя пацієнтів. Архітектура системи MediFog наведена на рисунку 1.4.

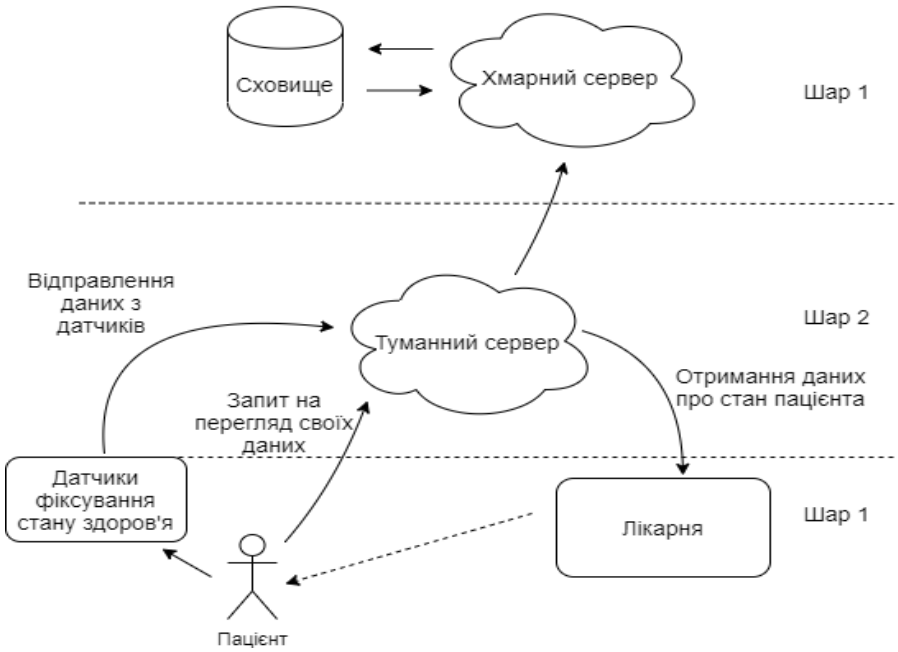

Рисунок 1.4 – Архітектура системи MediFog

Запропонована система використовується для виявлення та випередження будь-якої надзвичайної ситуації, пов’язаної з пацієнтом, наприклад зупинки серця або надмірного підйому та падіння артеріального тиску. Ці дані будуть зафіксовані датчиком і будуть надіслані туманним серверам. Лікарні матимуть доступ до даних пацієнта і зможуть негайно вжити заходів. Пацієнт, який зареєструвався у певній лікарні, може переглядати лише свої дані. Ці дані будуть приватними та потребуватимуть додаткової захищеності на сервері, щоб ніхто

сторонній не отримав до них доступ. Пацієнту не потрібно буде звіряти свої дані з лікарнями особисто, тому що лікарні в будь-який час зможуть зчитати ці дані для вирішення, чи потрібна пацієнту додаткова допомога або огляд в даний момент.

Наступна система в статті [10] має назву FoAgro і являє собою рішення використання туманного обчислення у сфері агрокультури та фермерства. Великі поля потребують значного контролю та управління. Туманна архітектура може запропонувати декілька рішень для вирішення певних проблем у процесах управління фермерськими культурами. На рисунку 1.5 наведена архітектура системи FoAgro.

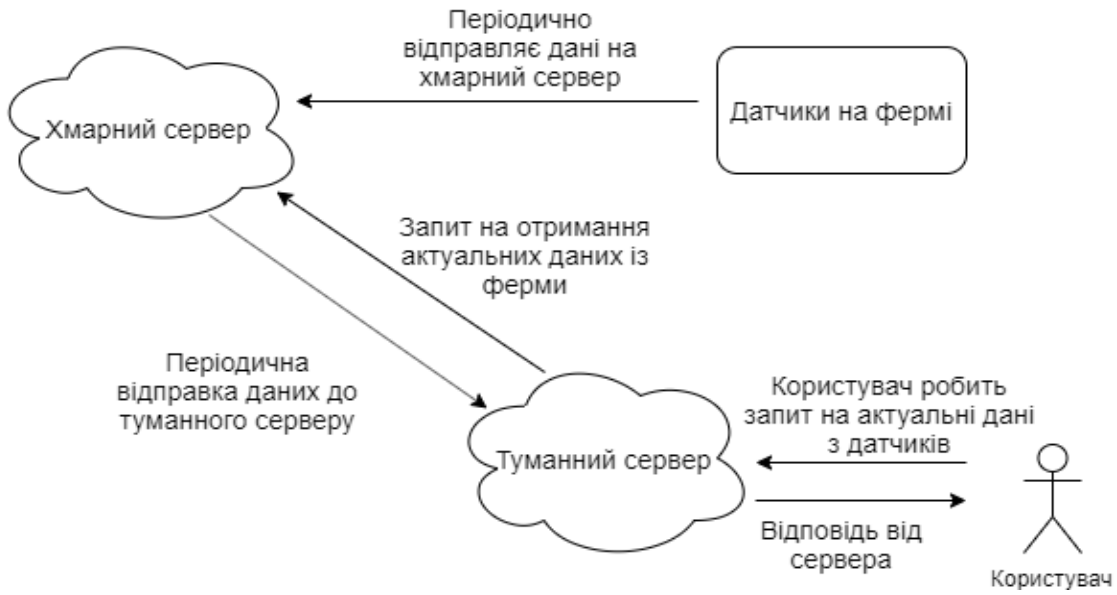

Рисунок 1.5 – Архітектура системи FoAgro

До сенсорів в цій системі належать датчики вологи, датчики з камер та інші Вони розміщені безпосередньо у полі і передають зібрані дані до туманних вузлів. Кожен фермер має доступ лише до тих даних, які були зібрані конкретно з його полів. Він можуть авторизуватися за допомогою своїх особистих даних щоб отримати доступ до даних.

У великих містах сьогодні досить поширені проблеми з паркуванням. Ці проблеми певною мірою були вирішені засобами туманних обчислювальних

ресурсів. На рисунку 1.6 продемонстрована архітектура системи Connected Parking System Architecture, яка також представлена в электронній статті [10].

Рисунок 1.6 – Архітектура системи Connected Parking System Architecture

Тут датчики розміщуються на парковках, а всі дані з них збираються на вузлах відповідних туманних серверів. Ці дані можуть переглядати користувачі, які хочуть знайти вільні місця для паркування. Датчики відображають стан конкретного місця для паркування, зайняте воно або вільне. Користувачі можуть ознайомитися зі статусом вільних місць для паркування за допомогою Wi-Fi або через Інтернет, отримуючи доступ до публічних даних. Ці дані необхідно впроваджувати додатковою безпекою, аби ніхто не зміг їх змінити.

Існуючі рішення дають певну уяву про те, як саме необхідно будувати архітектуру туманного обчислення, та які основні задачі необхідно вирішити під час її організації. Оскільки туманний сервер відповідає за основні обчислення в системі та є місцем взаємодії з клієнтами, однією з основних задач є створення умов, за яких час обробки клієнтських запитів буде мінімальним, а пропускна здатність мережі туманних обчислень не почне зменшуватись під час перенавантаження цієї мережі. Також через те, що основне навантаження від кількості клієнтів буде припадати саме на туманний сервер, при використанні

туманного серверу необхідно регулювати процес обробки даних між декількома серверами обчислення.

1.3 Автентифікація у туманній архітектурі

Туманне обчислення має декілька проблем пов’язаних із безпекою. Основна проблема безпеки - це автентифікація пристроїв, що відправляють чи отримують дані від шару туманного обчислення через різні шлюзи. Зловмисник може використовувати підроблену IP-адресу для доступу до інформації, що зберігається в конкретному туманному вузлі. Вважаючи, що до такого туманного серверу будуть одночасно підключені різнорідні пристрої, налаштування кожної обчислювальної машини та мережі, до якої вона підключена, займатиме значну кількість часу та буде містити в собі ряд складнощів.

Втрата приватних даних є дуже поширеною проблемою під час відправлення деякої інформації через мережу. Оскільки дані, в основному, передаються через бездротове з’єднання, то необхідно впроваджувати додатковий рівень захищеності під час передачі цих даних. Кожен туманний сервер потрібно налаштовувати вручну, і конфігурація безпеки повинна бути впроваджена для кожного вузла у мережі туманної архітектури, а це вимагає додаткових чималих витрат на обслуговування. Кінцеві користувачі та автоматизовані контролери передають дані напряму до певного вузла туманного обчислення. Це призводить до того, що ці вузли обробляють та зберігають у локальних сховищах найбільш важливу інформацію, перед тим, як відправити її на хмару [11].

Кінцеві користувачі або пристрої повинні пройти автентифікацію на стороні туманного серверу, перш ніж отримати доступ до будь-яких ресурсів та користуватися послугами туманного серверу.

Причиною того, що автентифікація займає таку важливу роль у туманній архітектурі, є те, що послуги туманних вузлів надаються одразу великій кількості

користувачів та пристроям різного типу. Крім того, необхідно впроваджувати перевірку прав користувачів на різних рівнях туманного вузла. Традиційна автентифікація на основі публічного ключа недостатньо ефективна і має слабку масштабованість.

Одним із очевидних рішень, які можна впровадити в даній архітектурі, є ідентифікація запитів по каналу обмеженому місцезнаходженням вузла. Такий підхід буде ефективно себе показувати о сценарії роботи з IoT контроллерами, тому що на відміну від користувачів, місцезнаходження контролерів можна обмежити на певній місцевості і налаштувати туманний вузол на обробку лише тих запитів, що надходять з певного типу пристроїв.

1.4. Безпека даних у туманному шарі

Зберігання даних або на сервері, або в хмарі не є дуже складною проблемою для вирішення. Але впровадження захищеності цих даних є відкритим і досить складним питанням. У контексті туманного обчислення, дані зберігаються у базі даних туманного сервера для подальшого використання. В основному, ці дані представляють собою важливу конфіденційну інформацію, тому безпека сховища даних повинна бути на високому рівні і відповідати стандартам сучасності.

По-перше, важко забезпечити цілісність даних, оскільки дані, що передаються, можуть бути втрачені або неправильно модифіковані. По-друге, завантажені дані можуть бути викрадені і використані в інтересах зловмисників.

Для вирішення цих проблем можна використати методи, описані нижче.

Гомоморфне шифрування – це форма шифрування з додатковою можливістю оцінювання для обчислення зашифрованих даних без доступу до секретного ключа. Результат такого обчислення залишається зашифрованим. Поєднання гомоморфного шифрування та шифрування з можливістю пошуку

(клас криптографічних алгоритмів шифрування, в яких існує можливість здійснювати пошук по зашифрованих даних) допомагають впровадити конфіденційність та цілісність даних для зберігання у туманних та хмарних сховищах без дешифрування [12]. Публічний аудит, що зберігає конфіденційність даних, зберігаємих у хмарі, може використовувати сторонній аудит третьої сторони (Third-party auditor – TPA), використовуючи гомоморфний автентифікатор та техніку випадкових масок для захисту конфіденційності від втручання стороннього аудита. Для забезпечення надійності зберігання даних у сховищі, сервер, що пов'язаний з цим сховищем може використовувати мережеве кодування при запобіганні пошкодження даних та для відновлення їх відновлення у майбутньому.

Іншим важливим питанням в туманному обчисленні є впровадження безпечного та конфіденційного процесу обчислення та обробки даних, що передаються на туманні вузли.

Якщо необхідно впроваджувати додаткову перевірку результатів обчислення серверу для того, щоб у майбутньому корегувати логіку обчислення, то потрібно додавати додаткові сервери для верифікації цих результатів. При завантаженні даних з туманного сервера до цих сторонніх серверів, можна зіткнутися з ситуацією, коли деякі з них будуть не довіреними. Сторонні сервери запрограмовані оцінювати обчислювальні функції та повернути результат, що доведе чи спростує, чи коректно вони виконують обчислення над даними. Оскільки дані, зібрані туманними сервісами, часто несуть в собі важливу конфіденційну інформацію від користувачів або від автоматизованих датчиків, то верифікацію результатів обчислення цих даних потрібно проводити на цьому ж туманному вузлі, щоб підтримувати безпеку даних на високому рівні [13].

Щоб захистити дані під час виконання операцій пошуку над ними, вони попередньо повинні бути зашифровані перед передачею до туманного вузла. Тоді проблема, як виконувати пошук за зашифрованими даними, може бути вирішена використовуючи шифрування з можливістю пошуку (searchable encryption).

Приватність особистих даних користувача під час їх завантаження на сервер туманного вузла є додатковим питанням під час проектування роботи туманної архітектури. Наприклад, при роботі з системами розумних будинків, туманний сервер буде отримувати всю інформацію щодо того, коли в цих будинках немає мешканців або коли вмикається телебачення або світло. Поширення інформації такого роду є абсолютним втручанням в приватне життя користувачів. Одним з існуючих рішень є використання сторонніх сервісів, як аудитів, але, як було описано вище, ніщо не гарантує збереження інформації під час передачі даних до цих сервісів.

Одне з найпростіших рішень полягає в тому, що клієнт може генерувати фіктивні завдання для генерації додаткових фіктивних даних і подальшого відправлення їх до декількох туманних вузлів, приховуючи точні реальні дані [14]. Однак це рішення збільшить витрати клієнта на ресурси та енергію. Іншим рішенням буде розробка способу моніторингу того, що після потрапляння даних на сервер, вони одразу ж проходять алгоритм шифрування перед тим, як їх буде використано на цьому сервері.

Під час роботи з інфраструктурою Інтернету речей, приватною інформацією є дані щодо місцезнаходження самих IoT пристроїв. Ці пристрої зазвичай відправляють результати своєї роботи до найближчого туманного вузла і цей самий вузол володіє інформацією про те, які пристрої знаходяться поруч і далі від інших таких вузлів. Більш за те, коли дані передаються від датчика до декількох вузлів одночасно, то це дозволяє розрахувати траєкторію передачі даних та виявити місцезнаходження датчика. Коли у ролі датчика виступає IoT пристрій користувача, або деякий важливий об’єкт, то її розташування є важливою персональною інформацією, яка знаходиться під загрозою.

Єдиним методом збереження інформації про місцезнаходження є приховування ідентичності таким чином, що навіть незважаючи на те, що туманний сервер знає, що відправник знаходиться поруч, він не може його ідентифікувати. Однією із можливостей для реалізації цього методу є

використання сторонніх сервісів для генерації підроблений ідентифікатора для кожного користувача.

1.5 Постановка задачі

Базуючись на дослідженні існуючих систем, що використовують туманне обчислення, на перевагах та недоліках архітектури, можна описати певні вимоги до побудови системи розподілених обчислень з великою кількістю споживачів та граничних пристроїв.

Кожен туманний вузол може складатися з одного або декількох пристроїв, створюючи таким чином віртуальний вузол для підтримки області покриття. Ці пристрої можуть бути маршрутизаторами, комутаторами, шлюзами або розгорнутими локальними серверами.

Граничні пристрої з'єднані з шаром туманного обчислення, який, в свою чергу, підключений до централізованого шару хмарного обчислення. Цей тип з'єднання формує ієрархічну платформу для обчислення, де на вищому шарі зберігається сервер найвищої потужності зі сховищем найвищої ємності. На нижчому рівні навпаки існує група пристроїв та серверів з меншою обчислювальною потужністю та сховищ з нижчою ємністю, які налаштовані на роботу з певною областю та з певними граничними пристроями.

Шар туманного обчислення може бути розподілений у різних географічних районах, щоб охопити широку територію в одну масштабну мережу. Клієнтські пристрої, такі як телефони, датчики та контролери, підключені напряму до шлюзу туманного вузла та отримують послуги з низькою затримкою ніби як від хмарного серверу. Така затримка можлива через одну з основних переваг туманного обчислення, а саме через близьке розташування вузлів до граничних пристроїв у своїй області. Для порівняння, хмарному серверу знадобилося б

значно більше часу для того, щоб обробити запит від клієнта та відправити йому відповідь.

У великомасштабних мережах, де працюють одночасно багато пристроїв та серверів, стратегії розподілення даних здатні позбутися обмежень сховища даних шляхом розповсюдження даних рівномірно між усіма вузлами [15]. Оскільки сервери в цих вузлах можуть раптово вийти з ладу, то необхідно додатково впроваджувати методи реплікації даних між серверами та сховищем даних та дотримуватись правил консистентності даних у системі. Для того, щоб підтримувати високу швидкість обробки та обчислення даних у всій інфраструктурі та в кожній локальній місцевості окремого вузла, потрібно впровадити алгоритми врівноваження навантаження.

Задачею даної роботи є моделювання системи, побудованої з використанням архітектури туманного обчислення та дослідження методів застосування такої архітектури при роботі зі сховищами даних, а також опис переваг та недоліків її використання у порівнянні з хмарним обчисленням. Необхідно створити модель розподілення даних між декількома сховищами в різних туманних вузлах та описати методи доступу до цих сховищ. Також слід порівняти шляхи розвантаження трафіку на хмарний сервер під час потоку великої кількості даних та змоделювати підхід паралельного обчислення інформації з крайових пристроїв на туманних серверах.

Перейдемо до опису моделі дослідження в наступному розділі.

2 ОПИС ТЕОРЕТИЧНИХ ДОСЛІДЖЕНЬ

2.1. Порівняння існуючих видів архітектури

На огляд візьмемо найбільш поширені характеристики ефективності для туманного обчислення є:

- зменшення затримки під час передачі даних або отримання результатів від сервера за рахунок зменшення відстані від граничних пристроїв до серверу;
- зменшення рівня навантаження процесора хмарного серверу, оскільки основна частина обчислювальних операцій проводиться на туманному вузлі;
- підвищення швидкості обробки транзакцій у хмарному сховищі, шляхом розташування додаткової бази даних у туманному вузлі. Така база даних бере участь у тимчасовому зберіганні частини даних, що надходять до вузла і тим самим хмарне сховище не потребує безперервної обробки та зберігання даних під час постійного надходження запитів від клієнтів;
- підвищення стабільності роботи системи за рахунок розширення кількості туманних серверів, що можуть підмінити один одного під час виникнення виключних ситуацій.

Для дослідження цих метрик ефективності та способів застосування туманної архітектури, необхідно описати існуючі варіанти побудови високонавантажених та широкомасштабних систем.

Як було описано в першому розділі, хмарне обчислення використовує метод обміну даними напряму між хмарними сервером та користувачами. Обробка запитів та вся логіка роботи системи виконується на самому хмарному сервері, а отримані дані зберігаються в хмарному сховищі. Ця архітектура наразі дуже поширена і використовується у великій кількості систем через свою простоту розгортання та управління у порівнянні із іншими видами архітектури. Але статичність серверів цієї моделі, неможливість географічного розподілення та ефективного масштабування обчислювальних ресурсів та

ресурсів мережі даної моделі робить її не підходящою для сучасних стандартів роботи зі швидко зростаючою кількістю клієнтів та підключених пристроїв.

На даний момент, окрім хмарного обчислення, до основних видів побудови високонавантажених та широкомасштабних систем відносяться також граничне та туманне обчислення (див. рис. 2.1).

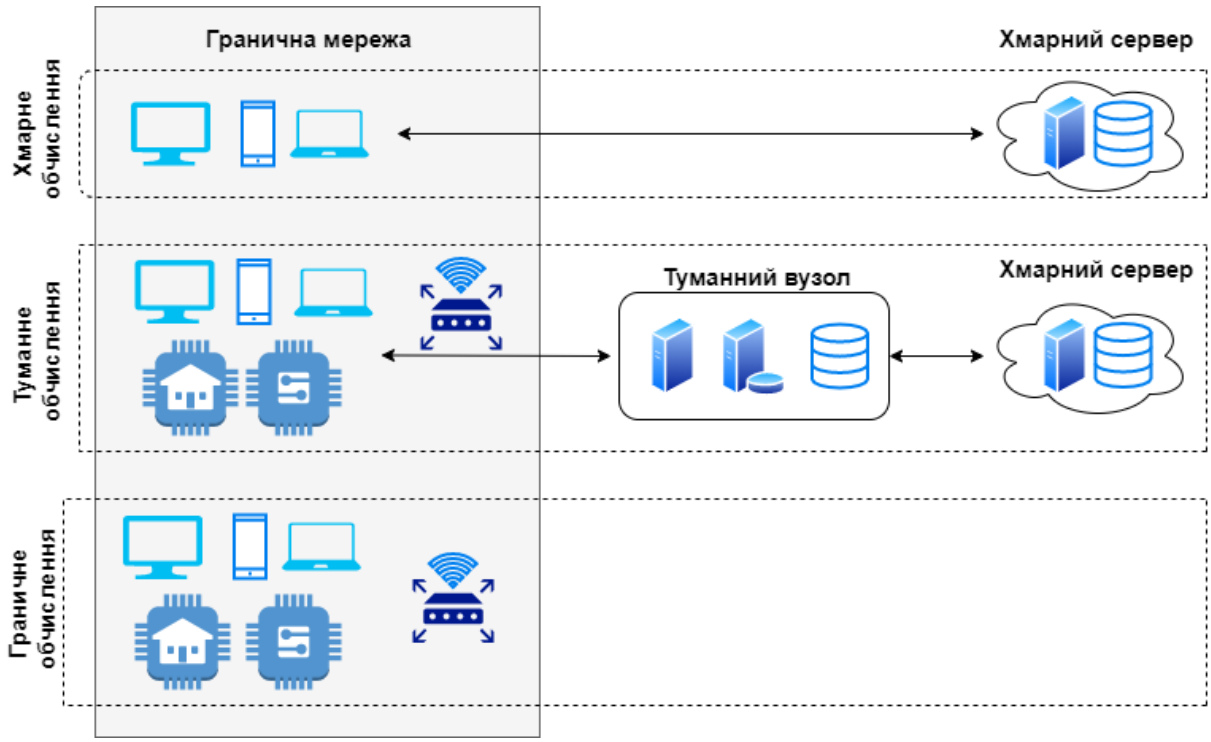

Рис. 2.1. – Види архітектур обчислення

Граничне обчислення являє собою архітектуру, у якій всі операції над даними відбуваються на самому пристрої, до якого під’єднано датчик або на шлюзовому сервері, що знаходиться поруч із датчиками, у той час, коли у туманному обчисленні операції виконуються у туманному вузлі, що розташовується або в межі локальної мережі пристроїв або навіть далі [16]. Також граничні обчислення відрізняються від інших тим, що дані після обробки не передаються далі по мережі.

Перевагами цього підходу є те, що дані обробляються на тому ж пристрої, звідки вони були отримані, і тим самим захищені від зловмисників, бо

їх не можна підмінити або викрасти із мережі. Також інформація із датчиків завжди актуальна, бо обробляється та аналізується в режимі реального часу, на момент отримання. Але водночас неможливість передавати дані для подальшого зберігання у хмарному або туманному сховищах і є найголовнішим недоліком даного підходу. При використанні цієї архітектури неможливо проводити глибокий аналіз даних на центральних серверах, бо дані присутні лише в місці отримання і не зберігаються надовго після опрацювання.

Також слід брати до уваги, що граничне обчислення зазвичай використовується при роботі з пристроями IoT, а вони мають не дуже високу потужність та обмежені обчислювальні можливості. Саме тому, що деякі системи потребують додаткового функціоналу та можливостей щодо роботи з даними IoT пристроїв, для цих самих систем необхідно впроваджувати сервери, які б брали на себе цю частину функціоналу. Також, ці сервери можуть бути використані для загального управління підпорядкованої мережею цих пристроїв, та налаштовувати певні підмережі на специфічні завдання. Граничне обчислення не дозволяє впроваджувати додаткові сервери для такого функціоналу, що робить цей підхід дуже специфічним та налаштованим лише на певну область задач.

Можна побачити, що туманне обчислення в деякому сенсі поєднує два інші підходи, додаючи прогалину у вигляді додаткового шару архітектури. Це розширює граничні обчислення в тому плані, що тепер дані можуть бути передані до сховища і у подальшому це надасть користувачам системи можливість зробити запит на серверну частину та отримати інформацію, яка була відправлена та опрацьована деякий час назад.

Хмарне обчислення тепер за допомогою додаткового шару перестають бути залежними від віддаленості від користувачів, а їх обчислювальні ресурси можуть бути розподілені та здатні покривати широку територію. У контексті роботи з постійно зростаючою кількістю онлайн користувачів або з потоком інформації, об’єм якого постійно збільшується, можливість розділяти основний

обчислювальний сервер на ієрархію розподілених серверів та використовувати їх ресурси одночасно в кількох географічних областях є дуже затребуваною. Саме ці особливості і роблять технологію туманного обчислення дуже актуальною на даний момент.

2.2. Модель архітектури з використанням туманного обчислення

Для того, щоб реалізувати концепт туманного обчислення і почати вимірювати рівень ефективності даної архітектури у порівнянні з хмарним обчисленням, потрібно спочатку описати модель, яка б відображала взаємодії між шарами архітектури.

Така модель представляє основні три шари взаємодії туманного обчислення. Хмарний рівень є основним в архітектурі такого типу, оскільки вся інформація зберігається у хмарних сховищах даних, після того, як була оброблена туманними серверами. Хмарні сервери відповідають за керування роботою туманних вузлів, слідкують за їх стабільністю та можуть збирати метрики ефективності їх роботи, наприклад час обробки даних певного об’єму, ефективність роботи алгоритмів сортування даних та інше.

Незважаючи на те, що кількість одночасно підключених до системи клієнтських пристроїв може бути досить значною, хмарний рівень та його сервери не будуть схильними до перенавантаження або до великих затримок мережі [17]. Ці сервери не приймають участі в обробці всього обсягу даних, який задіяний у системі, бо цю задачу виконують туманні вузли та їх проксі-сервери (див. рис. 2.2). Кожен з хмарних серверів повинен бути налаштований лише на базові задачі щодо керування інфраструктурою, за яку він відповідає, та повинен періодично приймати запити від туманного шару для збереження прийнятих даних.

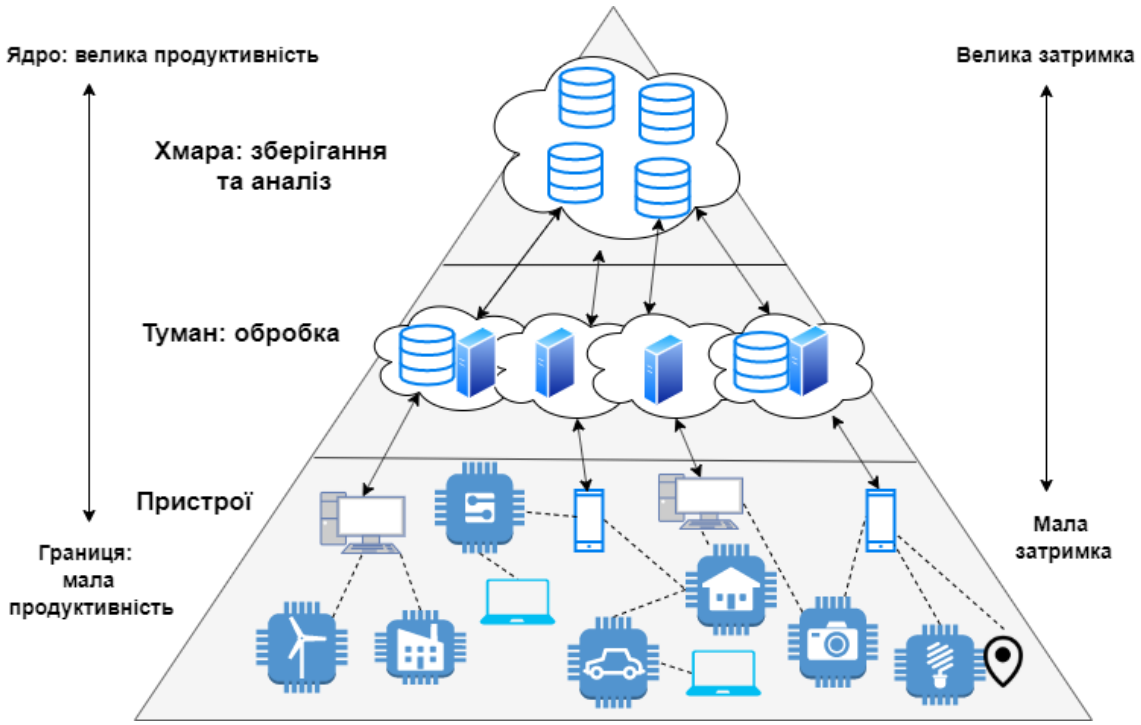

Рисунок 2.2 – Структура туманного обчислення.

Об’єм цих даних буде значно меншим у порівнянні з тим, який надходить до туманного шару, тому що дані попередньо були оброблені на туманних вузлах і передані до хмарного сервера на кінцеву обробку та на відправлення до головних хмарних сховищ даних.

Сховища даних на хмарному рівні повинні зберігати основну інформацію системи і у разі необхідності, передавати цю інформацію назад до туманних вузлів, коли вона знадобиться користувачам системи. Також дані з цих сховищ можуть бути використані для машинного навчання (якщо це є однією зі сфер застосування системи), оскільки вони є остаточно обробленими, згрупованими та структурованими.

Якщо говорити про рівень туманного обчислення (який є рівнем-посередником даної архітектури), то він є тією унікальною особливістю, яка відрізняє цю архітектуру від архітектури хмарного обчислення, або архітектури граничних (мобільних) обчислень [18]. Цей рівень відповідає за комунікацію та передачу даних між граничними користувачами та хмарними сховищами.

Оскільки основним атрибутом серверів-вузлів цього шару є те, що вони загалом знаходяться порівняно близько до кінцевих пристроїв-клієнтів, то і затримка і час обміну інформацією між цими рівнями повинні бути відносно низькі. Бізнес-логіка серверів цього рівня повинна відповідати за отримання та обробку інформації від клієнтів, а також її зберігання у локальному сховищі. Саме тому, потрібно враховувати, що при великій кількості одночасно підключених пристроїв, зростає і час обробки даних, що надходять з них. Саме тому, одного або іноді і двух обчислювальних машин може бути недостатньо для ефективної роботи і ця кількість напряму залежить від кількості потенційних клієнтських пристроїв.

Основною задачею цього шару є зменшення загального рівня навантаження на шар хмарного обчислення шляхом розподіленої роботи туманних вузлів. Коли система націлена на роботи зі значною кількістю даних та їх збереження, то кожен з вузлів цього рівня повинен бути оснащений власним локальним сховищем даних для попереднього тимчасового зберігання. Доступ до цих сховищ мають як і користувачі система на граничному рівні, так і сервери хмарного рівня.

Розподілення вузлів є ключовим елементом при плануванні архітектури туманного обчислення [19]. Одного такого вузла може бути недостатньо для розвантаження хмарного центру. Платформа туманного обчислення пропонує 2 види розподілення вузлів: горизонтальне та вертикальне. Перше пропонує збільшувати кількість вузлів для розподілення навантаження рівномірно між ними. Друге базується на тому, що весь шар туманного обчислення буде розділено на декілька шарів з різними обчислювальними потужностями: сервери з обмеженими обчислювальними ресурсами розташовуються на границі мережі (ближче до користувачів) та не виконують складних операцій, а лише первинно обробляють дані; сервери більшої потужності відповідають за складніші операції по обробці та зберіганню даних та за відправлення цих даних до хмарного шару.

Третій рівень складається з клієнтських пристроїв, що контактують напряму з туманними вузлами. До цих пристроїв можуть відноситися: смартфони, комп’ютери, автоматичні контролери, датчики та різні види IoT пристроїв.

2.3 Опис теоретичних досліджень

Щоб з’ясувати, в яких випадках, у якому вигляді та з якою кількістю одночасно обробляємих клієнтських пристроїв потрібно використовувати туманну архітектуру, потрібно побудувати її модель та порівняти з моделлю хмарного обчислення, які на даний момент є дуже поширеними та популярними. Основною різницею між цими моделями є наявність додаткового туманного шару, реалізованого за допомогою туманних вузлів-серверів.

Оскільки необхідно провести додатковий аналіз інформації, та за запитами клієнтів відправляти дані, які б зберігалися на серверах, потрібно впровадити сховище на кожному рівні архітектури. Основною метою дослідження є опис того, як буде керуватися потік даних між сховищами різних шарів, та як буде впроваджено доступ до кожного сховища даних під час запиту від клієнта або під час вилучення інформації серверною стороною для аналізу. Саме через необхідність використання сховищ даних, методологія граничного обчислення, що описана у попередньому розділі, не може бути використана для дослідження, оскільки вона не потребує зберігання даних після їх обробки або відправки даних на інші сервери. Тому, для вимірювання ефективності роботи архітектури, основним аналогічним рішенням будуть саме хмарні обчислення.

Час затримки при передачі даних повинен бути зменшеним, при порівнянні з хмарним обчисленням, тому що саме висока затримка при передачі інформації від клієнта до хмари є одним з основних недоліків хмарної

архітектури. Використання туманних вузлів на невеликій відстані від місця отримання даних дозволить знизити час передачі запитів та відповідей між клієнтською та серверною частинами. Також слід врахувати рівень споживання пропускної здатності під час використання туманних вузлів, а саме розподілити потоки даних між туманним та граничним рівнем таким чином, щоб знизити частоту запитів та об’єм передаваємої цими запитами інформації.

Також слід порівняти надійність кожного з підходів у плані стабільності роботи транзакційних операцій та безпеки передачі даних між серверами. Сервери та бази даних повинні мати механізм відновлення та реплікації даних, щоб мінімізувати випадки втрати важливої інформації.

Досягнувши описаних критеріїв ефективності та порівнявши метрики туманного обчислення з хмарним, можна буде зробити висновок щодо ефективності використання такої моделі архітектури. Для вирішення задачі порівняння цих метрик, необхідно відтворити основні моделі архітектури та провести їх тестування.

У наступному розділі наведемо опис побудови моделей дослідження.

3 РЕЗУЛЬТАТИ ТЕОРЕТИЧНИХ ДОСЛІДЖЕНЬ

3.1. Опис дослідження моделей архітектури

Для визначення ефективності роботи туманної архітектури, її переваг та недоліків у порівнянні саме з хмарним обчисленням, необхідно зібрати основні метрики під час навантаження системи. З цією метою, для порівняння необхідно взяти чотири основних сценарії побудови архітектури взаємодії серверів та сховищ даних.

Граничними вузлами в цих сценаріях будуть персональні комп'ютери, та ноутбуки, які емулюють роботу автоматизованих датчиків, що генерують блоки даних та відправляють запити з цими даними до серверів. Для роботи з хмарними сервісами будуть використані веб-сервіси Amazon (AWS – Amazon Web Services). Роль хмарного серверу буде виконувати віртуальна машина сервісу EC2 (Elastic Compute Cloud), а для сховища буде використана реляційна хмарна MySQL база даних.

Оскільки дослідити повноцінну розгорнуту модель туманної архітектури неможливо через високу складність побудови декількох серверів та баз даних у хмарному середовищі, а також через відсутність великої кількості обчислювальних машин для туманного сервісу, то всі тестові сценарії будуть проведені використовуючи спрощену модель взаємодії обчислювальних машин та сховищ для хмарного та туманного шарів. Цієї кількості буде достатньо для демонстрації можливостей туманного середовища та для отримання необхідних обчислювальних метрик, які будуть збільшуватися прямо пропорційно збільшенню кількості серверів.

Для кожної з моделей необхідно провести два види дослідження, а саме: реакцію системи на відправлення запитів з великим об'ємом даних з нечастими інтервалами від декількох клієнтської машин, а також навантажене тестування системи під час безперервних одночасних запитів від групи клієнтських машин.

Для тестування в однакових умовах, кількість зберігаємих об’єктів на сервері буде обмежена до сорока тисяч для кожної з моделей.

Для першої моделі необхідно створити найпростіший варіант архітектури системи, а саме використання хмарного серверу для основних операцій та бази даних для збереження інформації (рис. 3.1). У якості хмарного сервера виступає EC2-t2.micro, а у якості сховища даних RDS t2.micro.

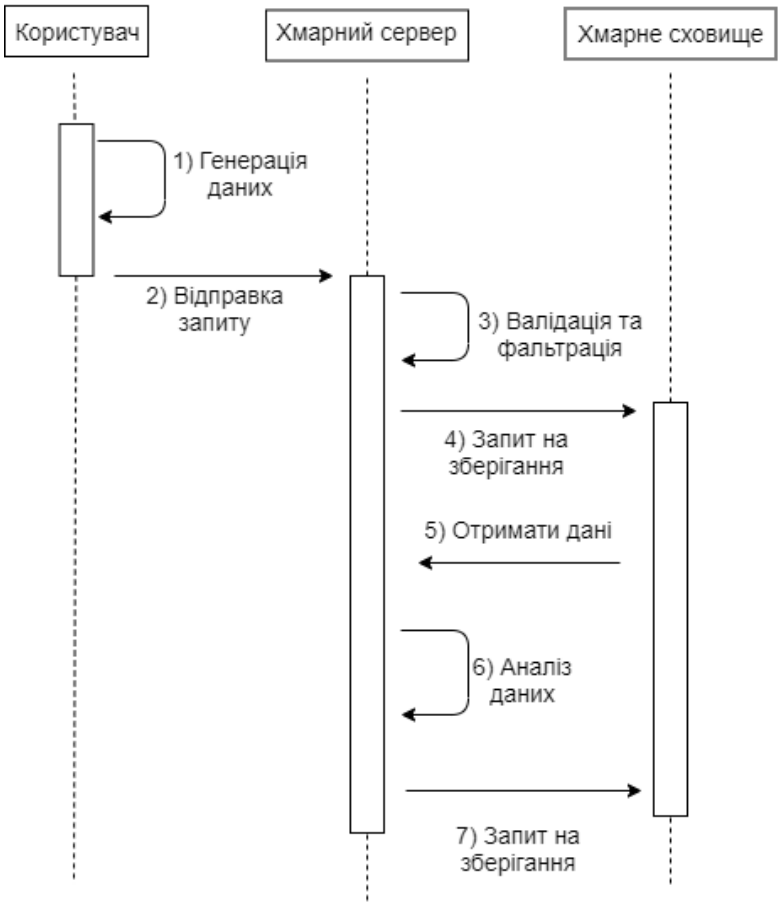

Рисунок 3.1. – Діаграма послідовності для першої моделі дослідження

У цьому сценарії задіяні два шари системи: граничний або клієнтський та хмарний. Ця модель допомагає зрозуміти, як будується класична хмарна архітектура. На граничному шарі знаходяться декілька комп’ютерів-клієнтів, які генерують дані з певною періодичністю та відправляють їх напряму до хмарного серверу. Уся бізнес-логіка працює саме на цьому хмарному сервері, що знаходиться на певному віддаленні від клієнтських пристроїв. Сервер

проводить валідацію клієнтських даних, фільтрує та залишає лише ті, які несуть у собі важливу інформацію. Після цих операцій, ці дані відправляються до бази даних у цій самій мережі хмари. Також цей сервер налаштований періодично робити вибірку блоків даних зі сховища та проводити їх аналіз.

Для першого тесту будуть використано запити на обробку десять тисяч об’єктів, передаваємих у вигляді JSON файлу розміром приблизно 40 мегабайт від декількох клієнтських машин з інтервалом в одну хвилину через HTTP запит. Середній час відправлення запиту на створення десяти тисяч об’єктів та отримання відповіді від сервера складає в середньому до 63 секунд. Основну частину цього проміжку часу займає саме валідація та фільтрація відправлених об’єктів, після якої до бази даних надходить лише п’ять тисяч об’єктів для зберігання.

Час відправлення клієнтом запиту на обчислення та аналіз хмарним сервером результатів зібраних даних та на отримання результатів такого запиту складає в середньому: до 17 секунд для обробки п’яти тисяч об’єктів, до 29 секунд для п’ятнадцяти тисяч та до 73 секунд для сорока тисяч об’єктів. Враховуючи, що клієнтами в даному дослідженні виступають лише два пристрої, а при збільшенні кількості даних на сервері прямо пропорційно збільшується навантаження процесору і час обробки цих даних, то, при обробці десяти тисяч таких самих об’єктів, сервер даного типу не буде справлятися з поставленими задачами і клієнти не будуть отримувати результатів від серверу через обмеження HTTP протоколу на час обробки одного запиту. Результати даного дослідження наведені в таблиці 3.1.

Таблиця 3.1 – Результати першого дослідження першої моделі

Номер тесту	Кількість об’єктів на сервері	Час створення об’єктів (секунди)	Час виконання аналізу (секунди)
1	5000	58	17
2	10000	60	23
3	15000	61	29
4	40000	63	73

Для другого дослідження кількість клієнтів збільшена до чотирьох машин, які також передають тисячу JSON об’єктів з інтервалом в 5 секунд, що дозволяє дослідити поведінку системи у високонавантаженому сценарії. Кількість передаваємих об’єктів була зменшена зважаючи на результати першого дослідження та через збільшення клієнтських машин у два рази. В результаті зменшення об’єму передаваємої інформації від клієнтів в десять разів, час обробки одного запиту складає в середньому від 7 до 13 секунд. Оскільки кількість одночасно підключених клієнтів складає чотири машини, через деякий час навантаження на хмарний сервер починає перебільшувати оптимальне обмеження і швидкість обробки навіть такої кількості об’єктів падає майже в три рази.

Під час відправлення запитів клієнтами на обчислення та отримання результатів від серверу, час на отримання результатів від серверу збільшується прямо пропорційно приблизно на 5 секунд через кожний інтервал тестування. Коли час на отримання відповіді від серверу для першого клієнта складає 31 секунду, то третій і четвертий клієнт не отримують відповіді від сервера взагалі через перевищення часу очікування від сервера. Під час обчислення сервером даних із зібраних об’єктів центральний процессор EC2-t2.micro сервера навантажений майже на 80 відсотків. Результати даного дослідження приведені в таблиці 3.2.

Таблиця 3.2 – Результати другого дослідження першої моделі

Номер тесту	Кількість об’єктів на сервері	Час створення об’єктів (секунди)	Час виконання аналізу (секунди)
1	1000	7	6
2	5000	9	16
3	10000	13	31
4	20000	16	53

Дослідження даної моделі показують (рис. 3.2), що при збільшенні кількості одночасно підключених клієнтів та при зменшенні інтервалу між запитами від цих клієнтів, час на обробку запитів від серверу зростає з великою швидкістю і, при перевищенні допустимого навантаження на сервер, деякі з клієнтів не отримують прийнятної відповіді від хмарного сервера.

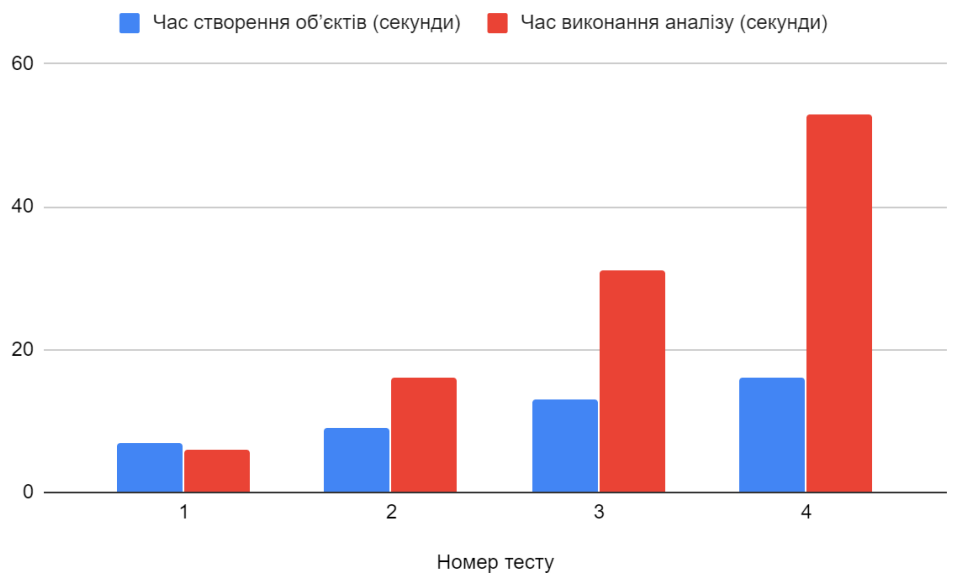

Рисунок 3.2. – Результати тестування навантаженням першої моделі

Другий підхід описує використання локального туманного вузла замість хмарного серверу (див. рис. 3.3). Ця модель системи допомагає використати одну із основних переваг туманного обчислення, а саме наближеність

обчислювальних ресурсів до граничних пристроїв. На відміну від першої моделі, затримка в передачі даних до серверу значно знижується, оскільки у даному випадку сервер розташований в одній локальній мережі з граничними пристроями. Цей сервер виконує роль туманного вузла і відповідає за валідацію, фільтрацію, обчислення та аналіз даних, що поступають через запити. Після цих операцій сервер робить запит до хмарної бази даних (що знаходиться на віддаленій географічній відстані від цього вузла) на зберігання даних.

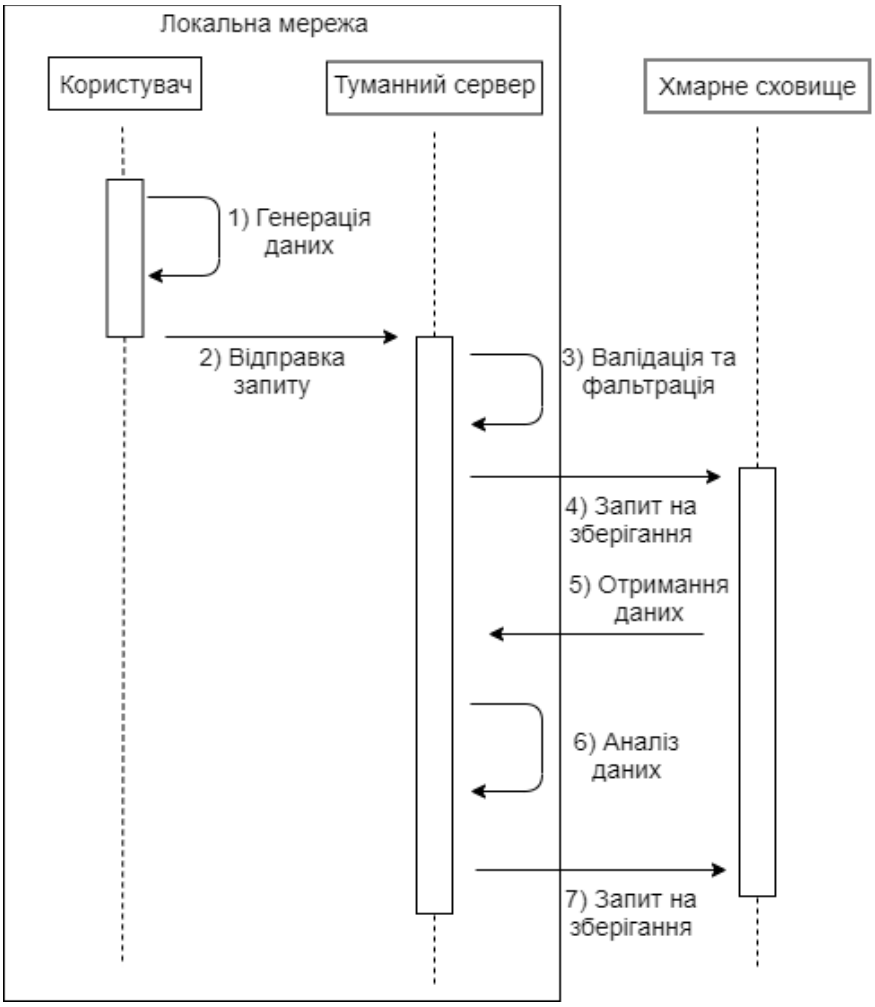

Рисунок 3.3. – Діаграма послідовності для другої моделі дослідження

Дослідження даного методу використовує таку ж саму кількість клієнтів у двох тестах та ідентичний об’єм даних, як при дослідженні першої моделі. У

першому тесті, час від відправки запиту та після валідації та фільтрування десяти тисяч об’єктів сервером зменшений майже в три рази саме через близьке розташування від клієнта до туманного серверу. Але час на відправку сервером оброблених даних такого об’єму до віддаленого сховища даних збільшується майже в сім разів. Тобто час на обробку однієї транзакції в такій моделі архітектури значно перевищує допустимий. На виконання сервером операцій обчислення витрачається в середньому до п’яти секунд для п’яти тисяч об’єктів таким чином, і ця продуктивність не знижується при зменшенні інтервалів між запитами і залишається стабільною. Результати дослідження приведені в таблиці 3.3.

Таблиця 3.3 – Результати першого дослідження другої моделі

Номер тесту	Кількість об’єктів на сервері	Час створення об’єктів (секунди)	Час виконання аналізу (секунди)
1	5000	397	5
2	10000	406	8
3	15000	403	9
4	40000	409	16

Після навантаженого тестування даної моделі, час обробки операцій зберігання даних у сховищі перевищує допустимий час для HTTP протоколу і вже при перших запитах три з чотирьох клієнтів не можуть отримати відповідь від сервера. Під час постійних запитів на обчислення з інтервалами в 5 секунд одночасно від чотирьох клієнтів, час на обробку та відправлення результату для одного клієнта складає від 5 до 7 секунд. Для виконання операцій аналізу даних з хмарного сховища туманному серверу необхідно витратити до 10 секунд, але після цього необхідно відправити оновлені дані назад до сховища, що займає стільки ж часу, як і при обробці запиту на створення цих даних. Результати даного дослідження приведені в таблиці 3.4.

Таблиця 3.4 – Результати другого дослідження другої моделі

Номер тесту	Кількість об’єктів на сервері	Час створення об’єктів (секунди)	Час виконання аналізу (секунди)
1	1000	126	3
2	5000	-	7
3	10000	-	11
4	20000	-	14

За результатами цих тестів (рис. 3.4) можна сказати, що дана модель архітектури може виконувати задачі щодо фільтрування та обчислення даних на сервері за досить невеликий час у порівнянні з віддаленим хмарним сервером за рахунок наближеності до місця генерації даних, але абсолютно не призначена для операцій зберігання або періодичного аналізу даних у сховищі через значну відстань між сервером та хмарною базою даних.

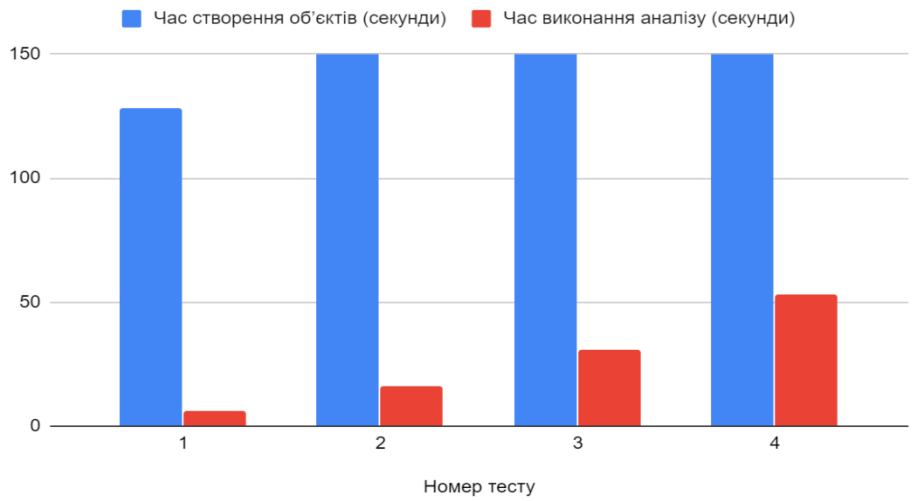

Рисунок 3.4. – Результати тестування навантаженням другої моделі

Третій підхід додає до туманного вузла додаткову базу даних для тимчасового зберігання інформації (див. рис.3.5). Це допомагає розвантажувати

основну хмарну базу тим, що не потрібно постійно відправляти всі дані від клієнта до хмари. Таким чином, дані спершу зберігаються на сервері, що знаходиться в тій самій локальній мережі, значно знижуючи затримку та навантаження на трафік під час передачі на відміну від другої моделі.

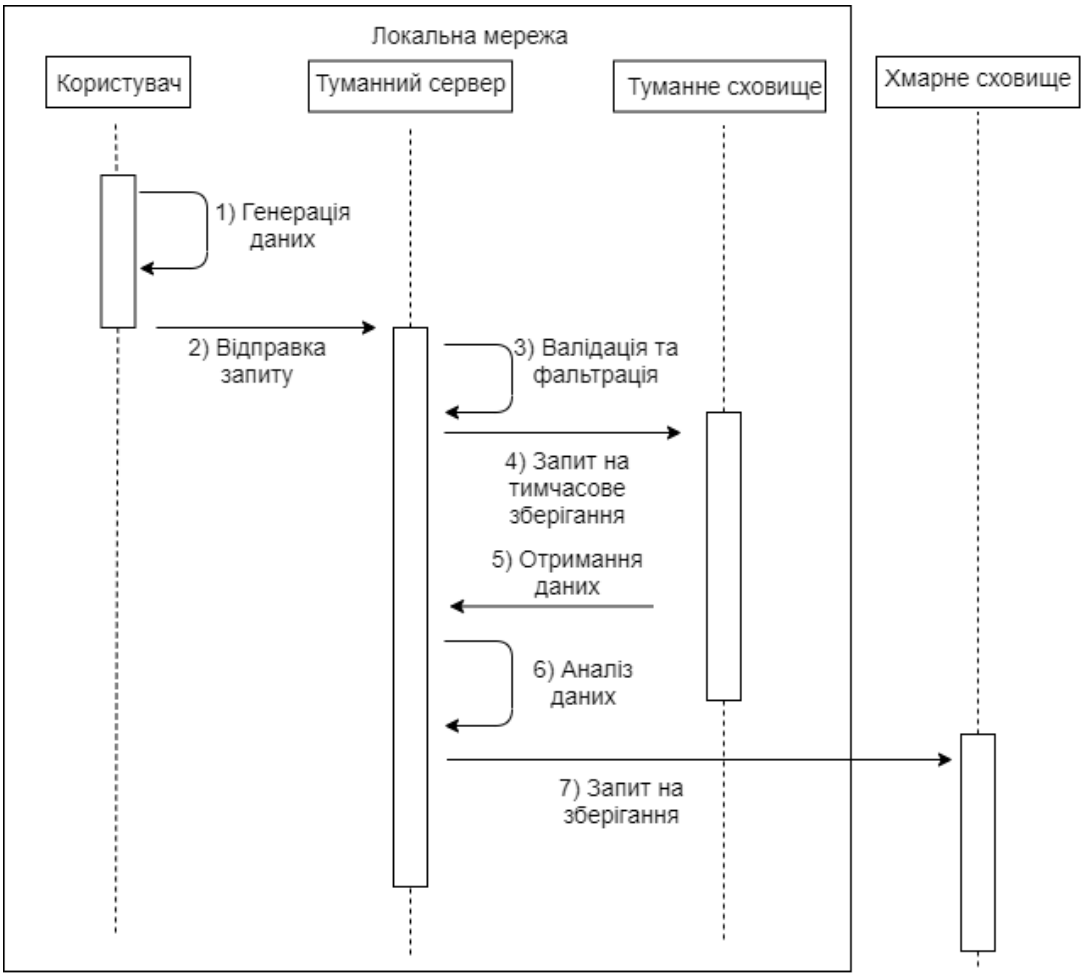

Рисунок 3.5. – Діаграма послідовності для третьої моделі дослідження

Під час тестування з двома клієнтами та завантаженням даних, дана модель показує значно кращий результат у часі обробки запиту на створення десяти тисяч об’єктів у порівнянні з попередніми моделями. Середній час варіюється від 28 до 33 секунд, що вдвічі швидше, ніж при використанні хмарного серверу при однаковому об’ємі відправлених даних. Час на вилучення даних з бази та на їх обчислення та аналіз складає в середньому: від 5 до 7 секунд для п’яти тисяч

об’єктів, від 13 до 17 секунд для десяти тисяч та від 35 до 44 секунд для сорока тисяч об’єктів. Такий результат досягається завдяки тому, що після закінчення обчислення певного об’єму даних, затримка при відправленні результату від сервера до клієнта значно менша, ніж при використанні хмарного сервера. Це допомагає вдвічі покращити час роботи при використанні туманних серверів з меншою обчислювальною потужністю, ніж у хмарного сервера. Результати цього дослідження приведені в таблиці 3.5.

Таблиця 3.5 – Результати першого дослідження третьої моделі

Номер тесту	Кількість об’єктів на сервері	Час створення об’єктів (секунди)	Час виконання аналізу (секунди)
1	5000	28	7
2	10000	30	17
3	15000	29	24
4	40000	36	44

Після проведення навантаженого тестування з чотирма клієнтами та об’ємом JSON файлу в тисячу об’єктів, час для кожного з клієнтів складає від 4 до 7 секунд в залежності від навантаження процессора туманного серверу. Час залишається майже незмінним, незважаючи на збільшення кількості клієнтів до шести. На відміну від другої моделі, дані після первинної обробки та фільтрації відправляються спочатку в локальну базу даних, що відбувається значно швидше ніж при роботі з хмарним сховищем, і це дозволяє туманному серверу підтримувати високу ефективність при роботі зі зростаючою кількістю клієнтів. Результати дослідження приведені в таблиці 3.6.

Таблиця 3.6 – Результати другого дослідження третьої моделі

Номер тесту	Кількість об’єктів на сервері	Час створення об’єктів (секунди)	Час виконання аналізу (секунди)
1	1000	5	5
2	5000	4	15
3	10000	4	20
4	20000	7	41

При роботі напряму з туманним вузлом, клієнтські машини не залежать від хмарного сховища та від великого часу передачі запитів від та до нього, а повністю обслуговуються туманним сховищем, що зберігає важливу інформацію та здатне за короткий час надати її за запитом клієнта. Графік результатів навантаженого тестування третьої моделі показано на рисунку 3.6.

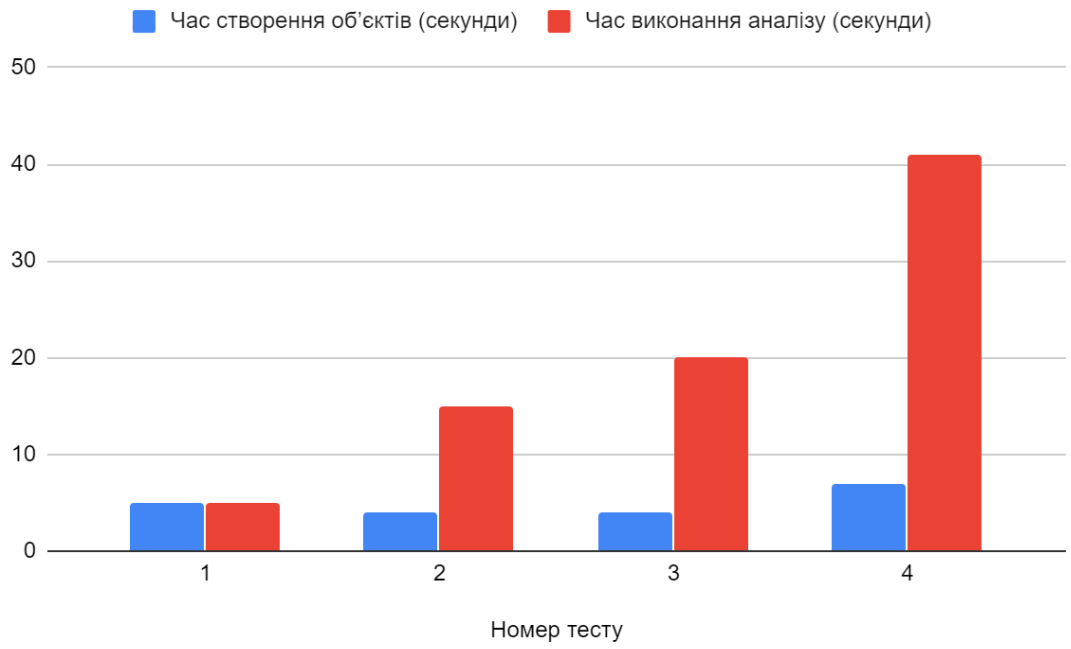

Рисунок 3.6. – Результати тестування навантаженням третьої моделі

Мінусом цієї моделі є те, що для виконання аналізу даних необхідно поступово збільшувати кількість паралельно працюючих серверів, тому що вже

при роботі з чотирма клієнтами при обчисленні сорока тисяч об’єктів, навантаженість процессора досягає ста відсотків за досить короткий час.

Четверта модель представляє собою повну архітектуру хмарно-туманних обчислень, додаючи до третього варіанту хмарний сервер для прямого доступу до хмарного сховища та аналізу великого об’єму даних (рис. 3.7). Такий підхід є складнішим для побудови та дизайну архітектури системи, але дозволяє використовувати одночасно всі переваги туманного обчислення та централізованого хмарного сервера.

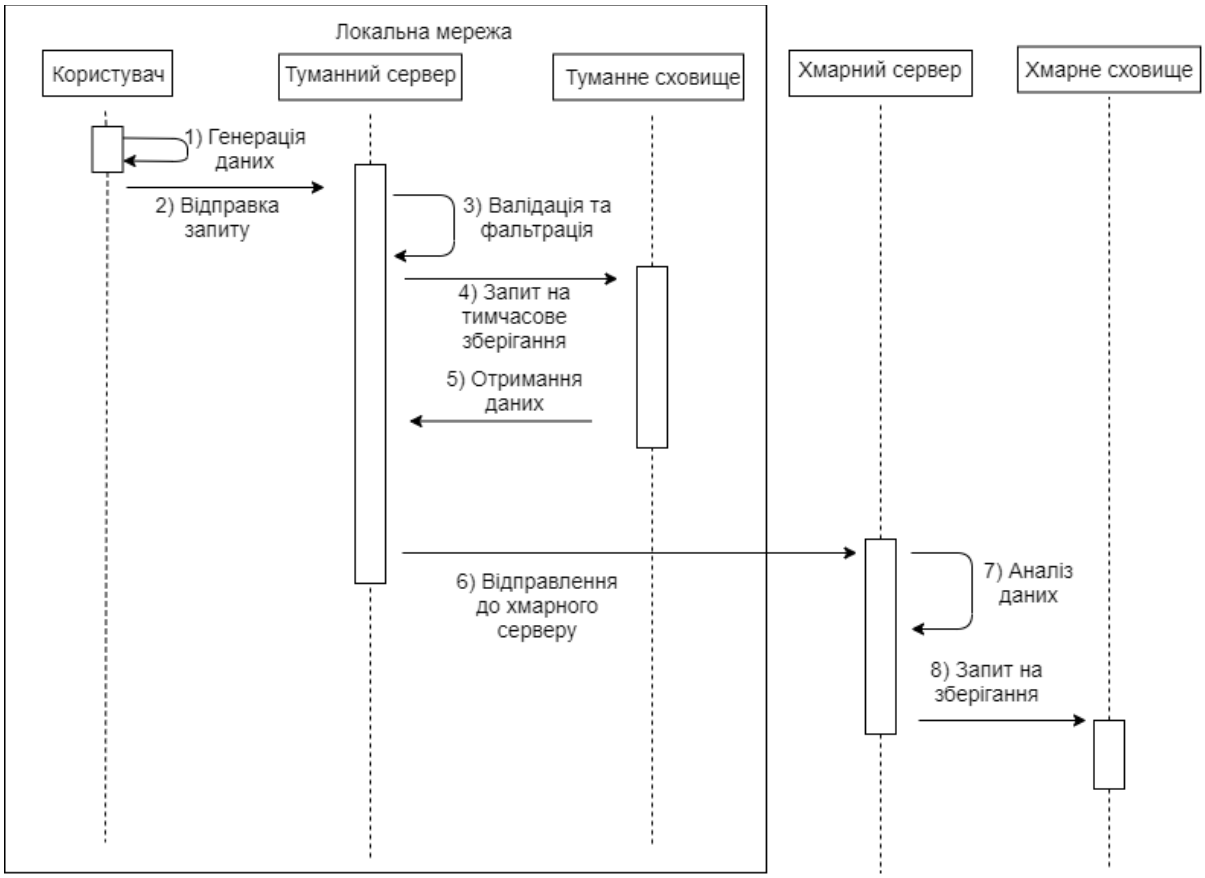

Рисунок 3.7. – Діаграма послідовності для четвертої моделі дослідження

В результаті першого тесту, витрачений час на запит створення десяти тисяч об’єктів не відрізняється від третьої моделі, тому що схема локального туманного вузла залишається такою ж, і середнє його значення становить до 31 секунд. Оскільки попередня валідація та фільтрація відбуваються на цьому ж

сервері і об’єм передаваємих даних після фільтрації зменшений приблизно вдвічі, то загальний час виконання операції, починаючи від відправлення запиту клієнтом та закінчуючи збереженням даних у хмарній базі даних, зменшується. Час передачі десяти тисяч об’єктів від туманного серверу до хмари складає в середньому до 60 секунд, як і при тестуванні першої моделі.

Ця модель надає змогу розділити обов’язки системи на туманний та хмарний сервери. Всі операції, у яких задіяні кінцеві користувачі, оперуються саме туманним сервером для зниження затримки під час обробки клієнтських запитів, а аналіз даних (який можна проводити через деякі інтервали та перезаписувати дані в хмарну базу даних) бере на себе хмарний сервер.

Для аналізу десяти тисяч об’єктів, що були передані туманним вузлом, хмарному серверу необхідно витратити від 29 до 33 секунд. Після цього оновлені дані відправляються хмарним сервером до хмарного сховища, на що додатково витрачається від 3 до 5 секунд. Якщо використовувати туманний сервер для періодичного аналізу даних з хмарного сховища, то цей час буде збільшено у порівнянні з обробкою хмарним сервером. В такому випадку час буде складати від 45 до 53 секунд, враховуючи час на передачу назад до хмарного сервера та на повторне збереження в базу даних. Результати дослідження приведені в таблиці 3.7.

Таблиця 3.7 – Результати першого дослідження четвертої моделі

Номер тесту	Кількість об’єктів на сервері	Час створення об’єктів (секунди)	Час виконання аналізу (секунди)
1	5000	29	21
2	10000	28	33
3	15000	31	42
4	40000	30	66

При проведенні навантаженого тестування зі збільшеною кількістю користувачів, час на обробку тисячі об’єктів від одного користувача складає від 4 до 7 секунд, як і в третій моделі. Це тестування виявило, що додатковою перевагою розподілення задач між хмарним та туманним шарами є зменшення загального навантаження на процесор туманного серверу з майже ста відсотків до шестидесяти. Результати дослідження приведені в таблиці 3.8.

Таблиця 3.8 – Результати другого дослідження четвертої моделі

Номер тесту	Кількість об’єктів на сервері	Час створення об’єктів (секунди)	Час виконання аналізу (секунди)
1	1000	4	8
2	5000	6	22
3	10000	5	36
4	20000	7	60

Дана модель показує, що використання хмарного серверу для періодичного аналізу та оновлення даних є більш ефективним рішенням. (див. рис. 3.8).

Рисунок 3.8. – Результати тестування навантаженням четвертої моделі

Цей підхід дозволяє знизити навантаження системи шляхом розподілення операцій між хмарним та туманним серверами, комбінуючи два види обчислення.

Завдяки такому розподіленню, хмарний сервер не втрачає швидкості обробки інформації та виконання логічних операцій при збільшенні об’єму відправляємих даних.

Слід зазначити, що в даній моделі також зменшується вірогідність перенавантаження мережі через вхідний трафік, оскільки дані з туманного серверу передаються частинами через певні часові інтервали.

3.2. Порівняння отриманих результатів

За результатами тестування та зібраними метриками можна порівняти ефективність роботи кожної моделі при доступних сценаріях. На рисунку 3.9 представлена діаграма зібраних метрик під час навантаженого тестування кожної моделі.

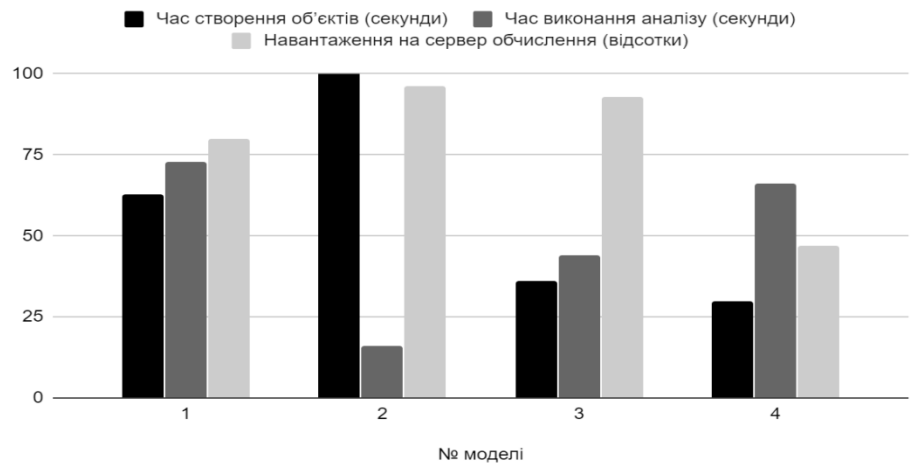

Рисунок 3.9. – Результати навантаженого тестування кожної моделі

Перша модель, що представляє собою типове середовище для хмарного обчислення, показує високу обчислювальну потужність при виконанні операцій

над даними на сервері та має досить високу місткість хмарного сховища даних (20 тисяч мегабайт для t2.micro).

Основними недоліками даного підходу є: дуже висока затримка під час обробки клієнтських запитів, високий час очікування відповіді від сервера клієнтом, неможливість одночасно обробляти декілька запитів великого об’єму та нестабільність роботи серверу під час перенавантаження запитами.

Друга модель системи використовує туманний сервер, що знаходиться безпосередньо біля клієнтських пристроїв, замість хмарного сервера, завдяки чому зменшується затримка між моментом відправлення запиту клієнтом і початком обробки сервером цього запиту. Це прискорює час виконання операцій, що пов’язані з обробкою даних на сервері, незважаючи на те, що туманний сервер поступається в обчислювальній потужності перед хмарним.

З іншого боку, всі операції, що пов’язані з додаванням нових даних до бази даних або з видаленням даних із неї, потребують значно більше часу в порівнянні з архітектурою хмарного обчислення через значну відстань між місцем обчислення та сховищем. Цей недолік робить даний тип архітектури дуже обмеженим у використанні при порівнянні з іншими типами, так как при роботі з системою, що надає можливість зберігати інформацію, половина операцій потребують невиправдано велику кількість часу на виконання і знижують загальну продуктивність системи.

Третя модель в певному сенсі покращує другу, шляхом розширення туманного вузла додатковим сховищем даних. Це значно знижує загальний час виконання транзакцій під час додавання або видалення інформації зі сховища, а саме вдвічі при порівнянні з хмарним сервером.

Однак, слід зазначити, що туманне сховище зазвичай має значно менший об’єм ніж хмарне, тому дані періодично необхідно передавати від туманного вузла до хмарного шару, на що відходить така ж кількість часу, як і при тестуванні другої моделі архітектури.

Туманний сервер на момент відправки даних або повинен мати невелику кількість підключених клієнтів, або необхідно додавати допоміжний сервер для того, щоб не знижувати продуктивність роботи обчислень у вузлі. Також, додатковий туманний сервер слід додавати при постійному зростанні навантаження на основний сервер та його процесор для підтримки розподіленої роботи вузла. Декілька паралельно працюючих серверів порівняно невеликої потужності здатні значно прискорити виконання клієнтських запитів та швидше надавати відповідь.

Четверта модель являється самою комплексною для побудови та налаштування інфраструктури на відміну від інших. Але саме така модель дозволяє виправити недоліки перших трьох, завдяки використанню туманного та хмарного обчислення одночасно.

У цій моделі швидкість обробки запитів від клієнтів займає приблизно вдвічі часу, ніж при роботі з хмарним обчисленням. Туманний сервер виконує фільтрацію та валідацію об’єктів, що зменшує об’єм відправляємих до хмари даних, залишаючи лише необхідну інформацію та зменшуючи навантаження на мережу. Сховище даних у хмарному шарі бере на себе частину навантаження хмарної бази даних під час обробки транзакцій, а також прискорює загальну швидкість роботи системи завдяки тому, що знаходиться на близькій відстані від граничних пристроїв.

Дана архітектура майже не втрачає швидкості обробки запитів при збільшенні кількості одночасно підключених клієнтів завдяки розподіленню навантаження між декількома серверами в туманному вузлі, а також завдяки логічному розподіленню операцій між хмарним та туманним шаром. Таким чином, хмарний сервер задіяний лише при роботі безпосередньо з хмарним сховищем, періодично проводячи необхідні аналітичні операції над попередньо зібраними даними.

У цей час, туманний сервер відповідає за безпосередню роботу з клієнтськими пристроями, а саме за обробку запитів, фільтрацію та валідацію

даних та за відправлення та отримання даних від хмарного шару. Основним недоліком даної архітектури є досить складне налаштування взаємодії між двома шарами обробки даних, а також постійний контроль всіх працюючих розподілених серверів.

На підставі проведеного аналізу моделей архітектури, можна зробити висновок, що поєднання архітектури туманного та хмарного обчислення дозволяє знизити навантаження на основний хмарний сервер та його базу даних, прискорити швидкість обробки клієнтських запитів та швидкість обчислювальних операцій за допомогою туманного шару, а також розподілити всю кількість задач системи на декілька шарів та серверів.

4 ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ У НАУКОВУ ТА ПРАКТИЧНУ ДІЯЛЬНІСТЬ

4.1. Опис можливості використання отриманих результатів у науковій і практичній діяльності.

Після проведення дослідження чотирьох моделей архітектури, можна зробити висновок щодо області їх застосування та ефективності виконання задач.

Основними двома критеріями, які відрізняють хмарні обчислення від туманних, є затримка передачі та пропускна можливість мережі. Хмарний сервер здатний обробляти одночасно велику кількість запитів з невеликим об’ємом передаваємих даних. Але при підвищенні об’єму інформації, яку ці запити передають, та при збільшенні частоти інтервалів запитів, такий сервер починає набагато більше залежити від пропускної здатності мережі, навантаження на яку в декілька разів більше, ніж під час роботи такого ж сценарію, але з використанням туманного шару. Звичайне хмарне обчислення не є достатньо ефективним рішенням при роботі з постійно оновлюємими даними в реальному часі. Такий вид архітектури більше підходить для зберігання великої кількості інформації в об’ємному хмарному сховищі, відправки відповідей щодо цієї інформації до серверів нижче по ієрархії та для аналізу даних, зібраний за певний період часу. Найкращими областями застосування хмарного обчислення є: робота з big data, соціальні мережі, GPS-навігація, інфраструктура для тестування та розробки.

Для роботи з частими запитами та для підвищення швидкості відповіді від сервера можна використовувати туманний сервер як посередник між граничними пристроями та хмарним сховищем. Цей підхід схожий на архітектуру граничного обчислення, але з використанням сховища даних. Найбільш підходящою областю застосування для такого підходу є системи, що призначені для збору і аналізу даних у реальному часі, які не залежать від частого використання сховища даних для зберігання. В такого роду системі основна частина роботи припадає саме на

серверну частину, яка повинна включати декілька паралельно працюючих машин для розподілення навантаження на кожну з них. Така група серверів, що знаходяться на близькій відстані до джерела виникнення даних, здатна дуже швидко проводити їх обробку та обчислення. Хмарне сховище повинне використовуватися лише в рідких випадках, для зберігання лише коректної і важливої інформації, оскільки час на відправку даних від туманних серверів до віддаленого хмарного шару є дуже великим і цей процес значно навантажує мережу. Низька затримка між джерелом надходження даних та місцем їх обробки робить цю модель архітектури значно ефективнішою ніж хмарне обчислення в системі такого призначення. Прикладами систем, для яких найкраще всього підійде саме така модель архітектури, є: програмне забезпечення автономних автомобілів, системи автоматизації виробництва, системи віртуальної реальності.

Якщо крім обчислення в реальному часі система потребує частого збереження певних даних, то другу модель потрібно обов'язково розширити туманною базою даних. Вона повинна зберігати лише часто використовувану в туманному вузлі інформацію для швидшого її вилучення, а більш довгострокові дані необхідно передавати через інтервали до хмарного сховища, оскільки об'єм туманного сховища обмежений. Використання такої моделі не втрачає швидкості обчислення та обробки запитів при порівнянні з граничними обчисленнями, а туманний сервер разом із сховищем даних являє собою повноцінний туманний вузол. Вимоги до регулювання навантаження на цей вузол більш суворі через обмежену обчислювальну потужність сервера та низький об'єм даних для зберігання. Якщо не розподіляти навантаження між декількома серверами на вузлі, то при постійному збільшенні об'єму даних у мережі, цей вузол швидко вийде з ладу і не зможе обробляти запити. Також для реалізації задачі відправлення даних з туманного сховища до хмарного необхідно використовувати спеціально виділений сервер у вузлі через значне навантаження при виконанні цієї операції через певні інтервали часу. Така модель найкраще підходить для

таких областей: контроль дорожнього трафіку, технологія розумних будинків або офісних приміщень, системи прямої трансляції відео.

Туманне обчислення показує себе найбільш ефективно саме як доповнення до хмарної інфраструктури. Така архітектура здатна показувати низьку затримку в часі при обробці даних в реальному часі біля місця їх надходження, знижувати загальний об’єм використання ресурсів мережі та підвищувати час виконання операцій системи. В системі такого роду також необхідно регулювати кількість працюючих серверів у туманному вузлі в залежності від змін навантаження на обчислювальні машини та сховища даних. Кожен тип операцій повинен виконуватися спеціально відведеним для нього сервером, в залежності від обчислювальної потужності машини. При необхідному розподіленні задач, загальне навантаження на хмарні сервер та базу даних знижуються, що прискорює час їх роботи над комплексними операціями. Технологія хмарно-туманних обчислень може бути застосована в таких самих областях, як і третя модель, а також у системах контролю стану здоров’я людини, при роботі з мережею IoT пристроїв, у системах безпеки, для автоматизації процесів та збору даних на підприємствах.

4.2. Відомі проблеми туманних обчислень

При використанні поширеної розподіленої мережі туманних вузлів, існує проблема алгоритмізації рівномірного навантаження на сховища даних в усіх вузлах. Деякі з них будуть отримувати більший об’єм даних ніж інші, через це вони будуть значно швидше переповнюватися і потребувати переправлення усіх даних до хмарного сховища. Тимчасовим рішенням може бути збільшення об’єму сховища в тих вузлах, які є найбільш затребуваними в певній географічній

області, але таке рішення є скоріше реакцією на виникнення проблеми в системі, ніж способом її запобігання.

Іншою проблемою є підтримка приватності та захищеності даних під час обробки в туманному вузлі. Основна взаємодія між рівнями архітектури відбувається через мережу, що робить можливим втрату або зміну важливих даних під час їх передачі.

Підтримка різних груп клієнтських приладів на рівні програмного забезпечення, а також впровадження мережевих протоколів для їх підключення – все це є складним процесом та проблемою, яку необхідно вирішити під час налаштування інфраструктури туманного обчислення [20]. Необхідно постійно слідкувати за виявленими помилками підключення або обробки запитів окремо у рамках кожного серверу, що вимагає додаткового налаштування системи логування процесів у кожному туманному вузлі.

Ще однією існуючою проблемою є дуже високе споживання електроенергії при одночасній роботі великої кількості розподілених серверів системи. Оскільки, кожний вузол для стабільної роботи потребує додаткових обчислювальних машин, витрати енергії на підтримку такої інфраструктури будуть дуже високими.

4.3. Питання подальшого дослідження

Для відповіді на існуючі актуальні питання та вирішення відомих проблем туманного обчислення необхідно проводити додаткові дослідження систем, що базуються на цій архітектурі.

Одним із варіантів подальшого дослідження цього типу побудови систем є питання географічного розміщення туманних вузлів таким чином, щоб задовольняти потребам користувачів системи. Дизайн такої інфраструктури повинен враховувати мобільність граничних пристроїв, що створює

непередбачувані ситуації зростання навантаження у випадкових туманних вузлах під час довгого періоду роботи системи. Правильне розташування туманних вузлів дозволить знизити вірогідність таких ситуацій, а також спростити процес розміщення та підтримки серверів обчислення системи.

Іншим важливим питанням стосовно туманного обчислення, яке необхідно вирішити у подальших дослідженнях, є підтримка високого рівня безпеки в кожному з туманних вузлів системи. Окрім існуючих проблем з безпекою інформації в хмарному обчисленні, розподілення та поширеність серверів туманних вузлів підвищує складність підтримки конфіденційності та цілісності даних, що передаються. Необхідно дослідити, які існуючі варіанти захисту даних, що надходять через мережу від клієнтських приладів, є найбільш прийнятними під час розробки системи.

Ці питання підлягають більш детальному вивченню та можуть бути вирішені в подальших дослідженнях.

ВИСНОВКИ

В роботі було проведено дослідження методів управління доступом до сховищ даних систем, які побудовані з застосуванням архітектури туманного обчислення та було здійснено аналіз переваг та недоліків використання даної архітектури у порівнянні з відомими архітектурами хмарних обчислень. Для проведення аналізу було обрано три моделі побудови систем з використанням серверів туманних обчислень, які були порівнянні з моделлю хмарного сервісу. На підставі результатів обчислення метрик роботи кожної з цих моделей було зроблено висновки щодо переваг та недоліків кожної з цих моделей, та щодо сфери їх застосування.

Під час проведення порівняльного аналізу методів управління доступом до сховищ даних було виявлено, що хмарні обчислення втрачають ефективність роботи пропорційно зі збільшенням кількості даних та користувачів. У зв'язку з цим виникають проблеми, що пов'язані з часовими характеристиками передачі даних та перенавантаженням мережі, що призводить до порушення стабільності роботи систем.

Поєднання технологій туманного та хмарного обчислення дозволяє зменшувати кількість проблем, пов'язаних з затримками запитів та перенавантаженням мережі шляхом впровадження ефективних рішень щодо розподілення обчислень на значній за розміром географічній зоні між великою кількістю туманних вузлів, які виконують роль серверів та тимчасовими сховищами даних. Таким чином, затримка у часі під час передачі запитів зменшується, одночасно з чим знижується навантаження на хмарний сервер та на його мережу. Використання технології туманних обчислень дозволяє значно підвищити ефективність використання хмарних серверів та покращити рівень роботи системи з великою кількістю користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Shiva Jegan R.D, Vasudevan S.K, Abarna K., Prakash P, Srivathsan S., Gangothri V., "Cloud computing: A Technical Gawk". 2014. URL: https://www.amrita.edu/publication/cloud-computing-technical-gawk.

2. Pedro Caldeira Neves, Jorge Bernardino, "Big Data in the cloud: A survey". 2015. URL: https://www.ronpub.com/OJBD_2015v1i2n02_Neves.pdf .

3. Cisco, “Cisco delivers vision of fog computing to accelerate value from billions of connected devices,” Cisco, Tech. Rep., Jan. 2014. URL: https://newsroom.cisco.com/press-release-content?type=webcontent&articleId=1334100.

4. F. Bonomi, R. Milito, J. Zhu, “Fog computing and its role in the internet of things”.2012.URL: https://conferences.sigcomm.org/sigcomm/2012/paper/mcc/p13.pdf.

5. Shanhe Yi, Zijiang Hao, Zhengrui Qin, "Fog Computing: Platform and Applications". 2015. URL: https://ieeexplore.ieee.org/document/7372286.

6. E. M. Tordera, X.Masip-Bruin, “What is a fog node a tutorial on current concepts towards a common definition,”. 2016. URL: https://arxiv.org/ftp/arxiv/papers/1611/1611.09193.pdf.

7. Anuj Kumar, K.P Saharan, "Fog in comparison to cloud: A survey". 2015. URL:https://www.researchgate.net/publication/281953376_Fog_in_Comparison_to_Cloud_A_Survey.

8. I. Stojmenovic and S. Wen, “The fog computing paradigm: Scenarios and security issues”. 2014. URL: https://annals-csis.org/proceedings/2014/pliks/503.pdf.

9. Souza, V.B.C.; Ramírez, W.; Masip-Bruin, X.; Marín-Tordera, E.; Ren, G.; Tashakor, G. “Handling service allocation in combined Fog-cloud scenarios”. 2016. URL: https://ieeexplore.ieee.org/document/7511465.

10. Sourav Kunal, Arijit Saha, Ruhul Amin, “An overview of cloud-fog computing: Architectures, applications with security challenges”. 2019. URL: https://www.researchgate.net/publication/333162093.

11. Cao, N., Wang, C., Li, M., Ren, K., Lou, W., “Privacy-preserving multi-keyword ranked search over encrypted cloud data.”. 2014. URL: https://pdfs.semanticscholar.org/cbe3/e7cf2c1f64ce667cacad6666e8f62fa2ff3f.pdf.

12. Viraj G. Mandlekar, Viresh Kumar Mahale, “Survey on Fog Computing Mitigating Data Theft Attacks in Cloud”. 2014. URL: https://www.ijircst.org/DOC/3_IRP2144caa8e0e-a940-4c1d-81c1-5286717307d3.pdf.

13. Cash, D., et al. “Dynamic searchable encryption in very-large databases: Data structures and implementation”. 2014. URL: https://eprint.iacr.org/2014/853.pdf.

14. Dsouza, C., Ahn, G.J., Taguinod, M. “Policy-driven security management for fog computing: Preliminary framework and a case study”. 2014. URL: https://sefcom.asu.edu/publications/Policy-driven-security-management-iri2014.pdf.

15. Z. Song, Y. Duan, S. Wan et al., “Processing Optimization of Typed Resources with Synchronized Storage and Computation Adaptation in Fog Computing”. 2018. URL: https://www.hindawi.com/journals/wcmc/2018/3794175/.

16. Pedro GL et al. “Edge-centric computing: vision and challenges”. 2015. URL: http://disi.unitn.it/~montreso/pubs/papers/ccr15.pdf.

17. Vilalta, R.; Lopez, L.; Giorgetti, A.; Peng, S.; Orsini, V.; Velasco, L.; “TelcoFog: A Unified Flexible Fog and Cloud Computing Architecture for 5G Networks”. 2017. URL: https://ieeexplore.ieee.org/document/8004151.

18. E. Baccarelli, P. G. V. Naranjo, M. Scarpiniti, M. Shojafar “Fog of Everything: Energy-Efficient Networked Computing Architectures, Research Challenges, and a Case Study”. 2017. URL: https://www.researchgate.net/publication/316787559.

19. Z. Hao, E. Novak, S. Yi, and Q. Li, “Challenges and Software Architecture for Fog Computing”. 2017. URL: http://www.cs.wm.edu/~liqun/paper/IC17.pdf.

20. S. Yi, C. Li, and Q. Li, “A survey of fog computing: Concepts, applications and issues”. 2015. URL: https://arxiv.org/pdf/1804.04365.