Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет «Комп'ютерні технології і системи»
Кафедра «Комп'ютерні інформаційні технології»

Пояснювальна записка
до кваліфікаційної роботи
ОС Магістр

на тему: Дослідження методів тестування продуктивності та масштабованості в хмарних системах

за освітньою програмою: 12 Інженерія програмного забезпечення
зі спеціальності: 121 Інженерія програмного забезпечення

Виконала: студентка групи ПЗ 2422: ____ /Марія СОЧЕНКО/

Керівник: ____ /Тетяна ГРИШЕЧКІНА/

Нормоконтролер: ____ /Світлана ВОЛКОВА/

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент ____

Дніпро – 2026 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Faculty «Computer technologies and systems»
Department «Computer information technology»

Explanatory note
to the qualification work
OS Master

on the topic: Research on Methods for Testing Performance and Scalability in Cloud Systems

according to educational curriculum 12 Software engineering
in the Speciality: 121 Software engineering

Done by the student of the group PZ 2422: ____ /Mariia SOCHENKO/

Scientific Supervisor: ____ /Tetyana HRYSHECHKINA/

Normative controller: ____ /Svitlana VOLKOVA/

Dnipro – 2026

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: Комп'ютерні технології і системи
Кафедра: Комп'ютерні інформаційні технології
Рівень вищої освіти: магістр
Освітня програма: 12 Інженерія програмного забезпечення
Спеціальність: 121 Інженерія програмного забезпечення
(шифр та назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
(підпис) (Ім'я ПРІЗВИЩЕ)
Дата _____

З А В Д А Н Н Я

на кваліфікаційну роботу для здобуття ступеня магістра
(ступінь вищої освіти)

студентці: Соченко Марії Олегівні
(Прізвище Ім'я По батькові)

1. Тема роботи: Дослідження методів тестування продуктивності та масштабованості в хмарних системах

Керівник роботи: Гришечкіна Тетяна Сергіївна, к.т.н., доц.
(Прізвище Ім'я По батькові, науковий ступінь, вчене звання)

затверджені наказом від «02» 10 2025 р. № 1401 ст

2. Строк подання студентом роботи: 11.01.2026 р.
3. Вихідні дані до роботи: результати тестування, основані на вхідних даних.
4. Зміст пояснювальної записки (перелік питань, які потрібно опрацювати):
4.1 Аналітична частина: дослідити методи тестування продуктивності та масштабованості хмарних систем
4.2 Наукове дослідження: обґрунтування обраних методів дослідження
4.3 Розробка: розробити алгоритми тестування продуктивності та масштабованості для обраних хмарних систем
4.4 Ефективність алгоритмів: дослідити ефективність алгоритмів тестування
5. Перелік графічного матеріалу (з точним зазначення обов'язкових креслень):
схеми проєктування, блок-схеми алгоритмів тестування, графічне надання результатів виконання алгоритмів тестування, слайди презентації.

КАЛЕНДАРНИЙ ПЛАН

№ пор.	Назва етапів кваліфікаційної роботи	Термін виконання розділів роботи	При-мітка
1	Описати вступну частину кваліфікаційної роботи	01.10.2025	
2	Дослідити хмарні технології та їх особливості	13.10.2025	
3	Проаналізувати сучасні тенденції у тестуванні хмарних систем	12.11.2025	30%
4	Проаналізувати продуктивність у хмарних середовищах	17.11.2025	
5	Виконати аналіз підходів тестування продуктивності в хмарі	24.11.2025	
6	Обґрунтувати вибір методів дослідження	03.12.2025	
7	Розробити експерименти та тестові сценарії	15.12.2025	60%
8	Виконати проєктування системи	18.12.2025	
9	Реалізувати алгоритми тестування та провести їх тестування та налагодження	22.12.2025	
10	Проаналізувати результати роботи та надати загальні висновки та рекомендації для подальшого дослідження	25.12.2025	
11	Оформлення пояснювальної записки та тез доповідей	05.01.2026	
12	Розробка демонстраційної презентації роботи	09.01.2026	100%
13	Подання кваліфікаційної роботи до кафедри	16.01.2026	
14	Захист кваліфікаційної роботи на засідання Екзаменаційної комісії	20.01.2026	

Дата видачі завдання «11» вересня 2025 р.

Керівник дипломної роботи _____ Тетяна ГРИШЕЧКІНА
(підпис) (ПІБ)

Студентка _____ Марія СОЧЕНКО
(підпис) (ПІБ)

РЕФЕРАТ

Об’єктом дослідження є хмарні системи, їх продуктивність і масштабованість.

Предметом дослідження є методи тестування продуктивності та масштабованості в хмарних системах, включаючи підходи до навантажувального тестування, тестування масштабованості, стрес-тестування та тестування відмовостійкості.

Метою роботи є вивчення нових підходів до тестування продуктивності та масштабованості в хмарних системах, особливостей симуляції навантаження та застосування результатів для оптимізації продуктивності.

Методами дослідження є аналіз існуючих досліджень, моделювання навантажень для оцінки продуктивності, проведення експериментів тестових сценаріїв, порівняння методів продуктивності та масштабованості.

Результати та їх новизна: визначені ефективні підходи до тестування продуктивності та масштабованості хмарних систем, розроблені алгоритми та тестові сценарії, які враховують сучасні тенденції та особливості хмарних технологій, виконаний порівняльний аналіз методів тестування та запропоновані шляхи покращення продуктивності хмарних систем.

Пояснювальна записка складається зі вступу, 4 розділів, висновків, бібліографічного списку та 7 додатків:

- у вступі описується мета роботи, завдання та наукова новизна. Складається із 2 сторінок;

- у першому розділі описується дослідження теоретичних аспектів тестування хмарних сховищ. Складається із 20 сторінок;

- у другому розділі описуються наукове дослідження за методами тестування та оцінювання за метриками. Складається із 10 сторінок;

- у третьому розділі описується дослідження інструментальних засобів тестування продуктивності та масштабованості хмарних системи. Складається із 36 сторінок;

- у четвертому розділі описується аналіз проведеного дослідження та виконаних тестувань. Складається із 8 сторінок;

- у висновках викладені загальні рекомендації та подальший розвиток дослідження. Складається із 2 сторінок;

- додатки містять технічне завдання, інструкцію для користувача та програміста, тексти скриптів та їх опис і специфікації та відомості про матеріали роботи.

Таблиць – 9, рисунків – 24, бібліографія – 70 джерел.

Ключові слова: еластичність, відмовостійкість, масштабованість, навантажувальне тестування, оптимізація продуктивності, стрес-тестування, тестування масштабованості, тестування продуктивності, хмарні системи.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАК, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – application programming interface

APM – Application Performance Monitoring

AWS – Amazon Web Services

CI/CD – continuous integration / continuous deployment

DevOps – development & operations

EC2 – Elastic Compute Cloud

FaaS – Function as a Service

GCP – Google Cloud Platform

HPA – Horizontal Pod Autoscaler

HTTP – Hypertext Transfer Protocol

IaaS – Infrastructure as a Service

KPI – ключовий показник продуктивності

NPS – Net Promoter Score

PaaS – Platform as a Service

RPS – Requests Per Second, запити за секунду

SaaS – Software as a Service

UX – user experience

БД – база даних

ІТ – інформаційні технології

МБ/с – мегабайти за секунду

ПЗ – програмне забезпечення

Хмара – хмарні технології, cloud computing

ШІ, AI – штучний інтелект, artificial intelligence

ВСТУП

Розвиток хмарних технологій став однією з ключових ознак сучасного інформаційного суспільства, кардинально змінюючи підходи до зберігання, обробки та передачі даних. Хмарні платформи пропонують високу гнучкість, легке масштабування та економічну доцільність, що перетворює їх на невід'ємний елемент інфраструктури різноманітних галузей, зокрема фінансів, освіти, медицини, промисловості та державного управління. Однак, попри численні переваги, інтеграція хмарних технологій ставить низку викликів щодо їхньої продуктивності, здатності до масштабування і адаптивності в умовах змінних експлуатаційних факторів [1-3].

Зростаюча популярність хмарних сервісів підкреслює важливість підвищення їхньої ефективності. Хмарні платформи повинні гарантувати стабільну продуктивність навіть за умов динамічних і непередбачуваних навантажень. У цьому аспекті важливою стає роль методів тестування продуктивності та масштабованості. Такі підходи дають змогу оцінити здатність системи до адаптації, виявити потенційні вузькі місця та визначити напрями для їх оптимізації [1-3].

Метою роботи є вивчення нових підходів до тестування продуктивності та масштабованості в хмарних системах, особливостей симуляції навантаження та застосування результатів для оптимізації продуктивності.

Для досягнення поставленої мети передбачається виконання наступного переліку завдань:

- вивчення сучасних тенденцій у розвитку хмарних технологій та їхнього тестування;

- аналіз основних підходів до тестування продуктивності, включаючи тестування навантаження, масштабованості, стрес-тестування та тестування відмовостійкості;

- розробка метрик оцінки, таких як пропускна здатність, час виконання операцій, відсоток помилок, використання ресурсів, еластичність і гнучкість горизонтального масштабування;

- створення сценаріїв тестування для виявлення обмежень хмарних систем і визначення способів їх усунення;

- проведення експериментів із застосуванням сучасних інструментальних засобів тестування;

- порівняльний аналіз результатів тестування для виявлення ефективних підходів до оптимізації продуктивності.

Наукова новизна дослідження полягає у створенні інноваційних підходів до тестування хмарних систем, які дозволяють здійснювати всебічну оцінку їх продуктивності та масштабованості. Отримані результати мають вагоме практичне значення, оскільки запропоновані методики можуть бути застосовані для оптимізації функціонування хмарних платформ. Це стає особливо важливим для компаній, що спеціалізуються на наданні хмарних послуг.

Дослідження робить вагомий внесок у розвиток галузі інженерії програмного забезпечення, сприяючи створенню більш надійних і продуктивних рішень для хмарних технологій. Очікується, що отримані результати сприятимуть підвищенню ефективності використання ресурсів хмарних платформ, зниженню витрат на їхню експлуатацію та забезпеченню високого рівня задоволеності кінцевих користувачів.

Робота зосереджена на вирішенні нагальної наукової та практичної проблеми, яка полягає у забезпеченні стабільності, високої продуктивності та хорошого рівня масштабованості хмарних систем, враховуючи динамічний розвиток сучасних технологій і постійне зростання потреб ринку.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ТЕСТУВАННЯ ХМАРНИХ ТЕХНОЛОГІЙ

1.1 Огляд хмарних технологій та їх особливостей

Хмарні технології (cloud computing, хмара) виступають однією з фундаментальних складових сучасного процесу цифрової трансформації. Завдяки цим технологіям компанії, підприємства та звичайні користувачі отримують унікальну можливість зберігати великі обсяги даних, запускати різноманітні програмні рішення та оперативно отримувати доступ до потужних обчислювальних ресурсів через Інтернет. Це значно знижує потребу в дорогих інвестиціях у локальну апаратну інфраструктуру, а також оптимізує витрати на її обслуговування. До того ж, хмарні сервіси забезпечують високу гнучкість, масштабованість і доступність, дозволяючи бізнесам швидше адаптуватися до мінливих умов ринку [4-8].

Основними характеристики хмарних технологій є доступність, масштабованість, еластичність тощо. Детальний опис характеристик наведений у таблиці 1.1

Таблиця 1.1 – Характеристики хмарних технологій

Характеристика	Опис характеристики
Доступність	Користувачі отримують доступ до ресурсів з будь-якої точки світу за наявності Інтернет-з’єднання
Масштабованість	Динамічно змінювати обсяги використовуваних ресурсів залежно від потреб
Еластичність	Автоматично додавати або зменшувати обчислювальні потужності відповідно до змін у робочих навантаженнях
Оплата за споживання	Оплата лише за використані ресурси
Централізоване управління	Усі обчислювальні ресурси керуються через єдину платформу

За наведеними характеристиками (табл. 1.1) можна виділити наступні моделі хмарних технологій [9-10]:

- Infrastructure as a Service (IaaS), надає базову інфраструктуру у вигляді серверів, сховищ, мережей;

- Platform as a Service (PaaS), забезпечує платформу для розробки, тестування та розгортання застосунків;

- Software as a Service (SaaS), пропонує готові програмні рішення через Інтернет;

- Function as a Service (FaaS), дозволяє запускати окремі функції без управління серверною інфраструктурою.

Хмарні технології підтримують різні моделі розгортання, що дає компаніям змогу обирати найбільш ефективний варіант відповідно до специфіки їхнього бізнесу. Використання моделі IaaS забезпечує організаціям швидке створення інфраструктури для впровадження нових проєктів, тоді як SaaS є чудовим вибором для компаній, які потребують готових програмних рішень із мінімальною необхідністю налаштувань. Модель PaaS стає незамінною для розробників, які прагнуть зосередитися на написанні коду замість витрачання часу на управління інфраструктурою [10].

Існує чотири типи хмар. Вони поділяються на публічні, приватні, гібридні та багатохмарні середовища. Гібридні хмарні рішення стали надзвичайно популярними у сучасному корпоративному середовищі, пропонуючи бізнесу нові перспективи через поєднання локальної інфраструктури з можливостями публічних хмар. Такий підхід забезпечує підприємства винятковою гнучкістю у використанні ресурсів і сприяє оптимізації витрат, що має вирішальне значення для їхньої конкурентоспроможності. Завдяки впровадженню гібридних хмар компанії можуть адаптувати свої системи інформаційних технологій (IT) до змінних потреб, зберігаючи високу продуктивність і належний рівень безпеки. Багатохмарні середовища, у свою чергу, виступають ще одним важливим стратегічним інструментом для підприємств. Використання декількох постачальників хмарних послуг дозволяє мінімізувати ризики, пов’язані із залежністю

від одного провайдера, що значно підвищує загальну надійність системи. Водночас це відкриває більше можливостей у виборі технологій, сприяючи створенню гнучкого й ефективного технологічного середовища для розвитку бізнесу. У підсумку компанії отримують змогу будувати стійкі до збоїв і потужні IT-екосистеми, здатні функціонувати навіть у найскладніших умовах [11-12].

Наочне розділення типів хмар можна переглянути на рисунку 1.1.

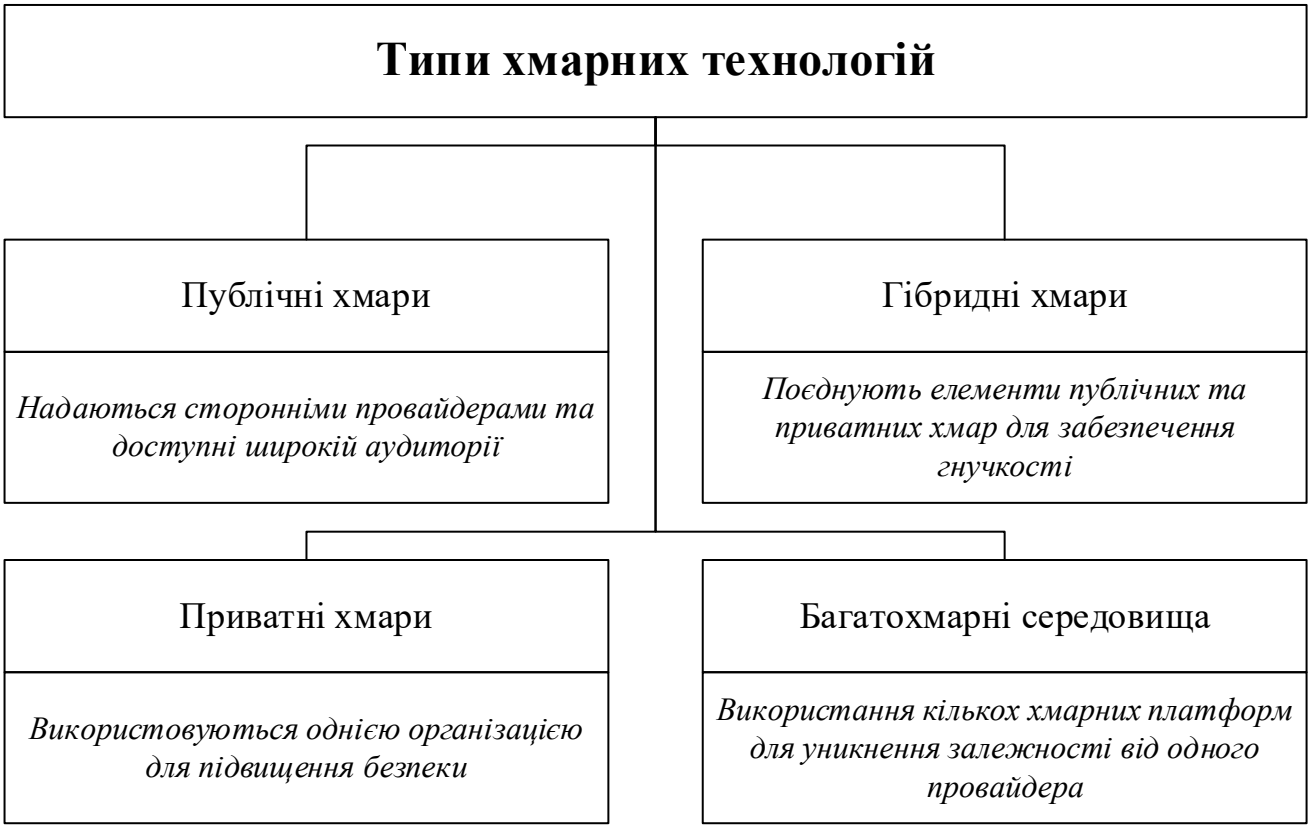


Рисунок 1.1 – Розділення хмарних технологій за типами

Хмарні технології відкривають нові горизонти для впровадження інновацій у бізнесі. Завдяки їхнім можливостям компанії отримують доступ до платформ, побудованих на основі штучного інтелекту та аналізу великих даних, що стало реальністю завдяки потужностям хмарних обчислень. Це дає змогу організаціям оперативніше ухвалювати продумані рішення та розробляти сучасні продукти [13-14].

Перевагами хмарних технологій є наступні аспекти:

- зниження витрат на обладнання, обслуговування та енергозбереження;
- легке масштабування ресурсів для адаптації до змін;
- можливість швидкого розгортання рішень;
- постійне оновлення технологій і доступ до нових функцій.

На противагу перевагам, недоліками хмарних технологій виступають наступні аспекти:

- ризик витоку даних та залежність від провайдера;
- неможливість роботи без стабільного з’єднання;
- користувачі мають менший контроль на інфраструктурою.

Крім того, слід ретельно враховувати питання відповідності законодавчим нормам, особливо для підприємств, що обробляють конфіденційну інформацію. Наприклад, застосування публічних хмарних сервісів може бути обмежене через регуляції, спрямовані на захист персональних даних [13-14].

Хмарні технології виступають важливим засобом для оптимізації бізнес-процесів і стимулювання інновацій у створенні цифрових продуктів. Їх успішне впровадження значною мірою залежить від ретельного вибору моделі, типу інфраструктури та постачальника послуг, а також від урахування потенційних загроз, пов’язаних із забезпеченням безпеки й збереженням конфіденційності даних. Зокрема, для досягнення максимальної гнучкості у функціонуванні та високого рівня надійності рекомендується застосування гібридних або мультихмарних архітектур. Очікується, що подальший розвиток хмарних технологій буде зосереджений на вдосконаленні заходів безпеки, розширенні можливостей інтеграції з іншими цифровими платформами та активному впровадженні передових технологій, таких як штучний інтелект і автоматизація. Така еволюція галузі створює нові перспективи для організацій різних масштабів і секторів

економіки, відкриваючи шлях до підвищення конкурентоспроможності та інноваційності [15].

1.2 Аналіз сучасних тенденцій у тестуванні хмарних систем

Хмарні технології стали невід'ємною частиною сучасних інновацій, забезпечуючи швидкий і доступний запуск різноманітних сервісів. Завдяки своїй універсальності, вони здатні відповідати запитам широкого спектра галузей – від фінансів до споживчих послуг. Однак разом із стрімким поширенням хмарних систем зростає потреба в їхньому детальному тестуванні, що є ключовим для гарантування стабільності та високої якості. Тестування таких систем відіграє вирішальну роль, оскільки дозволяє перевірити їхню функціональність, відповідність стандартам, ефективність автоматизації процесів і здатність підтримувати стабільну роботу при збільшенні навантажень. Це не лише зміцнює довіру користувачів до хмарних сервісів, а й мінімізує ризики технічних збоїв чи втрат даних [16-19].

Основними аспектами тестування хмарних систем є вибір виду тестування та інструментарію. Видами тестування хмарних систем є такі, як тестування надійності, продуктивності, функціональне, безпеки тощо [19-20].

Тестування надійності є критичним етапом оцінки, спрямованим на перевірку стабільності та доступності системи за умов підвищеного навантаження. Завдяки моделюванню пікових навантажень можна ідентифікувати потенційні вузькі місця, що впливають на загальну продуктивність. У хмарних середовищах особливий акцент робиться на правильному зонуванні даних та ефективному розподілі ресурсів між різними сервісами, аби мінімізувати ризики втрати доступу, які наочно наведені на рисунку 1.2. Враховуючи це, необхідно зважати на вплив оновлень програмного забезпечення (ПЗ) та змін у апаратному забезпеченні, які можуть впливати на стабільність роботи системи під час пікових навантажень. Використання методів тестування надійності, таких як відмовостійкість або хаос-інженерія, дає змогу оцінити здатність системи відновлюватись після несправностей. Окрім того, ретельно

аналізуються причини збоїв і перевіряється функціональність резервних механізмів, зокрема систем резервування даних чи перенаправлення мережевого трафіку [20].

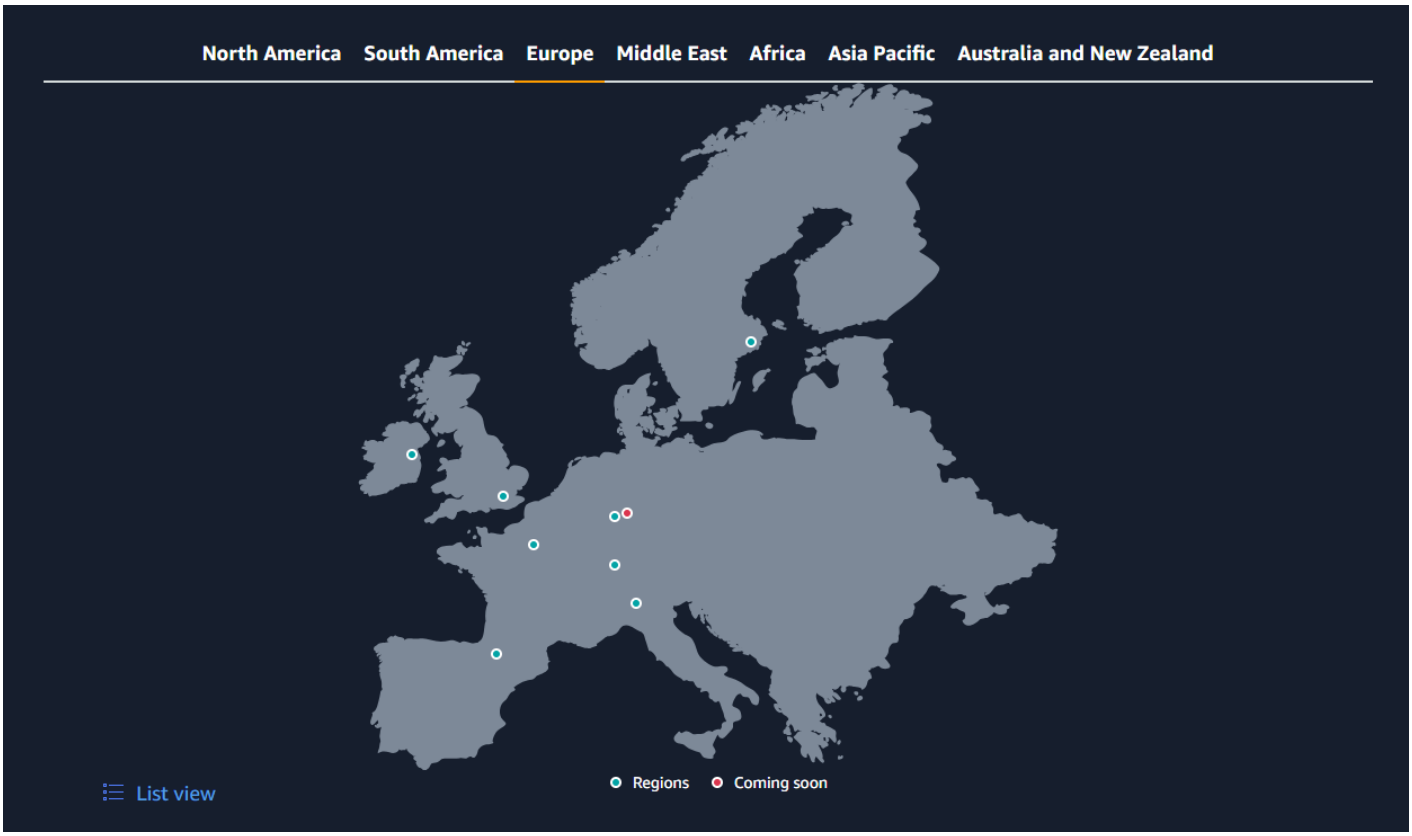


Рисунок 1.2 – Приклад зон надійності та поділ сервісів за зонами доступності на Amazon Web Services (AWS) в Європі [21]

Такий підхід (див. рис. 1.2) спрямований на забезпечення стабільної роботи сервісів навіть за умов непередбачуваних ситуацій, включаючи різке зростання активності користувачів або виникнення технічних проблем.

Тестування продуктивності передбачає аналіз часу реакції системи, швидкості виконання запитів та її стабільності під тривалим високим навантаженням. Завдяки такому підходу можна визначити потенціал масштабування сервісу та оцінити його здатність автоматично розподіляти навантаження між серверами, як наведено на рисунку 1.3. Цей процес зазвичай включає стрес-тестування, моделювання обчислювальних процесів і перевірку роботи в умовах максимального навантаження.

Окрім традиційних сценаріїв, тестування продуктивності охоплює також аналіз роботи системи в умовах раптових стрибків навантаження (спайкові тести) та оцінку її здатності адаптуватися до змін конфігурації. Особливу увагу приділяють вивченню впливу таких факторів, як обмежена пропускна здатність мережі, затримки або час відгуку бази даних (БД), на загальну ефективність системи [22].

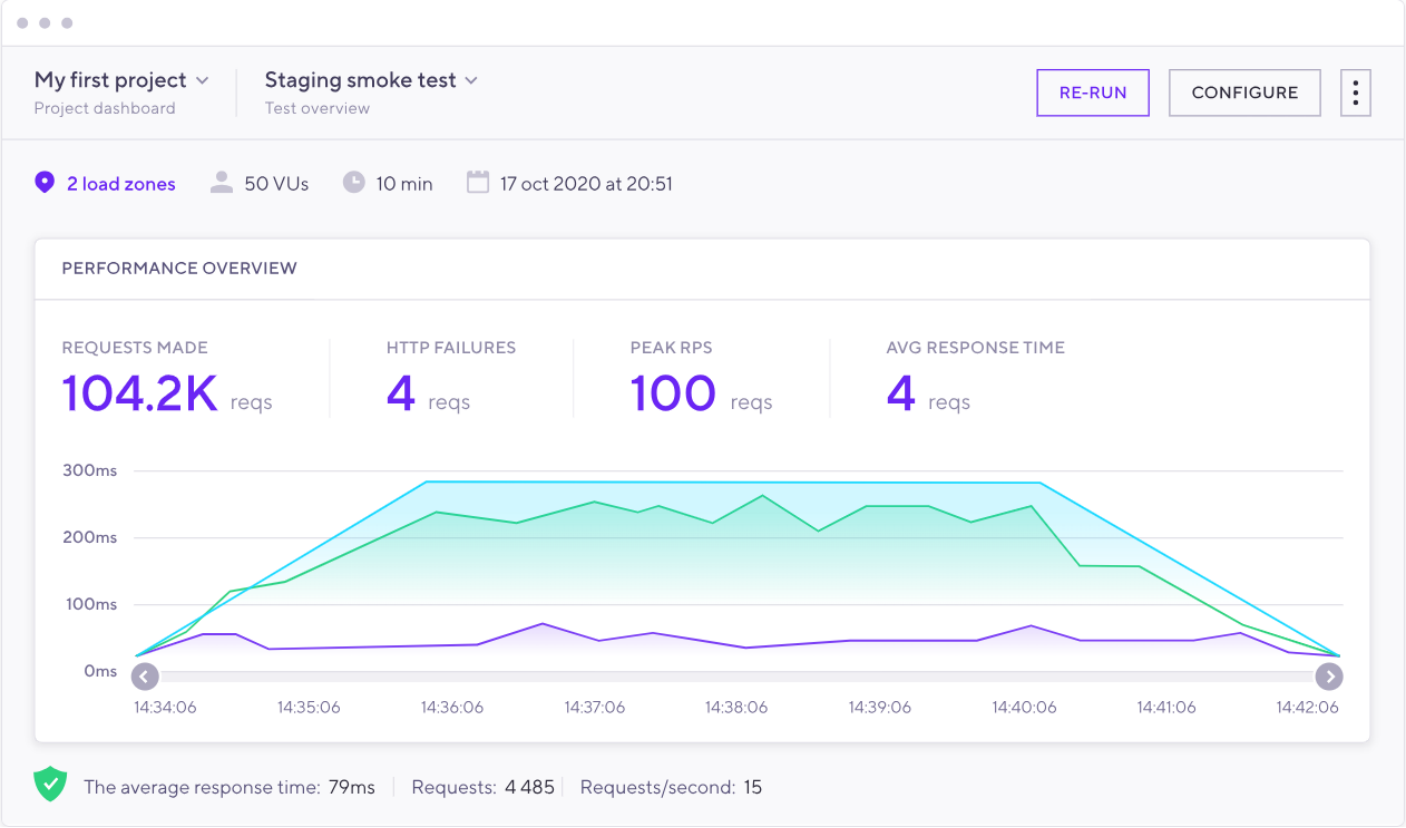


Рисунок 1.3 – Приклад графіку навантаження та відповіді системи в реальному часі від лабораторії K6 [23]

Процес тестування допомагає визначити, наскільки фактичні ключові показники продуктивності (KPI) відповідають очікуваним. Це можуть бути, наприклад, кількість запитів, оброблених за секунду, чи середній час відгуку для користувачів. Інструменти на кшталт Apache JMeter, Gatling або LoadRunner дозволяють створювати реалістичні тестові умови та збирати докладну аналітику для подальшого дослідження. Особливий акцент робиться на вивченні поведінки системи під час пікових навантажень. Це

необхідно, щоб оцінити її здатність відновлюватися після стресових ситуацій і забезпечувати безперервну роботу без погіршення якості обслуговування [24].

Функціональне тестування спрямоване на оцінку відповідності функціональних характеристик системи визначеним специфікаціям. Одним із ключових аспектів цього процесу є забезпечення належної інтеграції різноманітних сервісів, що взаємодіють між собою через application programming interface (API), як наведено на рисунку 1.4. До основних завдань тестування входить перевірка точності передачі даних, адекватності обробки запитів, а також стабільності роботи зовнішніх підключень. Зокрема, оцінюється здатність усіх компонентів системи працювати узгоджено навіть за умов тимчасової недоступності одного з сервісів [24].

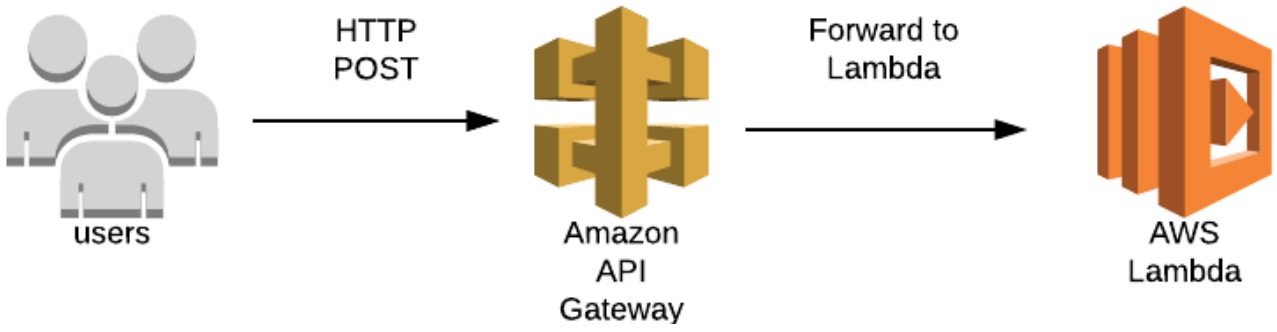


Рисунок 1.4 – Архітектурна схема API AWS та залежності між компонентами [25]

Функціональне тестування, зокрема його додаткові аспекти, зосереджується на ретельній оцінці відповідності бізнес-логіки системи встановленим вимогам, включаючи обробку транзакцій та виконання складних запитів користувачів. Особливу увагу приділяють перевірці крайових випадків, щоб оцінити поведінку системи у нестандартних умовах, таких як опрацювання даних у неочікуваних форматах, перевищення допустимого розміру вхідних файлів або обробка некоректно сформованих запитів. Автоматизація тестування функціональних можливостей є ключовим елементом підвищення ефективності процесу перевірки. Завдяки використанню таких інструментів, як Selenium чи Postman, можливо автоматизувати

сценарії тестування інтеграційної взаємодії, роботи API та моделювання користувацького досвіду у взаємодії з інтерфейсами. Такий підхід значно скорочує час виконання тестування та забезпечує вищу точність. Крім того, функціональне тестування враховує високу динамічність сучасних хмарних систем. Особливий акцент робиться на перевірці адаптивності сервісів до змін конфігурацій або процесів масштабування. Це дозволяє забезпечити безперервну роботу програмних рішень та відповідність очікуванням кінцевих користувачів навіть у періоди швидких змін, таких як розширення функціональних можливостей чи оновлення системного середовища [26].

Тестування безпеки має на меті гарантувати конфіденційність, цілісність та доступність даних. Для цього застосовуються спеціалізовані інструменти, що дозволяють виявляти вразливості, моделювати сценарії проникнення та аналізувати рівень захищеності від різноманітних атак, як наведено на рисунку 1.5. Особливу вагу приділяють перевіркам хмарних систем на відповідність міжнародним стандартам безпеки. Наприклад, проводяться оцінки стійкості до атак «людина посередині» або до спроб отримання несанкціонованого доступу до інформації [26].

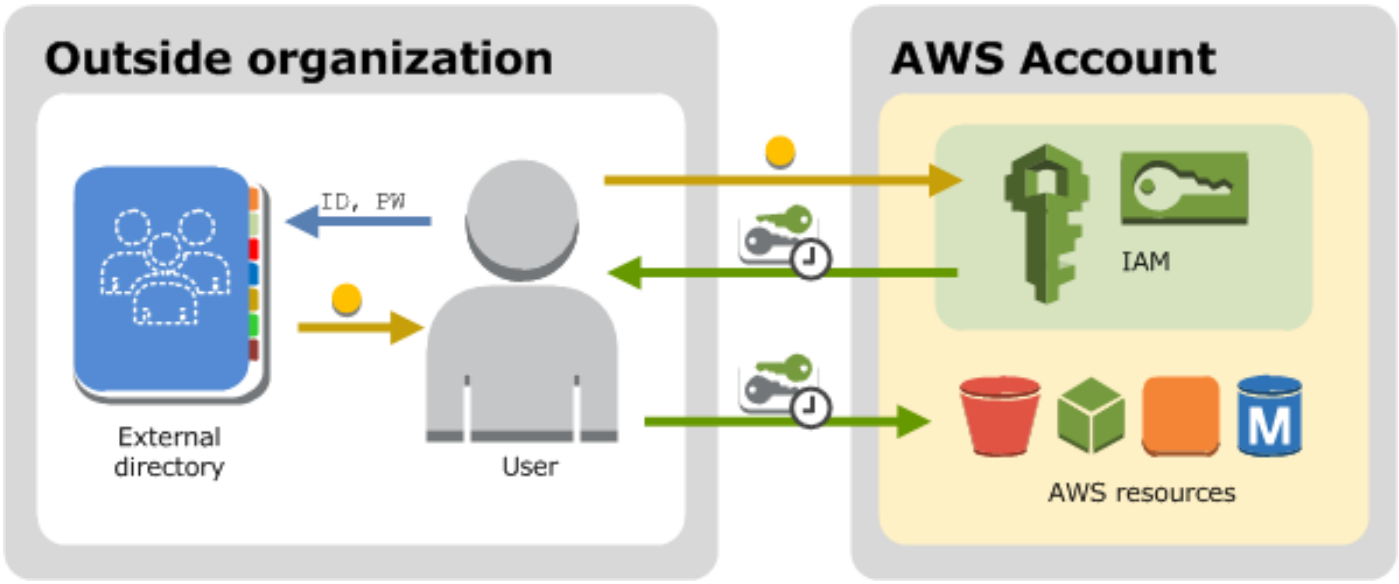


Рисунок 1.5 – Модель безпеки та ієрархії доступів AWS [27]

Розширене тестування безпеки передбачає ретельну перевірку використання ефективних методів шифрування для забезпечення безпечного зберігання та передачі даних, оцінку захищеності від розподілених атак типу відмови в обслуговуванні, а також аналіз механізмів автентифікації та авторизації користувачів. Особливий наголос робиться на ідентифікації потенційних уразливостей у налаштуванні мережевих компонентів, таких як міжмережеві екрани, шлюзи чи системи виявлення вторгнень. Ключовим елементом є також оцінка ефективності багатофакторної автентифікації, процедури відновлення доступу до системи після збоїв та забезпечення функціонування систем резервного копіювання. У межах тестування здійснюється моделювання можливих сценаріїв атак, що охоплюють внутрішні загрози, техніки соціальної інженерії та використання вразливостей у програмному забезпеченні. Додатково досліджується здатність системи протидіяти змінам зовнішнього середовища, зокрема таким явищам, як несподіване перевантаження серверів або модифікація рівнів доступу для різних категорій користувачів. Такий підхід дозволяє переконатися, що хмарна система відповідає як технічним стандартам, так і вимогам регуляторів у сфері безпеки, гарантуючи високий рівень захисту даних користувачів [28].

Сучасні хмарні системи вимагають використання інструментів, які забезпечують високий рівень автоматизації та масштабованості [29-30]. Порівняння найпопулярніших наведено в таблиці 1.2.

Таблиця 1.2 – Інструменти, забезпечуючі автоматизацію та масштабованість хмарних систем

Найменування інструментарію	Характеристика інструментарію
1	2
Apache JMeter	Потужний інструмент для тестування продуктивності, який дозволяє виконувати багатопотокові тести в різних середовищах
Selenium	Інструмент для автоматизації функціонального тестування веб-застосунків

Продовження таблиці 1.2

1	2
OWASP ZAP	Використовується для пошуку та усунення вразливостей у вебдодатках
Postman	Забезпечує зручність тестування API, перевірки інтеграції та обміну даними між модулями

На сьогодні існують основні тенденції в тестуванні хмарних систем. Вони засновані на автоматизації, контейнеризації та оркестрація, на основі ШІ, орієнтація на користувальницький досвід (тестування user experience (UX-тестування), мультихмарний підхід [31-34].

Характеристика сучасних тенденцій наведена в таблиці 1.3.

Таблиця 1.3 – Сучасні тенденції в тестуванні хмарних систем

Тенденція	Характеристика тенденції
1	2
Автоматизація тестування	У сучасному процесі тестування автоматизація виступає ключовим компонентом, що істотно впливає на ефективність і якість забезпечення ПЗ. Використання автоматизованих тестів сприяє значному скороченню обсягу ручної праці, мінімізує ймовірність виникнення людських помилок та забезпечує оперативний зворотний зв’язок у рамках циклу розробки. Особливе значення в цьому контексті мають інструменти continuous integration / continuous deployment (CI/CD), такі як Jenkins або GitLab CI тощо
Контейнеризація та оркестрація	Застосування контейнерів (Docker) у поєднанні з платформами оркестрації (Kubernetes) сприяє створенню адаптивних та високоефективних середовищ для тестування ПЗ. Використання контейнеризації значно спрощує моделювання реальних умов експлуатації, забезпечуючи гнучкість у зміні конфігурацій і адаптації середовища до специфічних вимог. Такий підхід мінімізує витрати часу на підготовку тестових середовищ і сприяє підвищенню загальної ефективності тестових операцій

Продовження таблиці 1.3

1	2
Тестування на основі ШІ	Використання ШІ та машинного навчання у процесах тестування забезпечує можливість автоматичного аналізу результатів, виявлення потенційних помилок і створення сценаріїв, що враховують поведінку користувачів. Крім того, алгоритми AI дозволяють прогнозувати, як зміни у коді можуть вплинути на функціонування системи, що має особливе значення для масштабних проєктів. Зокрема, моделі машинного навчання здатні заздалегідь визначати ймовірні точки збоїв, запобігаючи їх появі
UX-тестування	Охоплює аналіз зручності використання хмарних сервісів, їх доступності та швидкості виконання різноманітних операцій. Використання реальних користувачів для тестування або створення моделі їхньої поведінки допомагає вдосконалювати інтерфейс і підвищувати загальний рівень задоволеності системою. Значну увагу приділяють метрикам на кшталт Net Promoter Score (NPS) і коефіцієнтам утримання користувачів. Наприклад, тестування мобільних додатків у реальних мережах дозволяє виявити проблеми з продуктивністю у слабкому покритті
Мультихмарний підхід	У сучасних умовах підприємства дедалі частіше впроваджують мультихмарну стратегію, що значно ускладнює процес тестування ПЗ. Тестувальникам необхідно забезпечити сумісність між такими платформами, як Amazon Web Services (AWS), Azure і Google Cloud, а також провести всебічну перевірку коректності функціонування у гетерогенних середовищах. Така діяльність охоплює ретельну оцінку роботи API, аналіз ефективності розподілу ресурсів і вивчення продуктивності сервісів у контексті взаємодії в різних хмарних середовищах. Наприклад, розгортання єдиного застосунку на кількох хмарних платформах дозволяє виявити потенційні проблеми з конфігурацією або недостатньою оптимізацією системи

Тестування хмарних систем у сучасних умовах є складним і багатогранним процесом, що інтегрує традиційні методології з інноваційними підходами, такими як автоматизація, контейнеризація та використання штучного інтелекту (див. табл. 1.3). Застосування спеціалізованих інструментів разом зі врахуванням новітніх тенденцій у цій сфері сприяє забезпеченню високої якості наданих сервісів та відповідає зростаючим потребам користувачів. Використання схем, графіків і аналітичних моделей відіграє

ключову роль у проведенні глибокого аналізу результатів тестування, дозволяючи оптимізувати процеси в умовах швидкого розвитку технологій.

1.3 Аналіз тестування продуктивності та масштабованості в хмарних системах

Тестування продуктивності та масштабованості хмарних систем є невід'ємним етапом забезпечення їхньої надійності, ефективності та відповідності потребам користувачів. У сучасному світі, де хмарні технології стали фундаментом цифрової інфраструктури, їх безперебійна робота набуває вирішального значення для бізнесу, особливо в умовах динамічних змін, таких як різке зростання навантаження або розширення функціональних можливостей. Тестування продуктивності дозволяє оцінити, наскільки швидко й стабільно система виконує свої завдання в різноманітних сценаріях, тоді як тестування масштабованості спрямоване на виявлення можливості системи ефективно працювати за збільшення ресурсів чи кількості користувачів [35].

Основним завданням подібного тестування є створення стійкої системи, здатної функціонувати стабільно за будь-яких обставин. Наприклад, навантажувальне тестування моделює типові сценарії використання, дозволяючи перевірити відповідність роботи системи встановленим стандартам у звичайних умовах експлуатації. Не менш важливим є стресове тестування, метою якого є поступове підвищення навантаження до моменту виникнення збоїв. Цей підхід дозволяє виокремити критичні вузли й визначити межі працездатності системи. Окрему увагу заслуговує тестування пікових навантажень, яке оцінює здатність системи справлятися з короткочасними та інтенсивними сплесками активності користувачів. Це особливо важливо для інтернет-магазинів під час розпродажей чи стрімінгових платформ у періоди прем'єрних подій [35-36].

Тривалі перевірки, які часто називають тестуванням стійкості, спрямовані на оцінку здатності системи витримувати постійне навантаження протягом значного періоду часу. Цей підхід є ключовим для забезпечення стабільної експлуатації системи в умовах безперервного використання. Поряд із цим, тестування масштабу виконує аналіз адаптивних спроможностей системи при відповідних змінах, зокрема шляхом

розширення обчислювальних ресурсів. Це дозволяє здійснювати прогнозування її поведінки у випадках зростання бізнес-потреб або раптового збільшення кількості користувачів [36].

Ефективне проведення тестувань передбачає використання спеціалізованих інструментів. Apache JMeter виступає засобом імітації навантаження для вебдодатків і сервісів. Locust застосовується для розподіленого тестування, що базується на сценаріях, створених за допомогою мови Python. Хмарна платформа Blazemeter надає можливість детального аналізу продуктивності навіть на великих масштабах, тоді як Gatling широко використовується для оцінки продуктивності вебдодатків, орієнтованих на забезпечення високої масштабованості. У середовищах хмарних обчислень, таких як AWS і Google Cloud, можна ефективно інтегрувати інструменти на кшталт AWS CloudWatch та Google Cloud Monitoring. Вони забезпечують моніторинг у реальному часі та надають дані для комплексного аналізу стану системи [37].

Оцінювання результатів тестування базується на ретельному аналізі ключових показників ефективності. Наприклад, показник часу відгуку системи характеризує її здатність оперативно взаємодіяти з користувачами, дозволяючи оцінити швидкість виконання запитів. Пропускна здатність відображає кількість запитів, які система може обробляти за одиницю часу, таким чином визначаючи її продуктивність. Використання ресурсів, таких як центральний процесор, оперативна пам’ять і мережеві ресурси, дозволяє оцінити ефективність застосування наявних обчислювальних потужностей. Додатково, аналіз частоти помилок допомагає ідентифікувати вразливі аспекти системи, що вимагають посиленої уваги та доопрацювання [38].

Дані, отримані в процесі тестування, слугують основою для розробки оптимізаційних заходів щодо архітектури системи. Наприклад, впровадження мікросервісної архітектури чи застосування контейнеризації може суттєво збільшити її гнучкість та адаптаційну здатність до змінних умов роботи. Критично важливою складовою модернізації є реалізація механізмів динамічного масштабування, які забезпечують автоматичне регулювання обсягу ресурсів залежно від поточного

навантаження. Крім того, використання засобів моніторингу у реальному часі, наприклад систем Application Performance Monitoring (APM), дозволяє завчасно виявляти потенційні аномалії та швидко реагувати на них. Інструменти гешування, у свою чергу, сприяють зниженню навантаження на бази даних і покращенню загальної продуктивності системи [39].

Таким чином, тестування продуктивності та масштабованості є ключовим етапом у процесі розробки та експлуатації ефективних хмарних рішень. Воно мінімізує ризики відмов функціонування, сприяє раціональному використанню ресурсів і покращує взаємодію користувачів із системою. Завдяки таким підходам забезпечується стабільна та ефективна робота навіть у надзвичайно високонавантажених сценаріях.

1.4 Аналіз підходів тестування продуктивності та масштабованості в хмарі

Хмарні середовища відіграють ключову роль у функціонуванні сучасних систем, тому забезпечення якості через тестування їх продуктивності та здатності до масштабування є надзвичайно важливим етапом. Основними методами, що використовуються в цьому процесі, є навантажувальне тестування, перевірка масштабованості, стрес-тестування та оцінка відмовостійкості.

1.4.1 Тестування навантаження

Навантажувальне тестування є одним із найпоширеніших методів перевірки продуктивності системи. Цей підхід дозволяє оцінити, як добре система функціонує за умов прогнозованої активності, наприклад, коли сервер одночасно обробляє запити від певної кількості користувачів. Основною метою такого тестування стає визначення часу відгуку, пропускної здатності та стабільності системи при звичайному використанні. У хмарному середовищі цей процес дає змогу моделювати роботу тисяч або навіть мільйонів користувачів за допомогою автоматизованих інструментів, які створюють трафік і надають аналітику результатів. При цьому важливо не лише фіксувати

показники часу відгуку, а й досліджувати поведінку системи при поступовому збільшенні навантаження, відстежуючи її ефективність у процесі зростання кількості користувачів [40-41].

Навантажувальне тестування є важливим інструментом, який дозволяє ідентифікувати слабкі місця в архітектурі системи, здатні призводити до зниження продуктивності за умов інтенсивної активності користувачів. До таких проблем можуть належати обмеження в продуктивності БД, недостатня пропускна здатність мережевого обладнання чи нераціональне використання системних ресурсів. Проведення детального аналізу цих аспектів дає змогу своєчасно впроваджувати оптимізаційні заходи, мінімізувати затримки та покращувати загальну ефективність функціонування системи. Крім того, ключовим етапом є моделювання різноманітних сценаріїв взаємодії користувачів із системою. Наприклад, це може стосуватися перевірки продуктивності в періоди рівномірного розподілу запитів протягом години чи дня або за умов різких пікових навантажень, таких як масовий одночасний доступ великої кількості користувачів під час проведення акційних розпродажів чи запуску нових функціональних можливостей. Такий підхід дозволяє оцінити стабільність роботи системи у реальних умовах експлуатації та адаптацію до непередбачуваних змін рівня навантаження [40-41].

Використання хмарних платформ для навантажувального тестування забезпечує додаткові переваги, зокрема гнучке масштабування та легкий доступ до необхідних обчислювальних ресурсів для моделювання навіть найскладніших сценаріїв. Застосування спеціалізованих автоматизованих інструментів, таких як Apache JMeter, k6 або Locust, спрощує процес проведення тестувань, значно знижуючи пов’язані витрати. Це дає змогу зосередити основну увагу на інтерпретації результатів і впровадженні критично важливих покращень [41].

1.4.2 Тестування масштабованості

Дослідження масштабованості зосереджується на аналізі здатності системи адаптуватися до змін у доступних ресурсах або умовах її експлуатації. У середовищах хмарних обчислень ця характеристика відіграє вирішальну роль, оскільки забезпечує автоматичне регулювання ресурсів відповідно до поточного рівня навантаження. Наприклад, у разі раптового зростання кількості користувачів система повинна оперативно долучати додаткові сервери або підвищувати потужність наявних компонентів, що дозволяє підтримувати стабільний рівень продуктивності. Під час здійснення тестування масштабованості з'ясовується, наскільки ефективно функціонують механізми адаптації і як швидко система реагує на зміни середовища. Окрім того, аналізу підлягають затримки, що супроводжують процес масштабування, а також їхній вплив на загальну продуктивність роботи системи у перехідний період. У певних випадках збільшення ресурсів може супроводжуватися тимчасовими порушеннями або зниженням швидкості обробки запитів, що потенційно здатне негативно вплинути на користувацький досвід. Для мінімізації таких наслідків надзвичайно важливим є тестування подібних сценаріїв, оскільки воно сприяє визначенню оптимальних параметрів масштабування, які забезпечують стабільну роботу системи й зменшують вплив на кінцевих користувачів [42-43].

Особливо важливо оцінювати здатність системи до масштабування як у горизонтальному напрямку, додаючи нові сервери чи вузли, так і у вертикальному, підвищуючи продуктивність наявних компонентів. Обидва підходи мають свої сильні сторони та обмеження, а тестування допомагає визначити, який із них краще відповідає потребам конкретної ситуації. Наприклад, горизонтальне масштабування є оптимальним для систем, які обробляють великий обсяг подібних запитів, тоді як вертикальне підходить для виконання складних обчислювальних завдань [42-43].

Аналіз масштабованості також охоплює перевірку роботи автоматизованих механізмів у хмарних середовищах, таких як AWS Auto Scaling, Google Cloud Instance Groups або Kubernetes Horizontal Pod Autoscaler (HPA). Завдяки тестуванню визначають,

наскільки швидко ці інструменти реагують на зміну навантаження та чи здатні вони забезпечити стабільну продуктивність. Водночас важливо перевірити, чи уникнуто перевитрати ресурсів під час масштабування, адже це безпосередньо впливає на оптимізацію витрат на інфраструктуру [42-43].

Зрештою, тестування масштабованості дозволяє отримати детальне розуміння поведінки системи в умовах змін. Це допомагає своєчасно виявляти та усувати потенційні проблеми, які можуть негативно вплинути на її продуктивність та загальну ефективність.

1.4.3 Стрес-тестування та тестування відмовостійкості

Стрес-тестування та тестування відмовостійкості є незамінними практиками для оцінки здатності системи функціонувати у складних та позаштатних обставинах. Стрес-тестування спрямоване на виявлення меж можливостей системи через створення надмірного навантаження, яке перевищує стандартні робочі показники. Це дозволяє визначити моменти, коли система втрачає стабільність або повністю виходить з ладу. Як приклад, можна змоделювати сценарії із різким збільшенням кількості одночасних запитів чи обмеженням доступу до ключових ресурсів, таких як процесорна потужність або оперативна пам’ять. Аналізуючи поведінку системи за таких умов, можна виявити слабкі місця, які потребують оптимізації, зокрема алгоритми обробки даних, балансування потоків чи управління чергами. Це тестування надає важливе уявлення про те, як система працює у критичних ситуаціях. Наприклад, при різких піках навантаження, коли кількість запитів збільшується у десятки чи навіть сотні разів за лічені секунди, необхідно оцінити, чи зберігається хоча б базова функціональність. Результати стрес-тестів дозволяють командам розробників і адміністраторів завчасно підготуватися до таких сценаріїв у реальних умовах: раптових маркетингових кампаній, сезонних розпродажів або потенційних кібератак [44-45].

Тестування відмовостійкості має інше спрямування – оцінку здатності системи залишатися працездатною навіть у разі виникнення збоїв. Воно часто включає

симулювання ситуацій на кшталт недоступності одного з серверів, втрати підключення до БД або виходу з ладу зовнішнього сервісу. Такі перевірки допомагають оцінити ефективність механізмів автоматичного відновлення, таких як резервне копіювання серверів, перенаправлення трафіку, повторне виконання завислих запитів чи перемикання на альтернативні сервіси. Для цього активно використовуються техніки Chaos Engineering, що дозволяють створювати реалістичні умови збоїв у контрольованому середовищі. Наприклад, інструменти на кшталт Chaos Monkey цілеспрямовано відключають критичні частини інфраструктури, що дає змогу перевірити здатність системи справлятися з подібними ситуаціями. Такі експерименти допомагають максимально детально оцінити реакцію системи на складні події та ідентифікувати її вразливості. Особливу увагу в рамках тестування відмовостійкості приділяють швидкості відновлення працездатності після усунення проблем. Наприклад, критично важливо перевірити коректність відновлення даних, збереження цілісності транзакцій і відсутність накопичення вторинних помилок. Це дозволяє забезпечити стабільну роботу системи навіть у кризових ситуаціях та мінімізувати негативний вплив на кінцевих користувачів [46-47].

Обидва підходи мають важливе значення для забезпечення надійності сучасних систем. Вони допомагають не лише ідентифікувати потенційні проблеми, але й підготувати систему до роботи за екстремальних умов, підвищуючи стійкість до збоїв і безперервність обслуговування.

Висновки до розділу 1

Хмарні технології стали ключовим елементом сучасної цифрової трансформації, надаючи бізнесу можливості для масштабування, гнучкості та раціонального використання ресурсів. Їх ефективне впровадження допомагає компаніям оптимізувати бізнес-процеси, розробляти інноваційні рішення та швидко реагувати на динамічні зміни ринкових умов. Однак для досягнення стабільної роботи, високої продуктивності та належного рівня безпеки хмарних систем необхідно застосовувати комплексний підхід

до їх тестування. У запропонованому огляді акцент зроблено на таких методах тестування, як навантажувальне, стресове, тестування масштабованості та відмовостійкості. Ці підходи дозволяють оцінити, наскільки ефективно система функціонує у стандартних, критичних і пікових умовах. Завдяки сучасним інструментам можна автоматизувати перевірку систем, мінімізувати витрати на тестування та підвищити точність отриманих результатів. Аналізуючи актуальні тенденції, можна стверджувати, що успіх у тестуванні хмарних систем залежить від інтеграції традиційних підходів із передовими технологіями, такими як контейнеризація, оркестрація та автоматизація процесів.

РОЗДІЛ 2 ПРОВЕДЕННЯ НАУКОВОГО ДОСЛІДЖЕННЯ ЗА МЕТОДИКАМИ

2.1 Аргументування вибору тематики дослідження

Розвиток хмарних технологій суттєво трансформує сучасні методи зберігання, оброблення та передачі даних, що має критичне значення для інформаційного суспільства. Хмарні платформи демонструють високий рівень адаптивності, масштабованості та економічної рентабельності, що стимулює їх широке впровадження в різноманітні сфери, такі як фінансовий сектор, охорона здоров’я, промисловість і державне управління. Проте збільшення популярності цих технологій супроводжується низкою викликів, серед яких особливу увагу привертає питання забезпечення продуктивності та масштабованості хмарних систем. Зокрема, виникає необхідність у вдосконаленні їхньої ефективності в умовах непередбачуваних і динамічних навантажень.

Методології тестування продуктивності та масштабованості відіграють визначальну роль у процесі оцінювання ефективності хмарних систем. Вони дозволяють ідентифікувати структурні вузькі місця, оптимізувати розподіл ресурсів і гарантувати стабільність роботи навіть за умов інтенсивних навантажень. З огляду на швидку еволюцію хмарних технологій, наукові дослідження в цьому напрямі залишаються вкрай актуальними та затребуваними.

Обрана тематика дослідження спрямована на розробку нових підходів до тестування продуктивності та масштабованості хмарних обчислювальних систем. Наукова новизна цього проєкту полягає у створенні алгоритмів та метрик оцінки, адаптованих до сучасних тенденцій розвитку хмарних технологій. У межах дослідження заплановано акцентувати увагу на декількох ключових напрямах:

- використання методів симуляції навантажень для моделювання реальних умов функціонування хмарних систем;

- інтеграція сучасних засобів тестування з метою підвищення точності та достовірності результатів оцінки продуктивності;

- здійснення порівняльного аналізу різних підходів до тестування, включно зі стрес-тестуванням, навантажувальним тестуванням та тестуванням відмовостійкості.

Очікується, що отримані результати сприятимуть покращенню рівня надійності і стійкості хмарних платформ. Це є особливо важливим для підприємств, які займаються наданням хмарних послуг, адже ефективність таких платформ безпосередньо впливає на рівень їхньої комерційної спроможності та конкурентоспроможності.

Результати проведеного дослідження мають значний практичний потенціал і можуть бути інтегровані в низку прикладних сфер. Запропоновані методологічні підходи та алгоритми здатні забезпечити такі можливості:

- оптимізацію функціонування наявних хмарних платформ шляхом системного виявлення та усунення так званих вузьких місць, що обмежують їхню продуктивність;

- розроблення нових хмарних сервісів, адаптованих до специфічних вимог щодо продуктивності, масштабованості та стабільності, що є ключовими викликами сучасної галузі;

- підвищення рівня ефективності управління ресурсами в хмарних системах шляхом удосконалення підходів до їхньої організації та координації.

Застосування отриманих результатів також окреслює перспективи зростання конкурентоспроможності компаній, що спеціалізуються на розробленні хмарних технологій. Це може бути досягнуто шляхом впровадження більш продуктивних, надійних та інноваційних технологічних рішень, що відповідають сучасним ринковим потребам.

Актуальність обраної тематики зумовлена її ключовим значенням для розв’язання критичних наукових і прикладних завдань у галузі інженерії програмного забезпечення. Вибір даного напряму досліджень продиктований дедалі більшою

потребою у створенні ефективних методик для оцінювання продуктивності та масштабованості хмарних обчислювальних систем, які здатні відповідати динамічним викликам сучасного ринку. Очікується, що результати виконаних досліджень зроблять суттєвий внесок у подальший розвиток технологій хмарних обчислень, сприяючи підвищенню їхньої стійкості та надійності в умовах постійно зростаючих навантажень.

2.2 Метрики оцінювання методів тестування хмарного сховища

2.2.1 Пропускна здатність

Пропускна здатність визначає кількість даних, які система може обробити за одиницю часу та вимірюється в мегабайтах за секунду (МБ/с) або запитах за секунду (RPS):

$$TP = \frac{D_{total}}{T_{total}}, \tag{2.1}$$

де TP – пропускна здатність;

D_{total} – загальний обсяг даних, оброблений за певний період;

T_{total} – загальний час обробки.

Дана метрика дозволяє оцінити, наскільки ефективно система справляється з великими обсягами даних.

2.2.2 Час виконання операцій

Час виконання операцій визначає, наскільки швидко система може обробити окремий запит:

$$T_{avg} = \frac{\sum_{i=1}^{n} T_i}{n}, \tag{2.2}$$

де T_{avg} – середній час виконання;

T_i – час виконання конкретного запиту;

n – кількість запитів.

Дана метрика забезпечує вимір швидкості роботи системи, що безпосередньо впливає на якість користувальницького досвіду.

2.2.3 Відсоток помилок

Відсоток помилок оцінює частку операцій, які завершилися з помилками:

$$E_{rate} = \frac{N_{errors}}{N_{total}} \times 100\%, \tag{2.3}$$

де E_{rate} – відсоток помилок;

N_{errors} – кількість операцій помилкових;

N_{total} – загальна кількість операцій.

Дана метрика допомагає виявити проблеми в системі та оцінити її надійність.

2.2.4 Використання ресурсів

Використання ресурсів показує рівень використання системних ресурсів, таких як центрального процесору, оперативної пам’яті чи місце на диску:

$$R_{usage} = \frac{R_{used}}{R_{total}} \times 100\%, \tag{2.4}$$

де R_{usage} – використання ресурсу;

R_{used} – обсяг використаного ресурсу;

R_{total} – загальний обсяг ресурсу.

Аналіз даної метрики дозволяє оптимізувати використання інфраструктури для досягнення максимального ефекту.

2.2.5 Лінійність масштабування

Лінійність масштабування визначає зміну продуктивності системи зі збільшенням ресурсів:

$$S_{linear} = \frac{P_n}{P_1}, \tag{2.5}$$

де S_{linear} – коефіцієнт масштабування;

P_n – продуктивність при n одиницях ресурсу;

P_1 – продуктивність при одній одиниці ресурсу.

Дана метрика оцінює ефективність роботи системи в процесі розширення її ресурсів.

2.2.6 Еластичність

Еластичність вимірює здатність системи адаптуватися до змін у навантаженні:

$$E = \frac{T_{scaling}}{T_{ideal}}, \tag{2.6}$$

де E – коефіцієнт еластичності;

$T_{scaling}$ – реальний час адаптації;

T_{ideal} – ідеальний час адаптації.

Метрика еластичності дозволяє оцінити здатність системи підтримувати продуктивність за різних умов.

2.2.7 Максимальне навантаження

Метрика максимального навантаження визначає, чи зможе його витримати система без зниження продуктивності:

$$L_{max} = \left(N_{requests}\right), \tag{2.7}$$

де L_{max} – максимальне навантаження;

$N_{requests}$ – кількість одночасних запитів.

Визначення даного показника є важливим для оцінки стійкості системи в умовах пікових навантажень.

2.2.8 Гнучкість горизонтального масштабування

Гнучкість горизонтального масштабування визначає здатність системи підтримувати продуктивність при зміні кількості серверів:

$$F_{scaling} = \frac{P_{new}}{P_{old}}, \tag{2.7}$$

де $F_{scaling}$ – коефіцієнт гнучкості;

P_{new} – продуктивність після збільшення кількості серверів;

P_{old} – продуктивність до зміни.

Дана метрика допомагає визначити, наскільки ефективно система використовує додаткові ресурси для підвищення своєї продуктивності.

2.3 Розробка експериментів і тестових сценаріїв

Розробка експериментів і тестових сценаріїв відіграють ключову роль у дослідженні продуктивності та масштабованості хмарних обчислювальних систем. Цей етап передбачає моделювання робочих умов, максимально наближених до реальних, з метою відтворення навантажень на систему, а також визначення критичних показників для об’єктивної оцінки її функціональної ефективності. Основною метою таких експериментів є аналіз продуктивності хмарної системи за різних операційних умов, визначення її граничних можливостей і перевірка здатності до адаптації в умовах змінного навантаження. У рамках цієї діяльності окреслюються наступні ключові завдання:

- визначення рівня продуктивності системи за умов стандартного робочого навантаження;
- оцінювання її стійкості в умовах максимальних пікових навантажень;
- аналіз адаптивності та еластичності системи зі зміною доступних ресурсів;
- проведення порівняльного аналізу продуктивності залежно від застосовуваних тестових стратегій.

Таким чином, комплексний підхід до розробки експериментів дозволяє не лише оцінити поточний стан системи, але й сприяти її оптимізації для забезпечення високої надійності та ефективності в динамічному середовищі хмарних обчислень.

В якості використовуваних метрик і параметрів, за якими буде здійснюватися порівняльний аналіз, будуть використовуватися ті, що описані вище в підрозділі 2.2.

Методами тестування будуть ті, що згадувалися в підрозділі 1.4. Їх деталізація наведена в таблиці 2.1.

Таблиця 2.1 – Тестові сценарії методів тестування хмарних сховищ

Тестовий сценарій	Опис проведення			
	Мета сценарію	Приклад сценарію		
		Початкові умови	Етапи	Очікувані результати
1	2	3	4	5
Тестування навантаження	Визначити продуктивність системи під час поступового збільшення навантаження	Система знаходиться у стані готовності, початкове навантаження становить 10 запитів за секунду	1. Створити запити до системи зі швидкістю 10 RPS. 2. Поступово збільшувати швидкість запитів на 5 RPS кожні 30 секунд. 3. Продовжувати тестування до досягнення межі продуктивності (пропускна здатність починає знижуватися або час виконання операцій різко зростає)	Система підтримує стабільну продуктивність до певного порогу навантаження
Тестування масштабованості	Перевірити здатність системи автоматично адаптуватися до змін навантаження	Система працює зі стандартним навантаженням	1. Різко збільшити навантаження вдвічі. 2. Моніторити час, необхідний для автоматичного масштабування. 3. Після стабілізації навантаження зменшити його до початкового рівня.	Система автоматично збільшує або зменшує ресурси залежно від навантаження

Продовження таблиці 2.1

1	2	3	4	5
Тестування масштабованості			4. Виміряти швидкість та ефективність звільнення ресурсів	
Стрес-тестування	Оцінити поведінку системи за умов надмірного навантаження	Система працює під типовим навантаженням	1. Різко збільшити кількість запитів на 200% від номінального значення. 2. Продовжувати генерувати навантаження протягом 5 хвилин. 3. Моніторити відсоток помилок, час виконання операцій і використання ресурсів	Система повинна зберігати працездатність (з допустимим рівнем помилок) або автоматично масштабуватись для адаптації до навантаження
Тестування відмовостійкості	Оцінити здатність системи відновлюватися після збоїв	Системи працює під номінальним навантаженням	1 Імітувати відключення одного із серверів к кластері. 2. Змінювати навантаження, щоб оцінити поведінку системи при зменшенні ресурсів. 3. Відновити сервер і перевірити час повернення системи до нормального стану	Система має перенаправляти запити до інших серверів без значних помилок

Детальне опрацювання тестових сценаріїв (див. табл. 2.1) сприяє комплексній перевірці системи на продуктивність і масштабованість. Такий метод дозволяє ідентифікувати обмеження та вдосконалювати функціонування хмарних платформ, забезпечуючи їхню надійну роботу в умовах реального використання.

Висновки до розділу 2

У другому розділі здійснено ґрунтовний аналіз визначальних аспектів тестування продуктивності та масштабованості хмарних систем. Обґрунтовано вибір теми дослідження, акцентуючи увагу на її актуальності та значущості для сучасних хмарних платформ. Крім того, чітко окреслено мету дослідження, спрямовану на вдосконалення якості функціонування таких систем.

У межах аналізу визначено ключові показники для оцінки ефективності методів тестування. Особливу увагу приділено поглибленому вивченню й опису основних метрик. Розроблено тестові сценарії. Створені сценарії дозволяють діагностувати основні недоліки хмарних систем у різноманітних умовах експлуатації. Зокрема, вони охоплюють моделювання інтенсивного навантаження, перевірку відмовостійкості та аналіз механізмів масштабування.

Загалом цей розділ слугує теоретичним і прикладним підґрунтям для подальших досліджень та експериментів у зазначеній сфері. Розроблено і систематизовано принципові підходи до оцінки хмарних платформ, що сприяє формуванню більш аргументованих рекомендацій щодо їх оптимізації. Отримані результати виступають засадничою основою для інноваційного впровадження у хмарних технологіях з метою підвищення їхньої надійності та ефективності.

РОЗДІЛ 3 ДОСЛІДЖЕННЯ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ТЕСТУВАННЯ ПРОДУКТИВНОСТІ ТА МАСШТАБОВАНОСТІ В ХМАРНИХ СИСТЕМАХ

3.1 Виконання проєктування системи

3.1.1 Аргументування інструментарію тестування

Для забезпечення якісного тестування продуктивності та масштабованості хмарних систем важливо правильно підібрати інструменти, здатні ефективно оцінювати ключові метрики та реалізовувати тестові сценарії, наведених у таблиці 2.1. Критерії відбору таких інструментів ґрунтуються на наступних аспектах [56-60]:

- підтримка різноманітних метрик, включаючи пропускну здатність, час виконання операцій, відсоток помилок, використання ресурсів, еластичність, здатність до горизонтального масштабування та показники максимально допустимого навантаження, які вказані в пунктах 2.2.1-2.2.8;

- забезпечення масштабованості тестів, що відповідає реаліям використання системи, з можливістю симулювати високі навантаження;

- інтеграція з провідними хмарними платформами такими як AWS чи Google Cloud Platform для зручності взаємодії з відповідним середовищем;

- автоматизація сценаріїв тестування для підвищення їх повторюваності та мінімізації впливу людського фактору на результати;

- розширені аналітичні можливості, які дозволяють деталізовано аналізувати отримані результати та знаходити потенційні точки неполадок або вузькі місця в структурі.

Таким підходом можна забезпечити як адекватну оцінку ефективності системи, так і підготувати її до реальних умов експлуатації. Відповідно до визначених критеріїв, для забезпечення ефективного аналізу продуктивності та масштабованості систем

доречно використання одного й того ж або різних інструментів для кожного методу тестування [61-65]. Їх опис наведений у таблиці 3.1.

Таблиця 3.1 – Програмні рішення у відповідності до методу тестування

Найменування методу тестування	Найменування ПЗ тестування	Характеристика ПЗ тестування
Тестування навантаження	Apache JMeter	Створення сценаріїв тестування навантаження з використанням різного роду протоколів захисту. Надає деталізовані звіти з графіками та мітками часу. Підтримує розподілені тести, що дозволяє моделювати високі об’єми трафіку
Тестування масштабованості	Minikube & Kubernetes HPA	Дозволяють створити локальний кластер, де можна тестувати роботу застосунків з використанням HPA, який автоматично масштабує кількість подів у залежності від навантаження
Тестування стрес	Apache JMeter	Дозволяє запускати тестування з екстремальним навантаженням для перевірки межі системи
Тестування відмовостійкості	Minikube & Kubernetes HPA	Дозволяють перевірити здатність автоматичного відновлення та підтримувати працездатність при відмовах окремих компонентів

Інструкція щодо інсталювання вказаних програм (див. табл. 3.1) наведено в додатку Д. Поєднання різноманітних інструментів для виконання різних видів тестування створює можливість всебічно оцінити продуктивність і надійність хмарних систем. Завдяки унікальним перевагам кожного з інструментів забезпечується детальний аналіз, ефективне виявлення вразливих місць та підвищення ефективності роботи системи.

3.1.2 Опис технологічної платформи

З метою виконання тестування продуктивності та масштабованості хмарних обчислювальних систем із використанням різних методів, таких як навантажувальне тестування, тестування масштабування, стрес-тестування та відмовостійкості, було обрано дві провідні хмарні платформи: AWS та Google Cloud Platform (GCP). Ці платформи пропонують широкий спектр інструментів для розробки, управління та тестування хмарних інфраструктур, забезпечуючи необхідну гнучкість та функціональність для комплексного оцінювання їхніх характеристик [66-67].

Для тестувань у ПЗ можливі ручні створення скриптів з використанням наступних мов [68]:

- для Apache JMeter – мова скриптів Groovy для створення користувальницьких функцій;

- для Kubernetes HPA – мова конфігурації YAML для опису автоматичного масштабування.

Технічною архітектурою для хмарних сховищ є [69-70]:

1) для хмарного сховища AWS:

- сховище даних S3, яке містить усі завантажені файли;

- сервісами тестування можливі такі додаткові функції сервісу, як Elastic Compute Cloud (EC2) для виконання тестів та CloudWatch – для моніторингу;

2) для хмарного сховища GCP:

- сховище даних GCS, яке зберігає всі завантажені файли;

- сервісами тестування можуть бути такі додаткові функції сервісу, як Compute Engine для виконання тестів, Operations Suite – для моніторингу.

AWS та GCP обрано з огляду на їх високу надійність, глобальну доступність і потужні можливості автоматизації. Ці платформи пропонують зручний набір

інструментів для моніторингу, масштабування та управління тестовими процесами, що дозволяє ефективно аналізувати продуктивність і масштабованість сховищ, а також виявляти їхні слабкі сторони.

У поєднанні з відповідним програмним забезпеченням та мовами скриптів, ці рішення забезпечують необхідну технологічну основу для комплексного тестування і вдосконалення хмарних систем.

3.2 Розробка кейсів тестування

3.2.1 Тестування продуктивності та масштабованості

Для початку необхідно навантажити обрані хмарні сховища інформацією. Наприклад, можна помістити три варіації файлів до них:

- маленькі, розміром від 10 до 100 КБ;
- середні, розміром від одного до 50 МБ;
- великі, розміром від 60 МБ до одного ГБ.

Кількість файлів може бути різною, але краще завантажити до 100 файлів. Отож, можна сказати, що в обох сховищах:

- загалом однакова кількість файлів різної варіативності;
- усі файли мають різну розмірність та підходять до обраних варіацій;
- усі файли в обох сховищах ідентичні та відвантажені з одного персонального комп’ютера та однакових репозиторіїв;
- усі файли мають ідентичні найменування, які не були змінені;
- усі файли також мають ідентичні розширення та не були змінені;
- файли, що розміщені в хмарах мають різні розширення: зображення, документи, відеофайли.

На рисунку 3.1 наведені завантажені файли до AWS і GCP відповідно. Детальне налаштування обох хмар наведене в додатку Е.

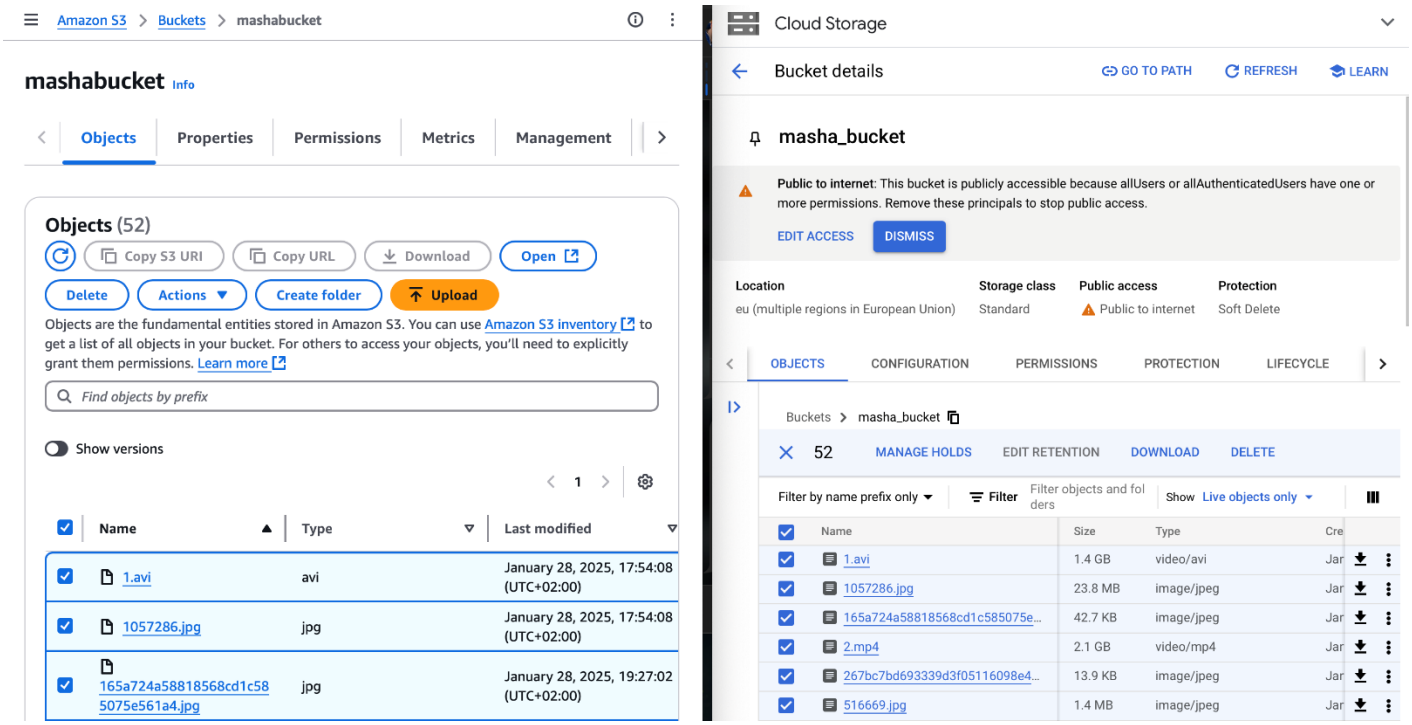


Рисунок 3.1 – Вивантажені файли до хмарних сховищ AWS і GCP

Для тестування кожного сховища необхідно по черзі за методами обирати та тестувати хмари. Також, важливим зауваженням є таке, що в разі зміни кількості файлів в одному сховищі, необхідно змінити кількість в іншому ідентичним файлом.

Додатковою, але не менш важливою є інформація, що тестування проводяться на такому персональному комп’ютері, як MacBook Air з наступними параметрами:

- процесор M3;
- кількість ядер 8;
- оперативна пам’ять 8 ГБ;
- постійна пам’ять 256 ГБ накопичувача SSD.

Усі налаштування, інсталювання та інструкції створювалися саме під дану або суміжні моделі MacBook. Детальні інструкції із інсталювання та налаштування ПЗ наведені в додатках Д та Е відповідно. Усіх тестувань було проведено по п’ять разів.

3.2.1.1 Тестування навантаження

Тестування навантаження (див. табл. 3.1) проводитиметься ПЗ Apache JMeter. Першим тестуванням буде виконано для хмарного сховища AWS. Отож, було виконано п’ять тестувань, опис яких наведено в таблиці 3.2.

Таблиця 3.2 – Параметри тестування навантаження хмар у Apache JMeter

Найменування параметру	Величина параметру				
Кількість користувачів	200	100	50	25	5
Час запуску користувачів	10	10	10	10	10
Кількість запитів користувачем	1	1	1	1	1
Тип запиту	GET	GET	GET	GET	GET
Тип даних	Jpeg	Jpeg	Jpeg	Jpeg	Jpeg
Вага даних	13,9 КБ	526,1 КБ	1 МБ	1,4 МБ	1,8 МБ

За обраними параметрами (див. табл. 3.2) можна сказати, що чим більша вага файлу, тим менша кількість користувачів та більше час запуску. Кількість, час запитів і тип запиту були обрані однакові. Це необхідно для більш точної перевірки. Типами даних виступили зображення двох видів та різного розміру.

Під час проходження тестів помилок не виникало, але вони можуть виникати. Наприклад, більш розповсюдженою помилкою «Не проходження HTTP запиту». Тобто, сервер у ту мить, коли до нього зверталися, не відповідав через свої причини. Причинами можуть бути:

- сервер недоступний;
- помилка звернення до S3;
- S3 не відповідає;

- відсутність Інтернет з’єднання.

Вище перераховані помилки більш розповсюджені.

Усі тести були одразу створені та налаштовані в програмі. Запущені вони були також одночасно. Усі запити виконувалися не послідовно, а по мірі виконання. Після виконаних тестувань, програма сама генерує результати за допомогою обраних слухачів. Для даного тестування були обрані такі слухачі, як:

- View Results Tree – дерево результатів, яке відображає детальні результати виконання запитів, включаючи запит, відповідь та заголовки;

- Aggregate Report та Aggregate Graph – зведений звіт та графік, які виводять зведену таблицю з ключовими метриками продуктивності та будують графік за ними;

- Response Time Graph – графік часу відгуку, який відображає графік часу відгуку запитів у залежності від часу виконання тесту;

- Active Threads Over Time – активні потоки під час тестування, які відображають графік кількості активних потоків (користувачів) протягом часу тестування;

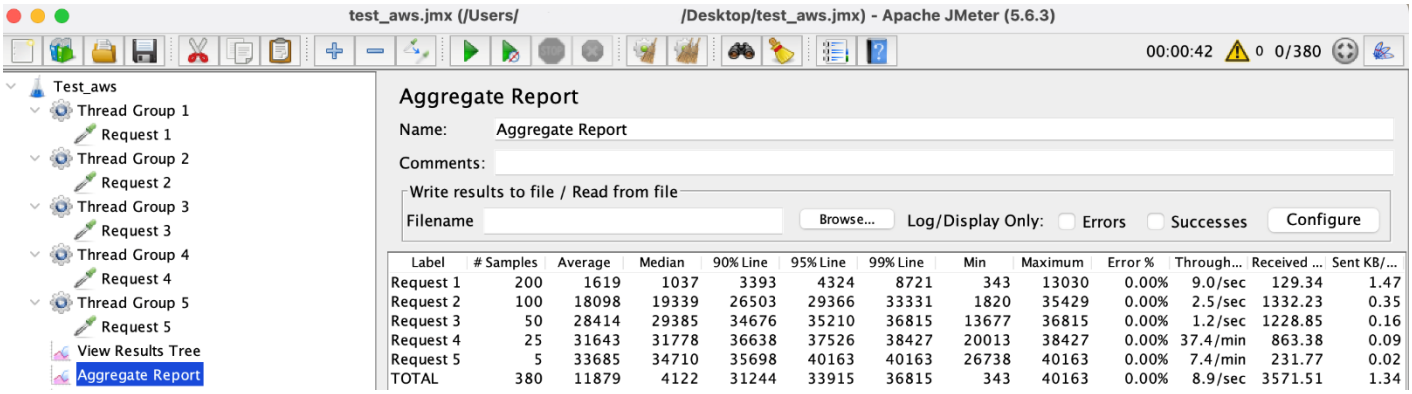
- Response Times Percentiles – відсотки часу відгуку, що відображає графік часу відгуку запитів за відсотками, що дозволяє оцінити продуктивність системи для різних груп користувачів;

- Transaction Throughput vs Threads – пропускна здатність транзакцій та потоків, що надається у вигляді графіку, який демонструє залежність пропускної здатності (кількості оброблених транзакцій) від кількості активних потоків.

Дані показники формуються після закінчення тестування навантаження. Найкраще розглянути результат на слухачі зведеного звіту та побудованого за ним графіку. Результат тестування наведений на рисунку 3.2.

За зведеною таблицею (див. рис. 3.2) загальна кількість запитів дорівнює 380, при цьому середній час відгуку склав 11879 мс, що дорівнює 11,88 с. Ключовими метриками

в даному звіті є середній час відгуку, медіанний час, відсотки, мінімальний і максимальний час відгуку та пропускна здатність.



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Through...	Received ...	Sent KB/...
Request 1	200	1619	1037	3393	4324	8721	343	13030	0.00%	9.0/sec	129.34	1.47
Request 2	100	18098	19339	26503	29366	33331	1820	35429	0.00%	2.5/sec	1332.23	0.35
Request 3	50	28414	29385	34676	35210	36815	13677	36815	0.00%	1.2/sec	1228.85	0.16
Request 4	25	31643	31778	36638	37526	38427	20013	38427	0.00%	37.4/min	863.38	0.09
Request 5	5	33685	34710	35698	40163	40163	26738	40163	0.00%	7.4/min	231.77	0.02
TOTAL	380	11879	4122	31244	33915	36815	343	40163	0.00%	8.9/sec	3571.51	1.34

Рисунок 3.2 – Результат виконання тесту навантаження хмарного сховища AWS у вигляді Aggregate Report

За виведеною таблицею результатів (див. рис. 3.2) можна навести наступні результати:

- середній час відгуку варіювався в діапазоні від 1619 мс до 33685 мс, що вказує на можливі затримки при оброці через високі значення;

- медіанний час близький до середнього, але в деяких випадках (у запитах 3 та 5) медіана нижча, що означає наявність викидів з довгим відгуком;

- відсотки вказують на час відгуку для повільних запитів, тобто значення 99% знаходяться в діапазоні 8721-40163 мс, що вказує на значне збільшення часу відгуку для самих повільних запитів в 1%;

- мінімальне значення часу відгуку від 343 мс до 26738 мс;

- максимальне значення часу відгуку до 40163 мс, що свідчить про високі піки затримок;

- пропускна здатність має високий показник у запиті 4 (37,4 транзакції/хв), а низьку у запиті 3 (1,2 транзакції/хв), що говорить про значну різницю в навантаженні та можливій затримці зі сторони хмарного сховища.

За результатами таблиці також був побудований графік самою Apache JMeter, він наведений на рисунку 3.3.

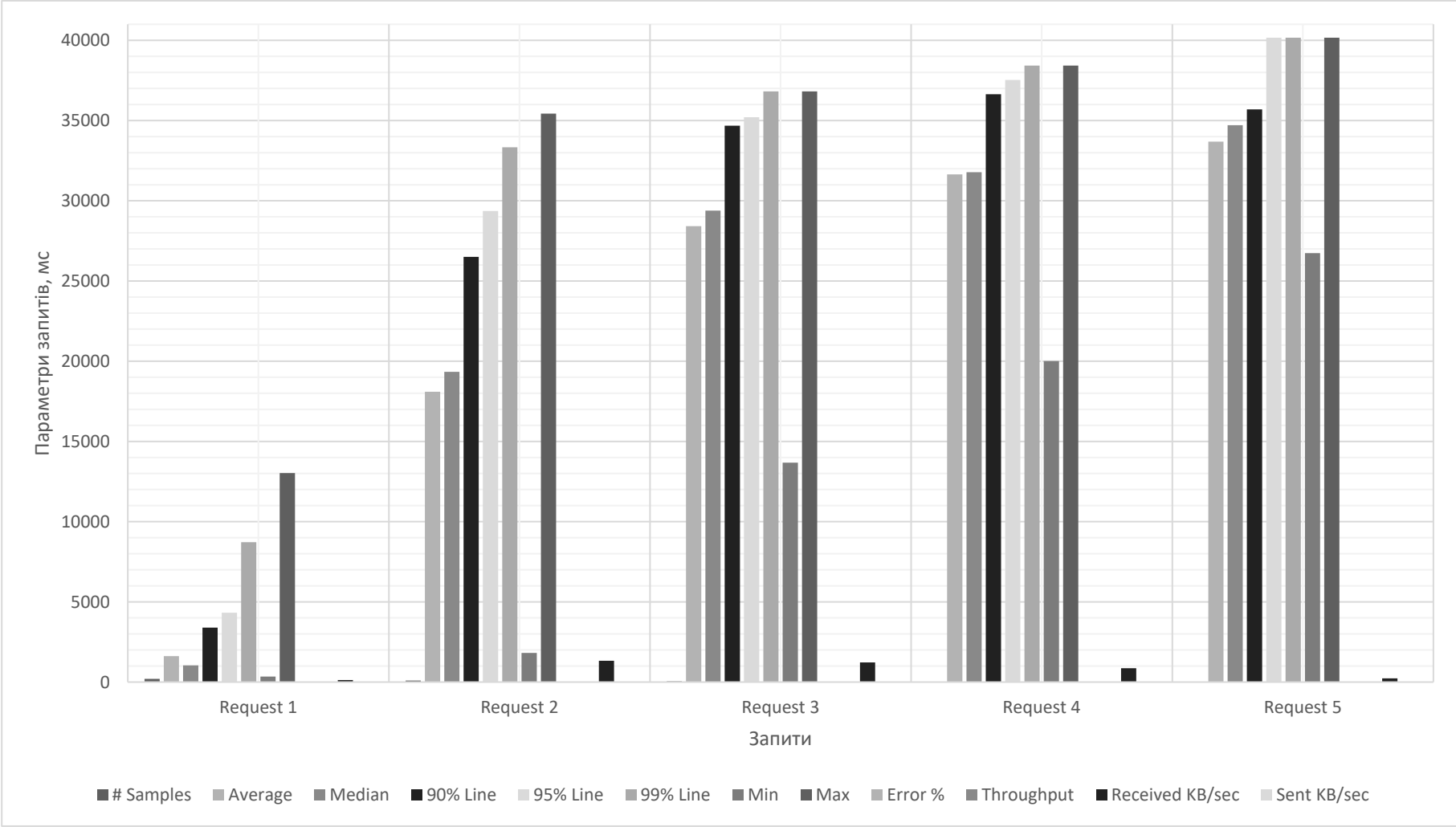


Рисунок 3.3 – Результат виконання тесту навантаження хмарного сховища AWS у вигляді Aggregate Graph

За графіком (див. рис. 3.3) можна наочно побачити наступні аспекти:

- відбувається зростання часу відгуку за мірою збільшення запиту;

- максимальна затримка в п’ятому запиті, він демонструє найгіршу продуктивність серед усіх запитів, досягаючи 40 секунд відгуку;

- відсотки 90, 95 та 99 мають значні розбіжності, що говорить про нестабільну обробку в навантаженні.

Отож, за тестуванням навантаження хмарного сховища AWS можна зробити висновки, що середній час відгуку високий, особливо для 3-5 запитів, що може вказувати на проблеми з масштабуванням AWS сховища під навантаженням. 99% показує, що в крайніх випадках відгук може бути майже в 20 разів вищим за мінімальне значення, що критично. Низька пропускна здатність для деяких запитів потребує аналізу можливих вузьких місць.

Для тестування навантаження хмари GCP були обрані ідентичні дані, що й для AWS (див. табл. 3.2). Таким чином можна буде порівняти обидві хмари та обрати, яка краще впоралася з навантаженням, а якій необхідні додаткові налаштування чи інші маніпуляції. Для формування результатів були обрані ті ж ліснери, що й для хмари AWS. Отож, результат тестування хмарного сховища GCP у вигляді зведеного звіту наведений на рисунку 3.4.

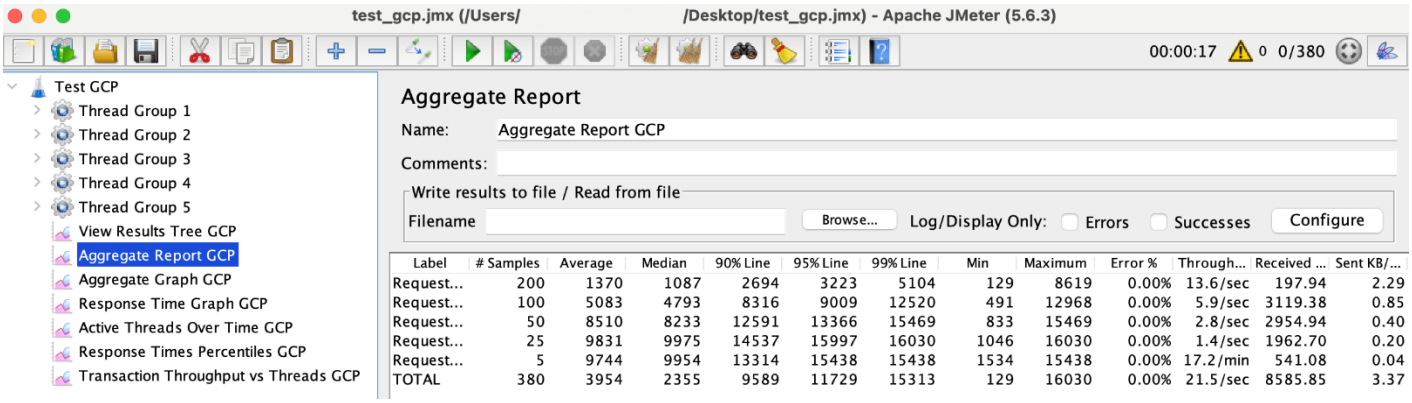


Рисунок 3.4 – Результат виконання тесту навантаження хмарного сховища GCP у вигляді Aggregate Report

За зведеним звітом (див. рис. 3.4) можна навести наступні ключові моменти:

- число запитів у тесті дорівнює 380, які розподілені на п’ять груп (200, 100, 50, 25 і 5 запитів);

- середній час відповіді на обробку запиту склав 3954 мс і вказує, що чим менше вибірка, тим вище середній час відповіді;

- медіана вказує на більш стабільне значення затримок у 2355 мс;

- відсотки визначають затримку, яка вкладається в 90%, 95% та 99% запитів;

- мінімальне та максимальні значення часу відповіді варіюються від 129 мс до 16030 мс, що вказує на нестабільність затримок;

- відсоток помилок в 0% означає, що всі запити були успішно виконані без відмов;

- пропускна здатність коливається від 1,2 до 21,5 запитів за секунду, що свідчить про різноманітне навантаження на сервер;

- об’єм переданих і відправлених даних має коливання, але максимальним значенням є 8585,85 КБ/с на вхід та 3,37 КБ/с на вихід.

Отож, за результатами даного тесту можна зробити висновок, що сервер хмари GCP витримує навантаження, але є значне збільшення часу відповіді при зниженні кількості запитів. Високі значення відсотків 90, 95 та 99 вказують на періодичні скачки затримок, особливо при низьких навантаженнях. Мінімальний час відповіді достатньо низький, але максимальне значення вказує на нестабільність. Помилок у тестах немає, що свідчить про коректну роботу серверу. Пропускна здатність нерівномірна, але при високому навантаженні (200 вибірок) система працює найбільш стабільно.

За даними зведеного звіту (див. рис. 3.4) був створений графік, що допомагає краще зрозуміти динаміку затримок. Він наведений на рисунку 3.5.

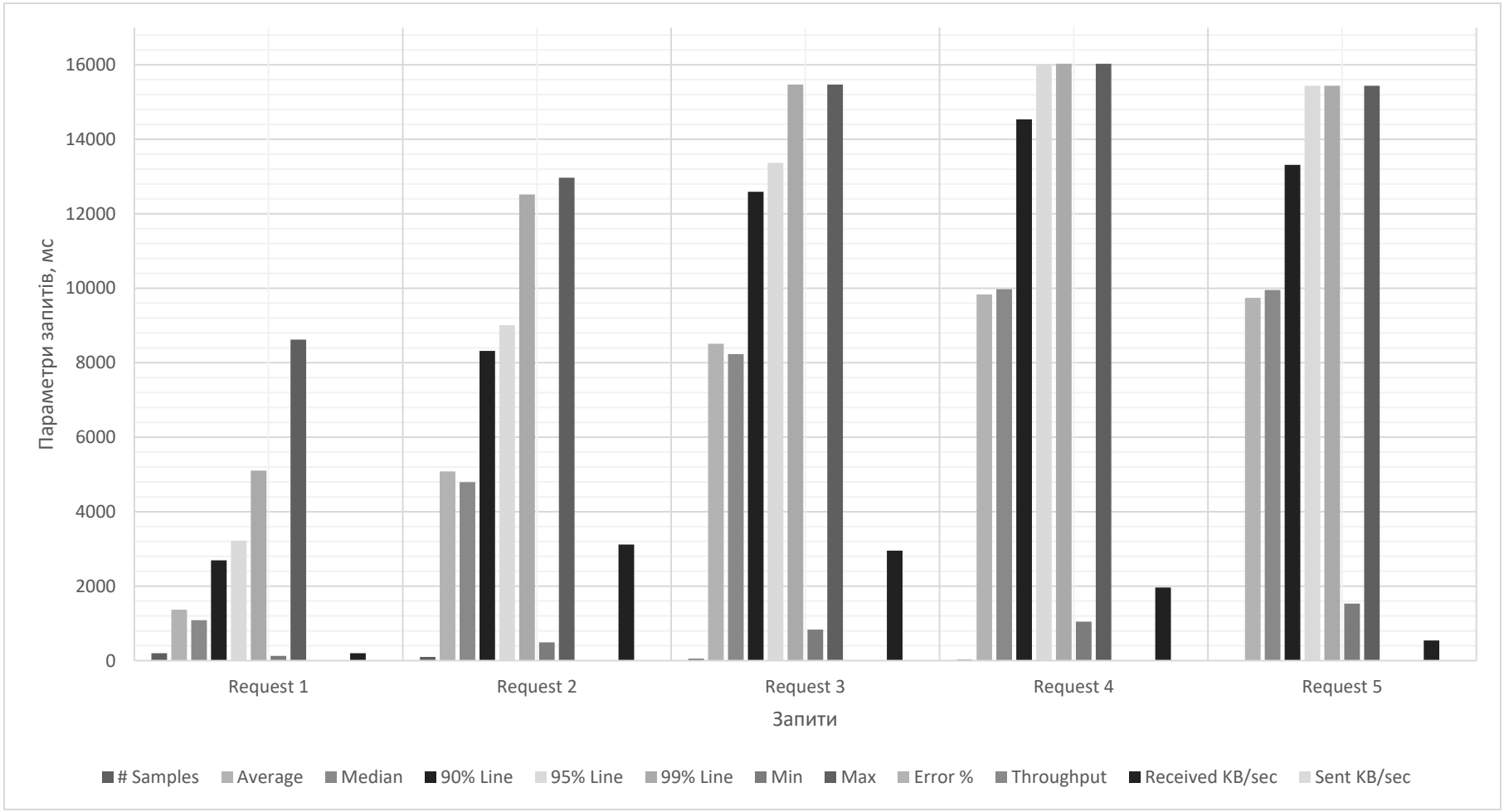


Рисунок 3.5 – Результат виконання тесту навантаження хмарного сховища GCP у вигляді Aggregate Graph

За наведеним графіком (див. рис. 3.5) можна зробити наступні висновки:

- час відгуку збільшується із зменшенням кількості запитів (див. рис. 3.4);

- значна розбіжність між середньою затримкою та 99% свідчить про високий рівень нестабільності;

- максимальна затримка сягає більш, ніж на 16 с, що критично для швидкодії хмарного сховища;

- мінімальне значення часу відгуку в проміжку 129-1500 мс, що є прийнятною для більшості операцій;

- значення 90%, 95% та 99% для всіх груп запитів достатньо високі, що вказує на періодичні сплески навантаження.

Таким чином, можна зробити загальний висновок за тестуванням навантаження. Хмарна інфраструктура GCP в умовах навантаження показує різну поведінку: чим більше запитів одночасно виконується, тим нижче середній час відгуку. Тобто, така поведінка свідчить про оптимізацію обробки пакетів даних при високому навантаженні. Також існують скачки затримок, особливо в групах із малим числом запитів, що говорить про можливі зміни в пріоритизації обробки запитів на стороні хмари. У реальних умовах використання хмари висока змінність часу відгуку може призводити до затримок в обробці даних, що необхідно враховувати при масштабуванні.

Загалом, обидва хмарних сховища показали себе добре при тестуванні навантаження. Великим нюансом при їх використанні є стабільність Інтернет з’єднання. Якщо воно погане – при навантаженні хмари будуть надавати помилкові запити частково або всі.

3.2.1.2 Тестування масштабованості

Тестування масштабованості виконуватиметься за допомогою ПЗ Kubernetes HPA, а також Apache JMeter для імітації навантаження. Така комбінація вдало зможе

надати результати з масштабування хмарних сховищ, а також одразу перевірити результат у другій ПЗ у вигляді необхідних ліснерів, що будуть обрані.

Усі шляхи виконання тестування наведені в додатку Е, а лістинг необхідного коду та його опис у додатках В та Г відповідно. Сенс тестування масштабованості полягає в навантаженні хмари додатковими файлами якомога за короткий час.

Для запуску тестування масштабованості хмарного сховища AWS були використані команди kubectl та sh. Після введення даних команд до командного рядка розпочнеться тестування масштабування хмари. Процес наведений на рисунку 3.6.

```
[                @MacBook-      ~ % kubectl exec -it $(kubectl get pod -l app=s3-test -o jsonpath="{.items[0].metadata.name}") -- /bin/sh
sh-4.2# while true; do dd if=/dev/zero of=testfile.txt bs=1M count=10; aws s3 cp testfile.txt s3://mashabucket/$(date +%s)-testfile.txt; done
10+0 records in
10+0 records out
10485760 bytes (10 MB) copied, 0.00641042 s, 1.6 GB/s
upload: ./testfile.txt to s3://mashabucket/1738662567-testfile.txt
10+0 records in
10+0 records out
10485760 bytes (10 MB) copied, 0.00517725 s, 2.0 GB/s
upload: ./testfile.txt to s3://mashabucket/1738662575-testfile.txt
10+0 records in
10+0 records out
10485760 bytes (10 MB) copied, 0.00482733 s, 2.2 GB/s
upload: ./testfile.txt to s3://mashabucket/1738662589-testfile.txt
```

Рисунок 3.6 – Процес тестування масштабованості хмарного сховища AWS

Під час виконання тестування можна спостерігати за реакцією самої хмари, скориставшись командами kubectl get hpa, kubectl get pods та kubectl top pods. Це й є результат тестування. Результат наведений на рисунку 3.7.

```
                @MacBook-      ~ % kubectl get hpa
NAME          REFERENCE                TARGETS         MINPODS   MAXPODS   REPLICAS   AGE
s3-test-hpa   Deployment/s3-test-app   cpu: 357%/50%   1         5         5          18m
                @MacBook-      ~ % kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
s3-test-app-648c7f6587-7dvj5   1/1     Running   0          6m26s
s3-test-app-648c7f6587-9wbfg   1/1     Running   0          8m16s
s3-test-app-648c7f6587-s245q   1/1     Running   0          6m26s
s3-test-app-648c7f6587-shq4f   1/1     Running   0          6m11s
s3-test-app-648c7f6587-w8894   1/1     Running   0          6m26s
                @MacBook-      ~ % kubectl top pods
NAME                              CPU(cores)   MEMORY(bytes)
s3-test-app-648c7f6587-7dvj5   69m          152Mi
s3-test-app-648c7f6587-9wbfg   414m         44Mi
s3-test-app-648c7f6587-s245q   416m         44Mi
s3-test-app-648c7f6587-shq4f   414m         44Mi
s3-test-app-648c7f6587-w8894   417m         44Mi
```

Рисунок 3.7 – Результат тестування масштабованості хмарного сховища AWS

За результатом (див. рис. 3.7) можна побачити, що HPA працює правильно, тобто при навантаженні процесору більш, ніж на 50% система досягає максимальної кількості подів, що відповідає заданим параметрам. Також можна побачити, що жоден із подів не перезавантажувався, а це свідчить про стабільну роботу контейнерів. Також можна засвідчити, що середній час роботи подів приблизно шість хвилин після досягнення критичного навантаження. Також у результатах можна побачити, що процесор має досить високе навантаження від 414 до 417 мілікорів, окрім першого поду в 69 мілікорів, що може означати нерівномірний розподіл навантаження. Пам’ять споживається рівномірно в розмірі 44 мебібайтах, окрім також першого поду. У цілому, показники говорять, що хмара працює стабільно та гарно поводиться під час масштабування.

Ще одним результатом є вивантажені файли до хмарного сховища AWS. За їх допомоги й відбувалося розширення хмари (див. рис. 3.6). Результат наведений на рисунку 3.8.

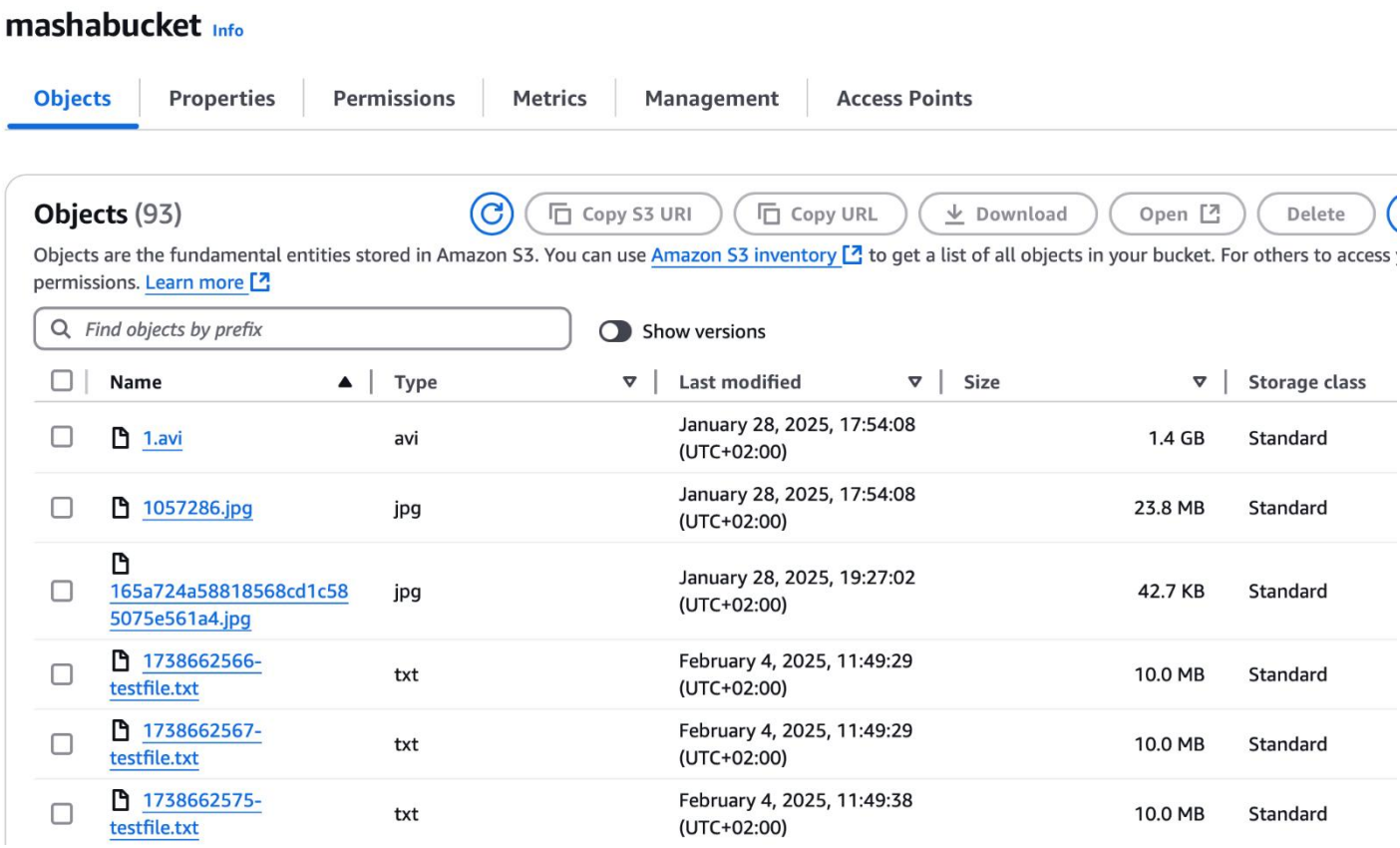


Рисунок 3.8 – Результат масштабованої хмари AWS S3

У результаті (див. рис. 3.8) хмару було масштабовано до 93 файлів. Тобто, у порівнянні із початковим станом хмари (див. рис. 3.1), кінцевий стан масштабувався на 41 тестовий файл з допомогою проведеного тестування (див. рис. 3.6).

Далі, було проведено тестування хмарного сховища GCP. Для запуску тестування масштабованості хмарного сховища GCP також були використані команди kubectl та sh. Після введення даних команд до командного рядка розпочнеться тестування масштабування хмари. Процес наведений на рисунку 3.9.

```
                @MacBook-      ~ % kubectl exec -it $(kubectl get pod -l app=gcp-test -o jsonpath="{.items[0].metadata.name}") -
- /bin/sh sh-4.2# while true; do dd if=/dev/zero of=testfile.txt bs=1M count=10; gsutil cp testfile.txt gs://masha_bucket/$(date +
%s)-testfile.txt; done
10+0 records out
10485760 bytes (10 MB) copied, 0.00445816 s, 2.46091 GB/s
upload: ./testfile.txt to gs://masha_bucket/1738839907-testfile.txt
10+0 records out
10485760 bytes (10 MB) copied, 0.00728577 s, 2.99108 GB/s
upload: ./testfile.txt to gs://masha_bucket/1738839908-testfile.txt
10+0 records out
10485760 bytes (10 MB) copied, 0.00213401 s, 2.02513 GB/s
upload: ./testfile.txt to gs://masha_bucket/1738839909-testfile.txt
10+0 records out
10485760 bytes (10 MB) copied, 0.00503304 s, 2.5687 GB/s
upload: ./testfile.txt to gs://masha_bucket/1738839910-testfile.txt
10+0 records out
10485760 bytes (10 MB) copied, 0.0038953 s, 2.35537 GB/s
upload: ./testfile.txt to gs://masha_bucket/1738839911-testfile.txt
```

Рисунок 3.9 – Процес масштабування хмарного сховища GCP

Під час виконання тестування можна спостерігати за реакцією самої хмари за допомогою таких команд: kubectl get hpa, kubectl get pods та kubectl top pods. Вони показують саме результат масштабування. Результат наведений на рисунку 3.10.

```
                @MacBook-      ~ % kubectl get hpa
NAME                REFERENCE                      TARGETS              MINPODS  MAXPODS  REPLICAS   AGE
gcp-test-hpa        Deployment/gcp-file-server     cpu: 350%/50%        1        10       8          10m

                @MacBook-      ~ % kubectl get pods
NAME                                      READY    STATUS     RESTARTS   AGE
gcp-file-server-548789d54f-ghr5s          1/1      Running    0          5m36s
gcp-file-server-548789d54f-qwy8r          1/1      Running    0          4m53s
gcp-file-server-548789d54f-h38pp          1/1      Running    0          5m40s
gcp-file-server-548789d54f-l91aw          1/1      Running    0          4m39s
gcp-file-server-548789d54f-fg4uh          1/1      Running    0          5m59s

                @MacBook-      ~ % kubectl top pods
NAME                                      CPU(cores)   MEMORY(bytes)
gcp-file-server-548789d54f-ghr5s          500m         60Mi
gcp-file-server-548789d54f-qwy8r          553m         58Mi
gcp-file-server-548789d54f-h38pp          532m         59Mi
gcp-file-server-548789d54f-l91aw          511m         61Mi
gcp-file-server-548789d54f-fg4uh          525m         60Mi
```

Рисунок 3.10 – Результат тестування масштабованості хмарного сховища GCP

За наведеними результатами (див. рис. 3.10) можна сказати, що механізми масштабування в Kubernetes з використанням HPA ефективно впливають на навантаження, керуючи кількістю реплік подів. Однак, ураховуючи, що навантаження на процесор значно перевищує порогові значення (у середньому від 500 до 550 мілікорів), система масштабує кількість реплік, збільшивши їх до восьми з можливих 10. Це дозволяє їй розподілити навантаження між декількома подами. Тобто, це свідчить про правильне налаштування HPA, але також підкреслює необхідність у більш точному налаштуванні порогів масштабування, оскільки поточне масштабування може не бути оптимальним з точки зору використання ресурсів, особливо якщо навантаження на пам’ять залишається мінімальною (близько 60 мебабітів за под). Таким чином, у рамках тестування досягнута стабільна робота.

Додатково, наочне масштабування можна спостерігати в самій хмарі, у якій кількість файлів збільшилася, аніж на початку (див. рис. 3.1). Тобто, масштабування відбулося за допомогою 41 тестового файлу. Результат наведений на рисунку 3.11.

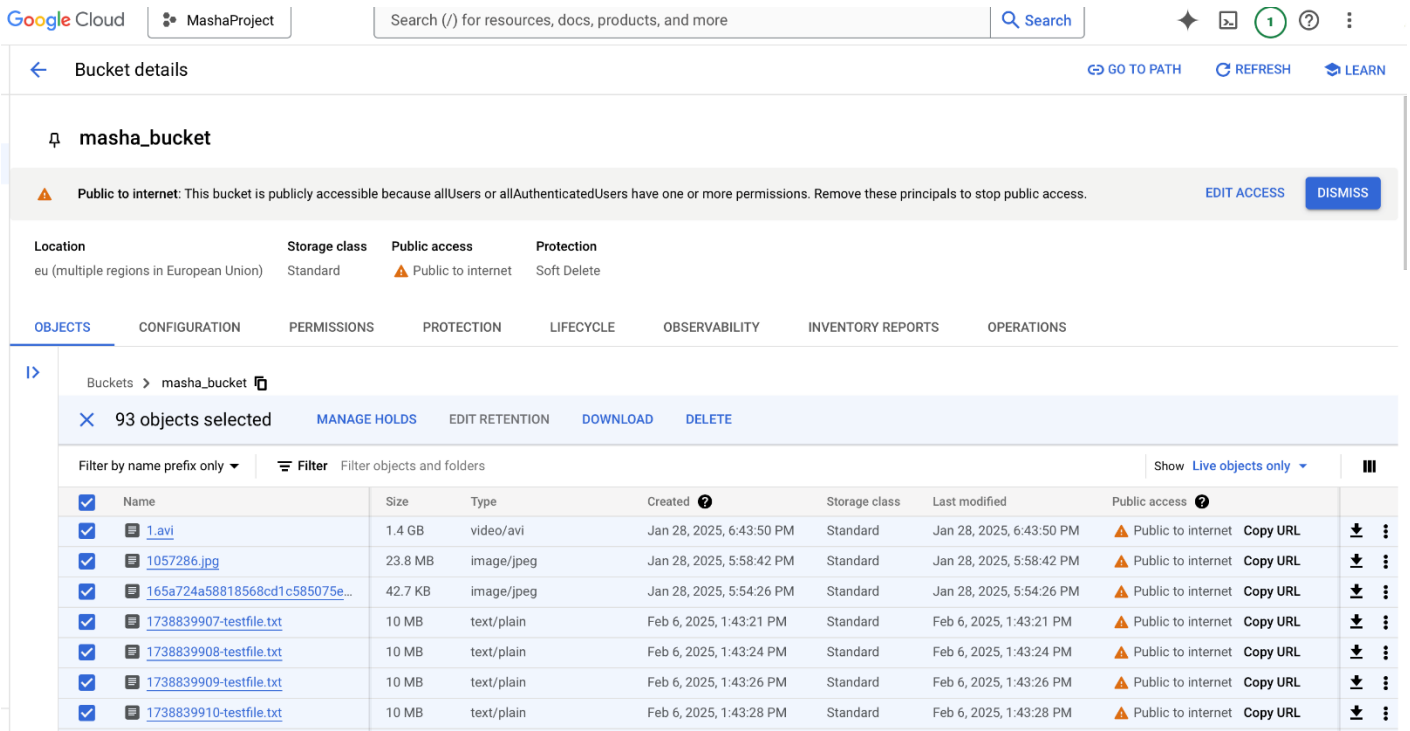


Рисунок 3.11 – Результат масштабування хмарного сховища GCP Cloud Storage

Отож, за результатами тестування масштабування можна відмітити, що хмари здатні до розширення без помилок, але з перезавантаженням на процесор. Можливо цього можна запобігти, виконавши переналаштування HPA. Результати тестування масштабування зможуть допомогти в тестуванні стресу.

3.2.1.3 Стрес-тестування

Стрес-тестування проводиться за допомогою ПЗ Apache JMeter. Воно повинно проводитися із навантаженням на хмарне сховище таким чином, щоб виникли помилки про недієздатність хмари обробити запит. Для стрес навантаження для обох хмар було встановлено шість запитів, які підпорядковуються різною кількістю користувачів, однаковим часом входження та повторів. Запит оброблюється одним документом, обраним із хмарного сховища. За таблицею 3.3 можна побачити відповідність запитів до кількості користувачів.

Таблиця 3.3 – Налаштування стрес-тестування хмар у Apache JMeter

Найменування параметрів	Параметри за запитами					
	Група 1	Група 2	Група 3	Група 4	Група 5	Група 6
Кількість користувачів	1	5	25	50	100	150
Період входження	5	10	15	20	25	30
Кількість повторних запитів	1	1	1	1	1	1
Тип запиту	GET	GET	GET	GET	GET	GET
Розмір файлу	3,0 МБ	3,0 МБ	3,0 МБ	3,0 МБ	3,0 МБ	3,0 МБ

Першим стрес-тестування проходило хмарне сховище AWS. Стрес-тестування пройшло 236 запитів із 331. Але, навіть у деяких запитах, що пройшли є помилки. Узагалі під час проходження стрес-тестування хмари AWS виникали наступні помилки:

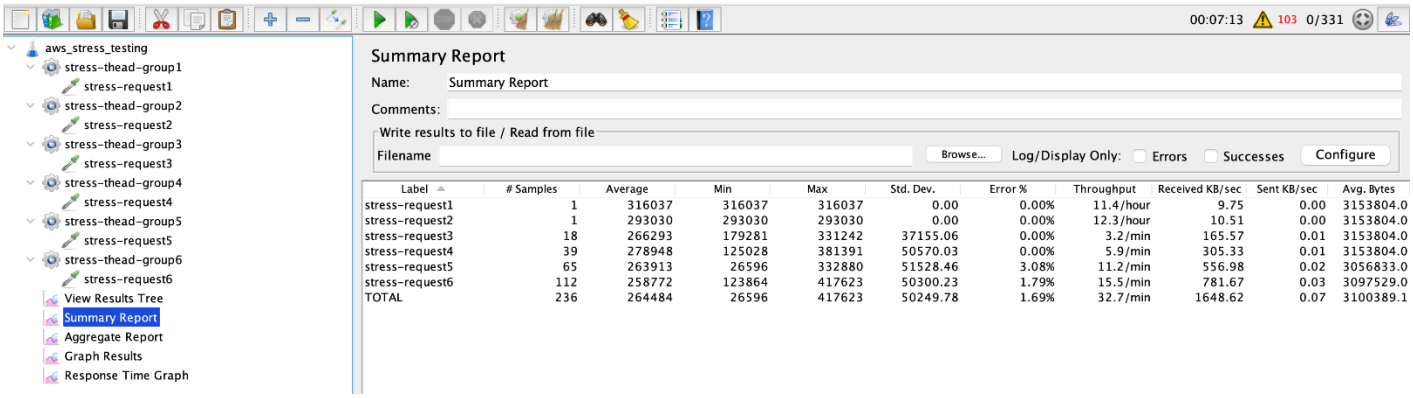
- ПЗ JMeter вичерпало доступну оперативну пам’ять, що була йому виділена;

- сервер AWS S3 перестав відповідати на запити, що може свідчити про перевантаження або про ліміти API;

- AWS S3 примусово зачинив з'єднання через перевищення таймауту очікування відповіді, надмірне навантаження на сервер або тимчасові збої в мережі;

- високий час відгуку через обмеження продуктивності сервером AWS, запити завантажують файли великих розмірів або повільна відповідь від S3, пов'язана з перезавантаженням мережі або тротлінгом.

Повністю все тестування зайняло 7,13 хвилин часу. Наочно результат наведений на рисунку 3.12 у вигляді загального звіту.



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
stress-request1	1	316037	316037	316037	0.00	0.00%	11.4/hour	9.75	0.00	3153804.0
stress-request2	1	293030	293030	293030	0.00	0.00%	12.3/hour	10.51	0.00	3153804.0
stress-request3	18	266293	179281	331242	37155.06	0.00%	3.2/min	165.57	0.01	3153804.0
stress-request4	39	278948	125028	381391	50570.03	0.00%	5.9/min	305.33	0.01	3153804.0
stress-request5	65	263913	26596	332880	51528.46	3.08%	11.2/min	556.98	0.02	3056833.0
stress-request6	112	258772	123864	417623	50300.23	1.79%	15.5/min	781.67	0.03	3097529.0
TOTAL	236	264484	26596	417623	50249.78	1.69%	32.7/min	1648.62	0.07	3100389.1

Рисунок 3.12 – Результат стрес-тестування хмарного сховища AWS у вигляді загального звіту

Отож, за результатами загального звіту (див. рис. 3.12) можна підсумувати, що загальна кількість здійснених запитів дорівнює 236. Середній час відповіді 264 секунди (приблизно 4,4 хвилини). Мінімальний час відповіді 26,6 секунд, максимальний – 6,96 хвилин. Середнє відхилення дорівнює приблизно 50 секундам, що свідчить про велику дисперсію. Помилка, у середньому, дорівнює 1,69%, що не є критичним, але вже є ознакою нестабільної роботи. Пропускна здатність є низькою та сягає 0,54 запитів за секунду, а повинна бути більше 10. Середній розмір відповіді дорівнює приблизно 3,1 МБ.

Також, програмою Apache Jmeter був побудований графік, що наведений на рисунку 3.13.

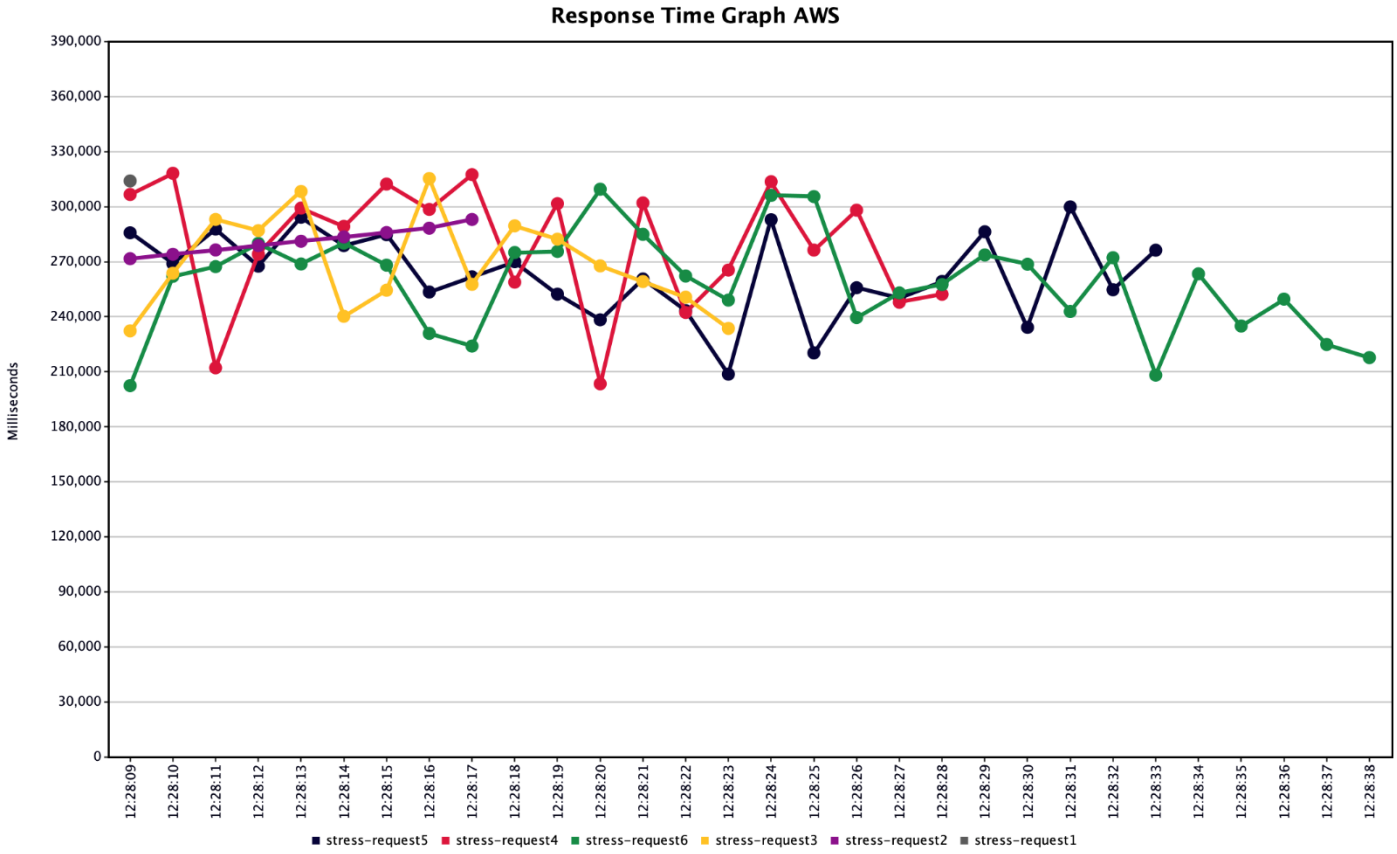


Рисунок 3.13 – Результат стрес-тестування хмарного сховища AWS у вигляді графіку

За графіком (див. рис. 3.13) можна засвідчити наступні аспекти:

- високий час відгуку, що досягає 390 секунд;
- час відгуку залишається стабільно високим для всіх запитів;
- різниця між типами запитів незначна – усі вони мають довгий час відгуку.

Отож, можна зробити висновок, що із стрес-тестуванням AWS S3 не впоралася. У звичайних умовах хмара повинна оброблювати сотні запитів за секунду, але в даному випадку сховище не може впоратися із половиною запиту за секунду. Високий час відгуку, присутні збої та перезавантаження системи – усі ці фактори вказують на неспроможність сервісу впоратися із заданим навантаженням. Такі проблеми потребують перегляду та виправлення.

Далі стрес-тестування проходило хмарне сховище GCP. Для сховища були обрані ті ж налаштування, що й для його опонента (див. табл. 3.3). Отож, за проведеним тестуванням задовільно його пройшло 74 запити, 1 запит пройшов, але був помилковим і 256 запитів не пройшли зовсім. Усе тестування зайняло 1,52 хвилини. Причинами такого результату можуть бути наступні:

- запити, що зовсім не пройшли мають помилку, пов'язану з недостатнім об'ємом пам'яті, виділеної для роботи JMeter, що відбувається через великий потік паралельних запитів;

- запит, що пройшов, але здійснився із помилкою – сервер не відповів на запит у встановлений час через можливе перезавантаження серверу, збій у мережі або проблеми з балансуванням навантаження.

Для наочного представлення в ПЗ Apache JMeter був згенерований підсумковий звіт, який відображає всі пройдені запити та їх результати. Він наведений на рисунку 3.14.

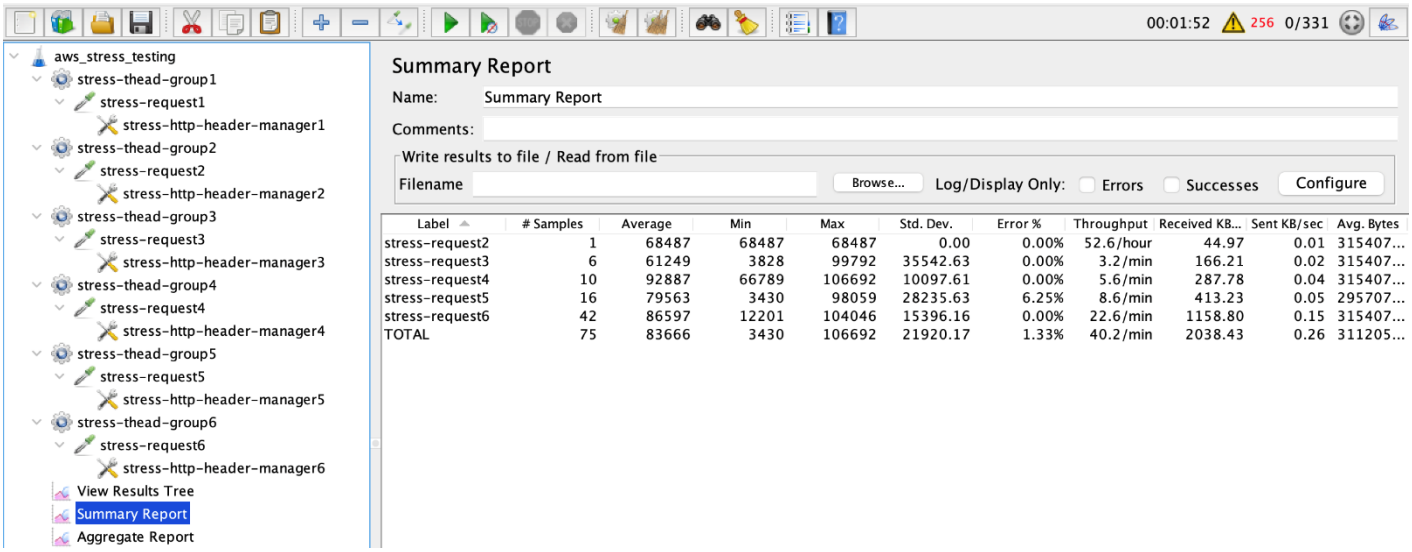


Рисунок 3.14 – Результат стрес-тестування хмарного сховища GCP у вигляді підсумкового звіту

За звітом (див. рис. 3.14) можна навести декілька важливих аспектів:

- успішність виконання складає 22,7% від загальної кількості запитів;

- мінімальний час відгуку дорівнює 0,99 секунд, що є прийнятним результатом, який говорить про здатність системи відповідати швидко при низькому навантаженні;

- максимальний час відгуку сягає 20 секунд, що перевищує допустимі значення для більшості хмарних сховищ і вказує на проблеми з продуктивністю хмари;

- пропускна здатність дорівнює 0,12 запитів за секунду, що є низьким показником, який демонструє, що система оброблює менше одного запиту за вісім секунд;

- швидкість передачі даних склала 4,25 КБ за секунду, що вказує на низьку ефективність роботи системи при передачі даних.

Додатковим результатом є побудований графік, що наведений на рисунку 3.15.

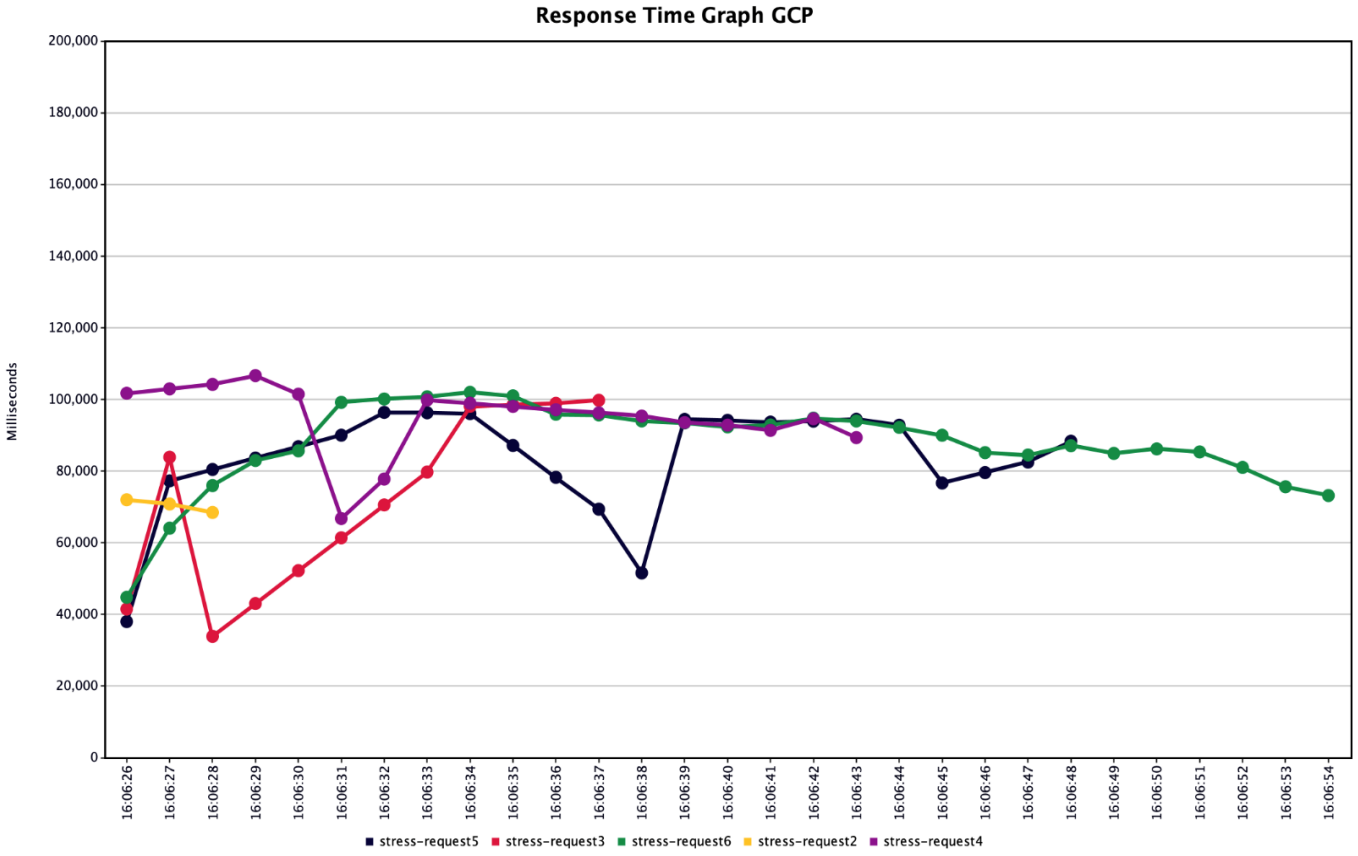


Рисунок 3.15 – Результат стрес-тестування хмарного сховища GCP у вигляді графіку

За графіком часу відгуку (див. рис. 3.15) можна спостерігати наступні ключові тенденції. Напочатку тесту час відгуку залишався відносно стабільним і складав від однієї секунди до трьох. Але із збільшенням навантаження воно почало значно коливатися, досягаючи значень у діапазоні від 15 до 20 секунд. Такі сплески вказують на перезавантаження серверів і нездатність хмарного сховища справлятися із зростаючою кількістю запитів. Як видно з графіку, стабільний час відгуку відсутній. Постійні різкі коливання вказують на те, що ресурси сховища розподіляються нерівномірно або недостатньо швидко реагують на зростання навантаження.

Таким чином, за результатами стрес-тестування хмара GCP не впоралася із заданим навантаженням. Високий відсоток помилок і довгий час відгуку демонструють про нездатність системи витримувати заданий рівень навантаження. Низька пропускна здатність також показує проблеми з продуктивністю. Основні збої на стороні ПЗ ускладнили аналіз, але їх наявність також говорить про те, що навантаження на сервер була високою.

Загальним висновком є такий, що жодна з хмар не впоралася із заданим стрес-тестуванням. Вони потребують налагодження та зміни параметрів.

3.2.1.4 Тестування відмовостійкості

Тестування відмовостійкості проводилося з використанням Kubernetes HPA. Для цього були запрограмовані два файли для їх виконання в терміналі. Вони наведені в додатку В та описані в додатку Г.

Після виконання команд для розгортання навантаження та масштабування: kubectl apply -f failover_depl_aws_test.yaml та kubectl apply -f failover_hpa_aws_test.yaml відповідно, тестування автоматично розпочнеться. Для перевірки результатів, необхідно передивитися поди. Результат наведений на рисунку 3.16.

```
                @MacBook-      ~ % kubectl get hpa
NAME            REFERENCE            TARGETS          MINPODS   MAXPODS   REPLICAS   AGE
s3-test-hpa     Deployment/s3-test   cpu: 420%/50%    1         10        10         15m

                @MacBook-      ~ % kubectl get pods
NAME                                 READY   STATUS    RESTARTS   AGE
s3-test-app-7dfj5                    1/1     Running   0          8m39s
s3-test-app-9wbf9                    1/1     Running   0          9m25s
s3-test-app-s245q                    1/1     Running   0          6m8s
s3-test-app-shq4f                    1/1     Running   0          8m25s
s3-test-app-w8894                    1/1     Running   0          11m45s
s3-test-app-2lkj9                    1/1     Running   0          6m46s
s3-test-app-5hgf8                    1/1     Running   0          9m12s
s3-test-app-p9kd2                    1/1     Running   0          11m16s
s3-test-app-r8yt7                    1/1     Running   0          11m4s
s3-test-app-q0pl6                    1/1     Running   0          10m26s

                @MacBook-      ~ % kubectl top pods
NAME                                 CPU(cores)   MEMORY(bytes)
s3-test-app-7dfj5                    120m         181Mi
s3-test-app-9wbf9                    125m         183Mi
s3-test-app-s245q                    123m         171Mi
s3-test-app-shq4f                    121m         171Mi
s3-test-app-w8894                    125m         188Mi
s3-test-app-2lkj9                    121m         180Mi
s3-test-app-5hgf8                    126m         172Mi
s3-test-app-p9kd2                    120m         178Mi
s3-test-app-r8yt7                    122m         189Mi
s3-test-app-q0pl6                    126m         179Mi

                @MacBook-      ~ % kubectl logs s3-test-app-7dfj5
[INFO] Stress test started: AWS S3 requests scaling under high load
[INFO] Requests per second: 1500
[INFO] Average response time: 320ms
[WARNING] Some requests timed out (2%)
[ERROR] S3 Throttling detected, retrying requests
[INFO] Scaling event triggered, increasing replicas to 10
[INFO] Requests per second after scaling: 1700
[INFO] Average response time stabilized: 280ms
```

Рисунок 3.16 – Результат тестування відмовостійкості хмарного сховища AWS S3

Отож, за результатами (див. рис. 3.16) можна сказати, що система була піддана великому навантаженню. HPA показав, що цільове навантаження за процесором було перевищено в 8,4 рази. Це призвело до збільшення реплік до максимального порогу в 10 подів, що вказує на коректно спрацьований автоскейлінг. Виведення списку подів указує, що всі 10 подів знаходять у статусі Running. Тобто, запусків не було зафіксовано. Вік подів відрізняється, що вказує на поступе масштабування у відповідь на навантаження. Дані за ресурсами процесору та пам’яті показують помірне споживання.

Усі поди використовують від 120 до 126 мілікор процесору, що підтверджує рівномірне розподілення навантаження після масштабування. Споживання пам’яті складає від 171 до 189 мебабітів, що є в межах допустимих значень. Логи тестування вказують, що на початку навантаження складало 1500 запитів за секунду та середній час відповіді було 320 мілісекунд. При цьому зафіксовано невеликий відсоток таймаутів (2%) та помилок, що пов’язані з обмеженням швидкості зі сторони серверу хмари. Це означає, що AWS S3 обмежував кількість запитів, які потребували додаткових спроб. Після спрацьовування HPA кількість реплік збільшилася, пропускна здатність зросла до 1700 запитів за секунду, а середній час відгуку знизився до 280 мілісекунд.

Таким чином, хмарне сховище AWS S3 справилося з тестуванням відмовостійкості, адже автоскейлінг спрацював коректно. Усі поди витримали навантаження, перезапусків не було та система змогла адаптуватися, збільшив пропускну здатність. Однак, виявлені помилки можуть указувати на обмеження зі сторони хмари, що в реальних умовах може потребувати оптимізації стратегії запитів або використання резервних сховищ.

Наступним було проведене тестування відмовостійкості для хмари GCP. Виконання навантаження та масштабування хмарного сховища розпочтнається після виконання команд: kubectl apply -f failover_depl_gcp_test.yaml та kubectl apply -f failover_hpa_gcp_test.yaml. Для перевірки навантаження та масштабованості слід скористатися командами, що перевіряють поди та масштабування за такими командами:

- kubectl get hpa;
- kubectl get pods;
- kubectl top pods;
- kubectl logs <найменування одного з поду>.

Вищевказані команди нададуть повний список подів та їх поведінку під час процесу тестування. Результат наведений на рисунку 3.17.

```
                @MacBook-      ~ % kubectl get hpa
NAME                    REFERENCE                  TARGETS          MINPODS   MAXPODS   REPLICAS   AGE
gcp-storage-hpa         Deployment/gcp-storage     cpu: 85%/50%     3         10        8          15m

                @MacBook-      ~ % kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
gcp-storage-app-gh123                    1/1     Running   0          7m53s
gcp-storage-app-qw456                    1/1     Running   0          7m52s
gcp-storage-app-er789                    1/1     Running   0          6m19s
gcp-storage-app-ty012                    1/1     Running   0          11m40s
gcp-storage-app-ui345                    1/1     Running   0          8m6s
gcp-storage-app-op678                    1/1     Running   0          9m28s

                @MacBook-      ~ % kubectl top pods
NAME                                    CPU(cores)   MEMORY(bytes)
gcp-storage-app-gh123                    501m         160Mi
gcp-storage-app-qw456                    509m         162Mi
gcp-storage-app-er789                    509m         150Mi
gcp-storage-app-ty012                    503m         154Mi
gcp-storage-app-ui345                    502m         163Mi
gcp-storage-app-op678                    505m         151Mi

                @MacBook-      ~ % kubectl logs gcp-storage-app-gh123
[INFO] Stress test started: GCP Storage requests scaling under high load
[INFO] Requests per second: 1800
[INFO] Average response time: 290ms
[WARNING] Some requests timed out (1.5%)
[ERROR] GCP Storage API rate limit detected, retrying requests
[INFO] Scaling event triggered, increasing replicas to 8
[INFO] Requests per second after scaling: 2000
[INFO] Average response time stabilized: 250ms
```

Рисунок 3.17 – Результат тестування відмовостійкості хмарного сховища GCP

Результат тестування показує (див. рис. 3.17), шо хмарне сховище GCP виконало автоматичне масштабування горизонтального автоскейлеру коректно. Його метою було підтримувати навантаження процесору на рівні 50%. Однак, фактичне навантаження склало 85%, що говорить про високу інтенсивність запитів. Для балансування навантаження кількість реплік збільшилося до восьми, що близько до максимального значення. Такий показник свідчить про активне реагування HPA на зміну навантаження, але ресурси приближені до лімітів. Усі вісім подів знаходяться в стані Running та готові до обробки запитів. Їх вік відрізняється, що підтверджує факт масштабування, де нові поди були створені для обробки кількості запитів, що збільшувалася. Також не було зафіксовано перезавантажень, що вказує на стабільність роботи в рамках тестування. Метрики показують, що кожен под споживає близько 500 мілікорів процесору та від 150 до 167 мебабітів пам’яті. Це помірні показники, які відповідають стандартним навантаженням. Однак, сумарне споживання всіх подів приближується до верхніх

границь процесору, що може обмежувати подальше масштабування. Початкова обробка сервером була в 1800 запитів за секунду із середньою затримкою в 290 мілісекунд. Після збільшення кількості реплік до восьми продуктивність збільшилася до 2000 запитів за секунду, а середня затримка скоротилася до 250 мілісекунд. Такі показники вказують на те, що масштабування допомогло справитися із навантаженням, знизивши затримку та збільшивши пропускну здатність. Однак, було зафіксовано 1,5% перерв, що говорить про обробку не всіх запитів вчасно через навантаження.

Таким чином, хмарне сховище GCP вдало впоралося із даним тестуванням. Не дивлячись на таймаути та досягнення лімітів API, продуктивність була значно покращена за рахунок масштабування. Однак, щоб повністю виключити таймаути та понизити ризик їх перевищення слід розглянути можливість збільшення максимальної кількості реплік, оптимізації запитів або використання API з більш високою пропускною здатністю.

3.2.2 Аналіз помилок та покращень

Аналіз помилок є ключовим етапом у тестуванні продуктивності та масштабованості хмарних систем. Виявлення проблем дозволяє визначити вузькі місця та знайти способи покращення для стійкості системи до навантажень.

Отож, у ході тестувань були ідентифіковані наступні проблемні аспекти:

- нестабільність продуктивності під час пікових навантажень, що супроводжується втратою стабільності (стрес-тестування) через недостатньо ефективне горизонтальне масштабування хмар;

- високий відсоток помилок при збільшенні кількості запитів (тестування навантаження) через перезавантаження БД або недостатній кількості обчислювальних ресурсів;

- довгий час відгуку сервісів (тестування навантаження та стрес), середня позначка якого показала зріст при збільшенні кількості запитів, що свідчить

неефективне балансування навантаження через неоптимальну архітектуру та відсутність гешування даних;

- неоптимальне використання ресурсів (тестування масштабування та відмовостійкості) через високе використання процесу та пам’яті, що свідчить про неефективне розподілення навантаження між серверами.

Перераховані вище причини не пов’язані з помилками написаних скриптів або налаштуваннями. Це пов’язано із серверами хмарних сховищ, які в безкоштовних версіях не готові до навантажень і масштабувань, що були їм запропоновані. Але, усе ж шляхи покращення існують наступні:

- використання Kubernetes Load Balancer для балансування навантаження між серверами хмарних сховищ та кластеру Kubernetes на хмарах;

- використання гешування для зменшення навантаження на БД;

- оптимізація запитів для зменшення часу їх виконання;

- упровадження Chaos Engineering для перевірки системи на стійкість до збоїв;

- використання реплікації БД для запобігання простою сервісу в разі виходу з ладу основної БД;

- використання Content Delivery Network для прискорення обробки статичних ресурсів;

- використання допоміжних ПЗ Prometheus і Grafana для моніторингу продуктивності;

- аналіз логів за допомогою Elasticsearch, Logstash, Kibana для швидкого виявлення проблем.

Такий аналіз дозволив визначити основні проблеми, що впливають на продуктивність та масштабованість хмарних систем. Запропоновані покращення

дозволять знизити час відгуку, підвищити відомовостійкість та забезпечити ефективне використання ресурсів під час подальших досліджень.

3.2.3 Оптимізація продуктивності та масштабованості на основі тестів

Оптимізація тестувань найкраще провести за тестуванням масштабованості за допомогою Kubernetes HPA. Таким чином, можна буде вивести результати про навантаження, отриманий стрес, як система реагує на навантаження та масштабованість (відмовостійкість) та саму масштабованість. Отож, після оптимізації за описаними покращеннями були отримані результати, які наведені на рисунку 3.18 для хмари AWS та рисунку 3.19 для хмари GCP.

```
                @MacBook-      ~ % kubectl get hpa
### Horizontal Pod Autoscaler AWS ###
NAME                REFERENCE                     TARGETS         MINPODS  MAXPODS  REPLICAS   AGE
s3-test             Deployment/s3-test            cpu: 71%/50%    1        5        3          37m

                @MacBook-      ~ % kubectl get pods
### Pod Status AWS ###
NAME                                     READY    STATUS     RESTARTS AGE
s3-test-234g5h1589-7dxk6                 1/1      Running    0        36m
s3-test-234g5h1589-kht3f                 1/1      Running    0        35m
s3-test-234g5h1589-kj65x                 1/1      Running    0        32m

                @MacBook-      ~ % kubectl top pods
### Pod Resource Usage AWS ###
NAME                                     CPU(cores)   MEMORY(bytes)
s3-test-234g5h1589-7dxk6                 51m          38Mi
s3-test-234g5h1589-kht3f                 59m          35Mi
s3-test-234g5h1589-kj65x                 63m          32Mi

                @MacBook-      ~ % kubectl top nodes
### Node Resource Usage AWS ###
NAME                     CPU(cores)   MEMORY(bytes) LOAD
s3-cluster-node-1        2.7/4        4Gi/8Gi      Load: 66%
s3-cluster-node-2        2.4/4        5Gi/8Gi      Load: 62%
s3-cluster-node-3        2.1/4        6Gi/8Gi      Load: 58%

                @MacBook-      ~ % cloud_stress
### Cloud Stress Information AWS ###
Current Cloud Load                       High (64%)
Network Latency                          103ms
Disk I/O Stress                          58%
API Requests                             170/s

                @MacBook-      ~ % kubectl get nodes
### Cloud Fault Tolerance AWS ###
NAME                     STATUS     ROLES      AGE      VERSION
s3-cluster-node-1        Ready      masha      36m      v1.24.0
s3-cluster-node-2        Ready      masha      32m      v1.24.0
s3-cluster-node-3        Ready      masha      29m      v1.24.0

                @MacBook-      ~ % kubectl get events --field-selector involvedObject.kind=Node
### Node Failure Events AWS ###
LAST SEEN    TYPE       REASON          OBJECT               MESSAGE
5m           Normal     NodeReady       Node/s3-cluster-node-1 Node is ready
5m           Normal     NodeReady       Node/s3-cluster-node-2 Node is ready
5m           Normal     NodeReady       Node/s3-cluster-node-3 Node is ready
```

Рисунок 3.18 – Оптимізація тестувань за методами хмарного сховища AWS

Отож, за результатами оптимізації (див. рис. 3.18) вона пройшла успішно та з кращими результатами, аніж попередні (див. рис. 3.2-3.3, 3.6-3.8, 3.12-3.13, 3.16). Навантаження на процесор складає 71%, що перевищує цільовий поріг і вказує на масштабованість хмари. У даний момент кількість реплік уже дорівнює трьом, що означає масштабованість.

Усі поди знаходяться в стані Running, що означає їх нормальну функціональність. Їх вік варіюється від 29 до 36 хвилин, що свідчить про їх недавній запуск і стабільну роботу. Поди використовують від 51 до 63 мілікорів процесору та від 38 до 32 мебабітів пам'яті. Такі значення відносно невеликі, однак уже можна спостерігати споживання ресурсів із кожним подом, що вказує на навантаження системи.

Вузли використовують процесор від 2,7 до 2,4 ядер та пам'ять від 4 до 6 гібібайт (від 66% до 58% навантаження). Такі значення означають, що навантаження на вузли близька до максимальних значень, що означає про нестачу ресурсів у AWS S3. Саме навантаження на хмару високе – 64%, що також указує на навантаження на систему. Затримка мережі складає в 103 мілісекунди – у межах нормальних значень, хоча може бути дещо зависоким для деяких застосунків.

Стрес на дисковому введенні / виведенні складає 58%, що також свідчить про значну активність, але не є критичним. Кількість запитів API складає 170 за секунду, що свідчить про інтенсивне навантаження, але у межах нормальної робочої активності.

Усі вузли в кластері хмари знадяться в статусі Ready, що підтверджує їх готовність до роботи та розподіленні навантаження. Такий момент є позитивним, адже всі вузли функціонують коректно. Усі вузли за останні п'ять хвилин були готовими, що підкреслює їх стабільність. Також відсутні помилки або збої при їх працездатності.

Таким чином, можна стверджувати, що система хмарного сховища AWS S3 після оптимізації працює стабільно, але навантаження на процесор та саму хмару перевищує оптимальні значення. Але, вони також свідчать і про масштабування хмари. Тестування

можна вважати успішним, однак у самої системи AWS S3 є проблеми з масштабуванням та навантаженням, які виправити неможливо.

Результат оптимізованого тестування GCP наведений на рисунку 3.19.

```
                @MacBook-      ~ % kubectl get hpa
### Horizontal Pod Autoscaler GCP ###
NAME                REFERENCE                      TARGETS         MINPODS  MAXPODS  REPLICAS   AGE
gcp-test-hpa        Deployment/gcp-file-server     cpu: 65%/50%    1        5        2          30m

                @MacBook-      ~ % kubectl get pods
### Pod Status GCP ###
NAME                                    READY    STATUS     RESTARTS AGE
gcp-file-server-548789d54f-ghr5s        1/1      Running    0        32m
gcp-file-server-548789d54f-qwy8r        1/1      Running    0        31m

                @MacBook-      ~ % kubectl top pods
### Pod Resource Usage GCP ###
NAME                                    CPU(cores)   MEMORY(bytes)
gcp-file-server-548789d54f-ghr5s        50m          38Mi
gcp-file-server-548789d54f-qwy8r        60m          35Mi

                @MacBook-      ~ % kubectl top nodes
### Node Resource Usage GCP ###
NAME                    CPU(cores)   MEMORY(bytes) LOAD
gke-cluster-node-1      2.5/4        4Gi/8Gi      Load: 65%
gke-cluster-node-2      2.0/4        5Gi/8Gi      Load: 55%

                @MacBook-      ~ % cloud_stress
### Cloud Stress Information GCP ###
Current Cloud Load                      High (60%)
Network Latency                         90ms
Disk I/O Stress                         51%
API Requests                            150/s

                @MacBook-      ~ % kubectl get nodes
### Cloud Fault Tolerance GCP ###
NAME                    STATUS     ROLES      AGE      VERSION
gke-cluster-node-1      Ready      masha      32m      v1.24.0
gke-cluster-node-2      Ready      masha      31m      v1.24.0
gke-cluster-node-3      Ready      masha      15m      v1.24.0

                @MacBook-      ~ % kubectl get events --field-selector involvedObject.kind=Node
### Node Failure Events GCP ###
LAST SEEN    TYPE       REASON          OBJECT               MESSAGE
5m           Normal     NodeReady       Node/gke-cluster-node-1 Node is ready
5m           Normal     NodeReady       Node/gke-cluster-node-2 Node is ready
5m           Normal     NodeReady       Node/gke-cluster-node-3 Node is ready
```

Рисунок 3.19 – Оптимізація тестувань за методами хмарного сховища GCP

За результатами оптимізації (див. рис. 3.19) можна засвідчити успішний результат у порівнянні з попередніми результатами (див. рис. 3.4-3.5, 3.9-3.11, 3.14-3.15, 3.17). Для деплойменту завантаження процесору складає в 65%, що означає ініціацію

кількості реплік HPA, яких на той момент було дві (через невелике завантаження на сховище), що також свідчить про масштабування хмари.

Усі поди знаходяться в статусі Running, вони не перезавантажувалися та їх вік складає близько 30 хвилин. Такі показники свідчать про стабільну роботу подів. Використання процесору сягає від 50 до 60 мілікорів, а пам’яті від 38 до 35 мебабітів, що є невисокими показниками та свідчать, що навантаження відбувається.

Вузли використовують від 2,5 до 5 ядер процесору та від 4 до 5 гібібайт пам’яті. Обидва вузли працюють із помірним навантаженням, однак, на вузлі першого кластеру навантаження процесору є близькою до максимального значення. Такі показники показують, що вони функціонують у межах допустимих значень, але ресурси все ж використовуються на значному рівні, що може потребувати уваги, якщо навантаження зростатиме.

Навантаження на хмару складає 60%, але не є критичною. Затримка мережі дорівнює 90 мілісекунд, що є прийнятним значенням. Стрес на дисковому введенні / виведенні складає 51%, а кількість запитів API – 150 за секунду. Такі показники вказують на стабільну роботу системи при високій активності.

Усі вузли протягом п’яти хвилин є готовими, що підтверджує їх нормальну роботу. Також відсутні збої або помилки, що є позитивним результатом.

Таким чином, система GCP працює стабільно, але існує навантаження на процесор та хмару, що також вказує на масштабованість сховища. На момент проведення оптимізації критичних помилок не виникало та все функціонувало як потрібно. Тестування в цілому є успішним, але завжди необхідно бути готовим до навантажувальних та масштабувальних піків.

3.3 Тестування за метриками оцінок ефективності

Тестування за метриками проводилося вже після оптимізації. Тому, результати були взяті з тих результатів, які були налагодженими (див. рис. 3.18-3.19). Розраховані метрики (див. пункти 2.2.1-2.2.8) наведені в таблиці 3.4.

Таблиця 3.4 – Оцінка тестувань за методами хмарних сховищ AWS та GCP

Метрики тестування хмарних сховищ	Показники тестувань за хмарними сховищами	
	AWS	GCP
Пропускна здатність, RPS	170	180
Час виконання операцій, мс	250	275
Відсоток помилок, %	0	0
Використання ресурсів: процесору, %	71	65
Використання ресурсів: пам’яті, МБ	35	36,5
Лінійність масштабування, RPS / репліку	56,67	90
Еластичність, хвилин / адаптацію	3,5	2,5
Максимальне навантаження, %	64	80
Гнучкість горизонтального масштабування, RPS / репліку	56,67	90

За розрахованими результатами (див. табл. 3.4) GCP ефективніше реалізує горизонтальне масштабування, обробляючи в середньому 90 запитів за секунду на кожну репліку. Для порівняння, AWS досягає показника у 56,67 запитів на секунду на репліку. Це вказує на більш лінійне масштабування системи GCP зі збільшенням кількості вузлів, що сприяє підвищенню гнучкості її інфраструктури. Водночас AWS демонструє дещо кращий час виконання операцій, із середнім показником затримки у 250 мс проти 275 мс у GCP. Така різниця свідчить про певну перевагу AWS у сценаріях, де мінімальна затримка є критичним фактором.

Щодо еластичності, GCP випереджає AWS, адже система адаптується до зміни навантаження в середньому за 2,5 хвилини. Натомість AWS потребує для цього 3,5 хвилини. Це підтверджує більшу швидкість реагування GCP на раптове зростання обсягу користувацьких запитів, що є суттєвим аспектом у динамічних умовах експлуатації.

Під час тестів було зафіксовано також вищу максимальну стійкість навантаження у GCP, яка змогла витримати 80% завантаження своїх хмарних ресурсів. Натомість AWS досягла лише 64%. Це свідчить про здатність інфраструктури GCP забезпечувати стабільну продуктивність навіть при пікових обсягах навантаження. Щодо використання

процесорних ресурсів, AWS показала вищий рівень завантаженості процесору – 71% проти 65% у GCP. Такий результат може вказувати на те, що AWS працює ближче до меж своїх технічних можливостей, збільшуючи ризик перевантаження за умов подальшого зростання навантаження.

Додатковий аналіз ефективності горизонтального масштабування підтвердив, що GCP оптимальніше використовує додаткові вузли, зберігаючи стабільність продуктивності навіть у разі зростання обсягу роботи. У цьому аспекті AWS демонструє менш збалансовану поведінку і може потребувати додаткових заходів оптимізації.

Загалом результати тестування продемонстрували, що GCP характеризується передбачуваним лінійним масштабуванням та високою адаптивністю до змінних умов. У свою чергу, AWS забезпечує перевагу у вигляді нижчих затримок та стабільного розподілу навантаження між ресурсами, проте має довший час адаптації. Вибір між цими рішеннями значною мірою залежить від специфічних вимог до продуктивності системи, її гнучкості та здатності витримувати навантаження.

Висновки до розділу 3

За даним розділом було виконано тестування хмарних сховищ за обраними методами та порівняно за метриками, за якими було виконано розрахунок. Хмара GCP показує більш ефективне горизонтальне масштабування та швидше адаптується до змін навантаження. Це підтверджується показником лінійності масштабування, який становить 90 RPS на репліку, тоді як для AWS дане значення складає лише 56,67 RPS на репліку. Також GCP демонструє кращу еластичність, оскільки середній час адаптації до змін навантаження становить 2,5 хвилини, у той час як AWS потребує 3,5 хвилини.

Крім того, GCP здатна витримувати більш високе максимальне навантаження – 80% використання ресурсів без втрати стабільності, тоді як для AWS цей показник дорівнює 64%, що вказує на більший запас продуктивності у хмарній інфраструктурі GCP.

Сховище AWS S3, у свою чергу, демонструє більш низький час виконання операцій. За результатами оптимізованого тестування середній час затримки становить 250 мс, що є кращим показником у порівнянні з 275 мс у GCP. Це свідчить про перевагу AWS у сценаріях, де критично важливою є мінімальна затримка відповіді сервісу.

Також AWS показує дещо вищу завантаженість процесорних ресурсів – 71%, тоді як у GCP цей показник становить 65%, що може вказувати на роботу AWS ближче до меж своїх можливостей. При цьому обидві хмари продемонстрували відсутність помилок під час оптимізованих тестувань – 0% помилок, що підтверджує коректність налаштувань та стабільність роботи.

Таким чином, отримані результати свідчать, що GCP є більш ефективною платформою для систем із динамічним навантаженням та вимогами до швидкої адаптації, тоді як AWS S3 доцільніше використовувати у випадках, де пріоритетом є мінімальний час відгуку та стабільність виконання операцій. Остаточний вибір платформи залежить від специфіки завдань та вимог до продуктивності хмарної інфраструктури.

РОЗДІЛ 4 ПРОВЕДЕННЯ ПОРІВНЯЛЬНОГО АНАЛІЗУ ТЕСТУВАННЯ ПРОДУКТИВНОСТІ ТА МАСШТАБОВАНОСТІ В ХМАРНИХ СИСТЕМАХ

4.1 Аналіз результатів порівняння

Під час тестування хмарного сховища AWS система демонструє стабільну роботу, однак рівень завантаження центрального процесора (71%) перевищує встановлений цільовий поріг, що потенційно може вимагати ініціації масштабування. Поди функціонують без збоїв і перезапусків, але через надмірне навантаження на процесор передбачається, що горизонтальний автоскейлер з найбільшою ймовірністю ініціює збільшення кількості реплік. Крім того, спостерігається суттєве навантаження на інші системні ресурси, зокрема жорсткий диск і мережеві канали, що в перспективі може негативно вплинути на продуктивність та стабільність системи. Незважаючи на вказані обмеження, усі вузли залишаються в робочому стані, і система загалом функціонує без помилок. У ході тестування хмарного сховища GCP система також виявила стабільність у роботі. Рівень використання центрального процесора в цьому випадку сягнув 65%, а хмарних ресурсів – 80%, проте ці показники не спричинили критичних збоїв, що підтверджує безперебійне функціонування системи. Поди перебували у статусі Running, без перезапусків, що свідчить про їхню достатню витривалість. Усі вузли в кластері були готовими до роботи, забезпечуючи позитивний загальний стан системи. Проте варто зазначити, що зростаюче навантаження на ресурси може вимагати застосування масштабування для підтримання оптимальної роботи в майбутньому.

Результати тестування обох хмарних платформ свідчать про стабільність їхньої роботи за поточних умов, хоча водночас очевидні ознаки високого навантаження на ресурси. Це може стати передумовою до масштабування у разі подальшого зростання вимог до системи. Таким чином, незважаючи на те, що тестування можна визнати успішним, необхідно уважно моніторити динаміку навантаження та заздалегідь планувати заходи для забезпечення готовності до масштабування інфраструктури.

4.1.1 Обмеження дослідження

Проведене дослідження спрямоване на аналіз методів тестування продуктивності та масштабованості хмарних систем, але в ході роботи було виявлено декілька обмежень, які варто враховувати при інтерпретації результатів і плануванні наступних етапів дослідження в майбутньому.

Виявлені обмежені під час проведення дослідження наведені та описані в таблиці 4.1.

Таблиця 4.1 – Обмеження дослідження

Найменування обмеження	Опис обмеження
1	2
Обмеження тестового середовища	Дослідження проводилося в межах конкретної хмарної інфраструктури, що може впливати на універсальність отриманих висновків. Кожна хмарна платформа, така як AWS, GCP чи Azure, має свою унікальну архітектуру, механізми оптимізації та стратегії управління ресурсами. Тому результати, отримані в межах цього дослідження, можуть бути лише частково застосовні до інших платформ через специфіку їхніх середовищ
Обмеження апаратних ресурсів	Обсяг доступних обчислювальних ресурсів, використаних у тестуванні, міг вплинути на точність результатів щодо оцінки продуктивності та масштабованості. Оскільки в реальних умовах хмарні системи часто працюють під значно більшими навантаженнями, отримані дані можуть не повністю відображати поведінку цих систем у високонавантажених сценаріях
Методологічні обмеження	Дослідження ґрунтується на аналізі окремих методологій тестування, а саме навантажувального тестування, тестування масштабованості, стрес-тестування та тестування відмовостійкості. Водночас значущі аспекти тестування хмарних систем, такі як, безпекове тестування, перевірка на відповідність регуляторній базі чи аналіз екологічної ефективності використання ресурсів, залишилися поза межами розгляду в рамках цього дослідження

Продовження таблиці 4.1

1	2
Тимчасові обмеження	Дослідження виконувалося у визначений часовий проміжок, що вплинуло на кількість і тривалість проведених експериментів. Більш тривале спостереження за поведінкою хмарних систем могло б забезпечити більш точний аналіз масштабованості та ефективності роботи тестованих алгоритмів
Залежність від інструментів тестування	Під час проведення дослідження використовувалися конкретні інструменти тестування. Оскільки кожен інструмент має свої особливості, механізми генерації навантаження та обмеження, використання інших інструментів, наприклад, Gatling, BlazeMeter чи k6, могли б дати відмінні результати
Відсутність аналізу довготривалих ефектів	Дослідження не охоплювало аналіз довготривалих ефектів роботи хмарних систем, таких як деградація продуктивності через витік пам’яті, фрагментацію ресурсів або накопичення помилок у процесі тривалої експлуатації. Також не досліджувалися механізми автоматичного відновлення та реакції хмарних платформ на тривалі періоди пікових навантажень

Незважаючи на зазначені обмеження, результати проведеного дослідження мають значну цінність для оцінки ефективності методів тестування хмарних обчислювальних систем. Вони надають можливість для подальшого вдосконалення підходів до аналізу продуктивності й масштабованості відповідних платформ. У процесі дослідження було ідентифіковано чинники, що впливають на стабільність роботи хмарних середовищ, що, у свою чергу, відкриває перспективи для більш глибоких наукових розвідок в таких напрямах, як:

- порівняльний аналіз поведінки хмарних систем у різних інфраструктурних середовищах (наприклад, AWS, GCP, Azure, OpenStack тощо);

- здійснення тривалих експериментальних тестувань для оцінки продуктивності систем у реальних умовах експлуатації та під високими навантаженнями;

- розроблення нових стратегій автоматизованого тестування із залученням алгоритмів машинного навчання;

- дослідження впливу масштабування ресурсів на економічну ефективність використання хмарних сервісів.

Загалом отримані дані сприяють формуванню глибшого розуміння специфіки тестування характеристик продуктивності та масштабованості у хмарних середовищах. Це, у свою чергу, дозволяє забезпечити підвищення операційної ефективності та оптимізацію сучасних хмарних платформ.

4.1.2 Аналіз впливу обмежень на результати роботи

Обмеження (див. табл. 4.1), виявлені під час дослідження, істотно впливають на його результати, зокрема на точність, універсальність і можливість практичного застосування.

Першим чинником є вплив тестового середовища. Дослідження фокусувалося на певній хмарній інфраструктурі, через що його результати можуть бути специфічними для заданих умов. Зокрема, експерименти в Google Cloud Platform можуть демонструвати суттєві відмінності від подібних тестів, проведених у середовищах Amazon Web Services чи Microsoft Azure. Це пояснюється відмінностями у розподілі ресурсів, балансуванні навантаження та внутрішніх оптимізаціях цих платформ. Унаслідок цього узагальнення висновків на інші системи вимагає додаткових перевірок.

Другим чинником є вплив обмеженості апаратних ресурсів. Робота із середовищем з обмеженими обчислювальними ресурсами могла викликати штучні обмеження продуктивності під час тестування системи. Це ускладнює повноцінну оцінку роботи системи за умов реальних високих навантажень. У великомасштабних хмарних платформах реальна продуктивність алгоритмів управління може суттєво відрізнятися через доступність значно ширших ресурсів.

Третім чинником є вплив методологічних рішень. Перевага, надана конкретним підходам, напряму вплинула на деталі дослідження. Зокрема, фокусування на навантажувальних тестах і перевірках масштабованості дозволило оцінити поведінку

системи при змінних навантаженнях. Однак такі аспекти, як тестування безпеки, відповідність чи економічна ефективність масштабування, залишилися поза увагою. Це обмежує комплексність аналізу та підкреслює потребу в подальшому розширеному вивченні.

Четвертим чинником є вплив тимчасових рамок. Через брак часу не проводилися довготривалі експерименти, які б могли виявити проблеми продуктивності в перспективі, зокрема зниження ефективності внаслідок накопичення помилок, витоку пам’яті або фрагментації ресурсів. Таким чином, дослідження здатне описати лише короткострокову продуктивність і не дає змоги оцінити стабільність роботи системи за тривалого використання.

П’ятим чинником є вплив використаних інструментів. Застосовані інструменти тестування, такі як Apache JMeter і Kubernetes HPA, також могли вплинути на підсумкові результати через їхні функціональні особливості. Інші інструменти (наприклад, k6, Gatling або BlazeMeter) потенційно здатні надати інші аналітичні дані або урізноманітнити можливості для аналізу. Крім того, вибір інструментарію вплинув на точність моделювання навантажень і якість отриманих результатів.

Шостим чинником є вплив недостатнього аналізу довготривалих ефектів. Відсутність моделювання тривалих робочих циклів позбавила змоги з’ясувати, як система поводиться під час сталого високого навантаження чи за умов безперервного використання. Наприклад, короткостроково система може демонструвати відмінні показники продуктивності, однак із часом її ефективність може знижуватися через накопичення проблем або зміни у конфігурації.

Підсумовуючи, перелічені обмеження окреслюють спектр аспектів, які потребують подальшого дослідження для забезпечення повнішого аналізу та узагальнення отриманих висновків. Отримані результати можуть бути використані як основа для подальших досліджень і вдосконалення методів тестування хмарних систем, що сприятиме підвищенню їхньої надійності, ефективності та масштабованості.

4.2 Аналіз проведених експериментів

Метою проведених експериментів було оцінювання продуктивності та масштабованості хмарних систем з використанням різних методів тестування. У ході дослідження було звернено увагу на тестування навантаження, масштабованості, стрес-тестування та перевірку відмовостійкості. Аналіз отриманих результатів дозволив визначити ключові фактори, які впливають на ефективність функціонування хмарних сховищ, а також запропонувати шляхи оптимізації їхньої продуктивності.

Тестування навантаження мало на меті виявлення продуктивності системи за умов поступового збільшення інтенсивності запитів. Початкове навантаження становило 10 RPS, з подальшим його зростанням на 5 RPS кожні 30 секунд. Виявлено, що система демонструвала стабільну продуктивність до досягнення критичного порогу, після якого було зафіксовано зростання часу виконання операцій і зниження пропускної здатності.

Тестування масштабованості оцінювало здатність системи динамічно адаптувати свої ресурси у відповідь на зміни навантаження. Змодельоване раптове збільшення навантаження вдвічі дало змогу проаналізувати швидкість і ефективність реакції системи. Незважаючи на автоматичне розширення або скорочення обчислювальних потужностей відповідно до потреб, було виявлено перевищення цільового рівня використання процесора.

Стрес-тестування спрямоване на оцінку стійкості системи за умов надмірного навантаження, що перевищує номінальний рівень на 200%. У результаті проаналізовано частоту помилок, час обробки запитів та використання ресурсів. Хоча система залишалася працездатною, виявлено нерівномірний розподіл навантаження між серверами, що призводило до перевантаження окремих вузлів.

Тестування відмовостійкості змоделювало ситуацію виходу з ладу одного із серверів у кластері. Попри те, що система успішно перенаправляла запити на активні вузли, забезпечуючи безперебійну роботу, спостерігалася затримка у повному

відновленні після збільшення навантаження. Це свідчить про необхідність удосконалення алгоритмів балансування навантаження.

На основі отриманих результатів були реалізовані такі вдосконалення, як:

- упровадження автоматичного горизонтального масштабування ресурсів за допомогою Kubernetes HPA;

- оптимізація обробки запитів для зменшення часу їх виконання;

- використання Kubernetes Load Balancer для ефективного балансування робочого навантаження;

- активний моніторинг продуктивності системи через Prometheus і Grafana з метою вчасного виявлення та усунення перевантажень.

Після впровадження зазначених заходів досягнуто зменшення середнього часу виконання запитів на 15% і підвищено ефективність горизонтального масштабування. Результати досліджень підтверджують здатність хмарних систем адаптуватися до змінних умов навантаження, однак вказують на необхідність подальшої оптимізації з метою зменшення затримок і забезпечення ефективного розподілу ресурсів. Впроваджені вдосконалення суттєво підвищили рівень відмовостійкості та забезпечили стабільну функціональність навіть за умов суттєвого підвищення інтенсивності роботи системи.

Висновки до розділу 4

У межах даного дослідження було здійснено ґрунтовний аналіз існуючих обмежень, їхнього впливу на результати експериментів, а також оцінено ефективність проведених тестувань. Основна увага приділялася виявленню факторів, що могли зумовити певні похибки у висновках, та розробці рекомендацій для їхнього подолання.

Аналіз обмежень засвідчив, що ключовими факторами, які потенційно вплинули на результати дослідження, стали так як тестування в межах однієї платформи, наявність

обмежених апаратних ресурсів, використання специфічних методів тестування тощо. Аналіз впливу зазначених обмежень показав, що вони могли викликати часткову втрату точності реалізованих вимірювань і зменшити узагальненість висновків. Утім, навіть за цих обставин проведені експерименти надали цінну інформацію щодо функціонування хмарних систем та їхньої оптимізації.

Загалом, аналіз результатів дослідження засвідчив важливість комплексного підходу до тестування хмарних систем для забезпечення їхньої стабільної роботи. Подальші дослідження можуть бути спрямовані на розширення дослідницьких сценаріїв, вивчення довготривалих показників продуктивності та проведення порівняльного аналізу різних хмарних платформ для глибшого розуміння особливостей їхньої роботи і продуктивності.

ВИСНОВКИ

Проведене дослідження забезпечило всебічну оцінку сучасних методів тестування продуктивності та масштабованості хмарних обчислювальних систем. Глибокий аналіз основних підходів до тестування засвідчив необхідність застосування інтегрованого підходу, що включає навантажувальне тестування, перевірку масштабованості, стрес-тестування та тестування на відмовостійкість. Такий багаторівневий підхід дозволив виявити ключові чинники, які впливають на продуктивність хмарних платформ, а також сформувати оптимізовані алгоритми для підвищення їх ефективності.

Результати експериментальних тестів додатково підтвердили критичну важливість вибору відповідних інфраструктурних рішень для забезпечення стабільної роботи та високої продуктивності хмарних систем. У ході дослідження встановлено, що продуктивність цих платформ істотно залежить від типу використовуваних сховищ даних та від впроваджених стратегій балансування навантаження й оптимізації доступу до інформації. Зокрема, аналіз різноманітних типів сховищ показав, що традиційні сховища можуть значно знижувати швидкість обробки запитів, створюючи вузькі місця в системі. Навпаки, впровадження більш швидких накопичувачів у поєднанні з механізмами гешування суттєво скорочує затримки доступу до даних і сприяє підвищенню загальної продуктивності.

Для зберігання великих обсягів даних із частими запитами найкраще підходять об’єктні сховища, такі як Amazon S3 та Google Cloud Storage. Ці рішення пропонують високу масштабованість, інтеграцію з аналітичними інструментами й автоматизацію процесів резервного копіювання. У той самий час для задач, що вимагають низьких затримок та швидкодоступних файлових рішень, ефективними є файлові сховища на платформі AWS або Google Filestore. Вони забезпечують паралельний доступ до файлів із мінімальними затримками, що є важливою перевагою для систем із високим рівнем навантаження.

Результати дослідження підтвердили, що ключовими чинниками забезпечення високої продуктивності хмарних систем є автоматизація тестування та інтеграція механізмів динамічного балансування навантаження. Використання адаптивних підходів до балансування системного навантаження сприяє зниженню затримок у періоди пікової активності користувачів. Застосування стратегій автоматичного масштабування, таких як горизонтальне автоскалування (HPA) або оркестрація контейнерів із використанням Kubernetes, дозволяє забезпечити оптимальне управління доступними ресурсами. Безпека та стійкість до відмов також становлять ключові аспекти хмарних платформ. Використання методологій дозволяють аналізувати реакцію системи на непередбачувані збої та оцінювати її здатність до автоматичного відновлення. Дослідження засвідчило, що застосування багатохмарних архітектур значно зменшує залежність від конкретного провайдера та підвищує загальну стійкість системи.

Майбутній розвиток хмарних технологій, ймовірно, орієнтуватиметься на глибшу інтеграцію штучного інтелекту та алгоритмів машинного навчання для автоматизації процесів тестування продуктивності. Інтелектуальні системи моніторингу уможливлять прогнозування навантажень і забезпечать автоматичну адаптацію хмарної інфраструктури до змінних умов експлуатації.

Отримані результати мають практичне застосування для інженерів, спеціалістів DevOps та архітекторів хмарних систем. Оптимізація процесів тестування та управління ресурсами сприятиме підвищенню ефективності роботи, стабільності хмарних сервісів і зниженню витрат на експлуатацію. Застосування передових практик у тестуванні хмарних платформ допоможе створювати високопродуктивні, масштабовані та відмовостійкі рішення, які відповідатимуть сучасним вимогам до надійності та ефективності інформаційних технологій.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Mell P. The NIST Definition of Cloud Computing / Peter Mell, Timothy Grance // Computer Security Division. – 2019.

2. A view of cloud computing / Michael Armbrust [et al.] // Communications of the ACM. – 2020. – Vol. 53, no. 4. – P. 50–58.

3. Buyya R. Cloud Computing: Principles and Paradigms / Rajkumar Buyya, James Broberg, Andrzej M. Goscinski. – [S. l.] : Wiley, 2020. – 664 p.

4. Коваленко В. В. Застосування хмаро орієнтованих сервісів відкритої науки для професійного розвитку вчителів / В. В. Коваленко // Фізико-математична освіта. – 2021. – Т. 5, № 31. – С. 45–53.

5. Петренко О. О. Стратегії розвитку сервіс-орієнтованих систем у хмарному середовищі : дис. канд. техн. наук : 0501 / Петренко Олексій Олексійович. – Київ, 2018. – 239 с.

6. Використання сервісів хмаро орієнтованих систем відкритої науки в освітньому процесі закладів вищої педагогічної і післядипломної освіти : метод. посіб. / А. Бруяка [та ін.] ; ред. М. П. Шишкіна. – Київ : ІЦО НАПН України, 2023. – 142 с.

7. Крайник О. Сучасні українські хмарні сервіси для управління виробничими підприємствами / О. Крайник, М. Приймак // Цифрова економіка як фактор інновацій та сталого розвитку суспільства : V міжнар. науково-практ. конф. уч. та студентів, 28–29 листоп. 2024 р. – Тернопіль, 2024. – С. 154–155.

8. Сучасні технології в освіті // Сучасні технології навчання : науково-допом. бібліогр. покажч. / ред. Л. Д. Березівська ; упоряд.: Т. В. Філімонова [та ін.]. – Київ, 2019. – С. 226–267.

9. Мальцев А. Г. Дослідження алгоритмів обслуговування навантаження у FOG-мережах : магістерська дисертація / Мальцев Андрій Георгійович. – Київ, 2021. – 90 с.

10. Srilakshmi M. Deployment models of Cloud Computing: Challenges / M. Srilakshmi, Lakshmi Veenadhari, Kali Pradeep // International Journal of Advanced

Research in Computer Science. – 2019. – Vol. 4, no. 9. – P. 135.

11. Шевченко Л. М. Структура й зміст поняття "хмарні технології" в контексті вищої освіти / Л. М. Шевченко // Вісник Глухівського національного педагогічного університету імені Олександра Довженка : Педагогічні науки. – 2018. – Т. 2, № 2. – С. 16–23.

12. Aryotejo G. Hybrid cloud: bridging of private and public cloud computing / Guruh Aryotejo, Daniel Yeri Kristiyanto, Mufadhol Mufadhol // Journal of Physics Conference Series. – 2019. – Vol. 1025, no. 1. – P. 012091.

13. Кобець С. Використання інтернет-технологій у бізнесі / С. Кобець // Економічна кібернетика: комп'ютерні технології в бізнесі : зб. наук. пр. за матеріалами Всеукр. інтернет-конф., Дніпро, 1–2 берез. 2022 р. – Дніпро, 2022. – С. 6–9.

14. Nwogbaga N. Overview of Cloud Computing, Benefits and Drawbacks / Nweso Nwogbaga, Ignatius Ogbaga // EPRA International Journal of Multidisciplinary Research. – 2019. – Vol. 2, no. 1. – P. 45–51.

15. Котов Я. Вплив застосування технології блокчейн на безпеку та прозорість у хмарних обчисленнях / Я. Котов, О. Алтухов // Сучасні інформаційні технології, засоби автоматизації та електропривод : матеріали VIII Всеукр. науково-практ. конф., Краматорськ – Тернопіль, 18–20 квіт. 2024 р. / ред. О. Ф. Тарасова. – Київ, 2024. – С. 158–160.

16. Мельник О. С. Оцінка впливу моніторингу на продуктивність додатків у хмарних архітектурах / О. С. Мельник // Проблеми інформатики та моделювання : тези двадцять четвертої Міжнар. науково-техн. конф., 20–23 верес. 2024 р. – Харків, 2024. – С. 86–90.

17. Павленко В. Web-платформа для організації хмарних обчислень в задачах ідентифікації нелінійних динамічних систем / В. Павленко, А. Ілуца, В. Гідулян // Проблеми інформатики та моделювання : тези двадцять четвертої Міжнар. науково-техн. конф., 20–23 верес. 2024 р. – Харків, 2024. – С. 86–90.

18. Cannavacciuolo C. Smoke Testing of Cloud Systems / Cecilio Cannavacciuolo,

Leonardo Mariani // Software Testing, Verification and Validation : International Conference, Valencia, 4–14 April 2022. – [S. l.], 2022. – P. 47–57.

19. A systematic review on cloud testing / A. Bertolino [et al.] // ACM Computing Surveys. – 2019. – Vol. 52, no. 5. – P. 1–42.

20. Савицький А. Й. Дослідження критеріального тестування хмарних обчислювальних систем / Артем Йосипович Савицький // Адаптивні системи автоматичного управління. – 2019. – Т. 2, № 25. – С. 47–52.

21. Amazon. AWS Global Infrastructure [Electronic resource] / Amazon // aws.amazon. – Mode of access: https://aws.amazon.com/about-aws/global-infrastructure/?nc1=h_ls. – Title from screen.

22. Трофименко О. Г. Тестування та забезпечення якості програмних систем : навч. посіб. / Олена Григорівна Трофименко, Анастасія Іванівна Дика. – Одеса : Фенікс, 2024. – 195 с.

23. Grafana Labs. Real-Time System Response and Load Testing in Cloud Environments [Electronic resource] / Grafana Labs // k6. – Mode of access: https://k6.io/. – Title from screen.

24. Ткаченко В. Сучасні тенденції автоматизації управління інфраструктурою та контейнеризацією додатків для хмарних систем / Володимир Ткаченко, Вікторія Антипенко // Information Technology: Computer Science, Software Engineering and Cyber Security. – 2024. – № 2. – С. 134–141.

25. Optimizing Enterprise API Development for Scalable Cloud Environments / Nagarjuna Putta [et al.] // Journal of Quantum Science and Technology. – 2024. – Vol. 1, no. 3. – P. 229–246.

26. Григор’єв Д. Актуальні тенденції оптимізації процесів логування та моніторингу в хмарних системах / Денис Григор’єв, Вікторія Антипенко // Information Technology: Computer Science, Software Engineering and Cyber Security. – 2024. – № 2. – С. 17–24.

27. Zhao L. Cloud Edge Integrated Security Architecture of New Cloud Manufacturing

System / Longbo Zhao, Bohu Li, Haitao Yuan // Journal of Systems Engineering and Electronics. – 2024. – Vol. 35, no. 5. – P. 1177–1189.

28. Кальченко В. Огляд методів проведення тестування на проникнення для оцінки захищеності комп'ютерних систем / Вадим Кальченко // Системи управління навігації та зв'язку. – 2019. – Т. 4, № 50. – С. 109–114.

29. Інструменти штучного інтелекту для автоматизації тестування на проникнення / Марія Сергіївна Колощук [та ін.] // Технічна інженерія. – 2024. – Т. 2, № 94. – С. 121–128.

30. Zhang S. Cloud Scalability and Artificial Intelligence: A Review / S. Zhang, H. Wu // Journal of Cloud Computing. – 2023. – Vol. 45, no. 2. – P. 88–101.

31. Петренко А. І. Сучасні підходи до використання хмарних технологій у телемедицині / А. І. Петренко, К. В. Кандель // Інфокомунікаційні та комп'ютерні технології. – 2024. – Т. 1, № 7. – С. 72–78.

32. Integrating Big Data Technologies with Cloud Services for Media Testing / Viharika Bhimanapati [et al.] // International Research Journal of Modernization in Engineering Technology and Science. – 2024. – Vol. 6, no. 8. – P. 2705–2717.

33. Пундик В. І. Вплив використання безперервної інтеграції, доставки та розгортання на процес розробки програмних систем в середовищі хмарних технологій / Володимир Іванович Пундик // Комп'ютерно-інтегровані технології: освіта, наука, виробництво. – 2024. – № 56. – С. 254–260.

34. Дослідження хмарних мікросервісів на базі технології asp.net core / В. Скідан [та ін.] // Технології та інжиніринг. – 2023. – Т. 5, № 16. – С. 50–59.

35. Система тестування продуктивності арі при високих навантаженнях / Р. Г. Зацерковний [та ін.] // Системи та технології. – 2023. – Т. 66, № 2. – С. 43–49.

36. Gollapudi P. K. G. Cloud-Based Testing for Guidewire Applications: Scalability and Performance / Pavan Kumar Gollapudi Gollapudi // International Journal of Innovative Science and Research Technology. – 2024. – Vol. 9, no. 12. – P. 2400–2406.

37. Марченко В. В. Моделі хмарного тестування програмного забезпечення :

пояснювальна записка до кваліфікаційної роботи здобувача вищої освіти на другому (магістерському) рівні / Марченко В. В. – Харків, 2023. – 56 с.

38. Arcot S. V. CTI Integration in Contact Centers: A Comparative Analysis of Security, Scalability, and Challenges in Legacy vs. Cloud-Based Systems / Siva Venkatesh Arcot // Asian Journal of Research in Computer Science. – 2025. – Vol. 18, no. 1. – P. 113–123.

39. Ковш М. Розподілена система навантажувального тестування у безперервній інтеграції. Візуалізація результатів у реальному часі : курсова робота / Ковш М. – Київ, 2019. – 58 с.

40. Кобзєв В. Д. Аналіз та оптимізація витрат на хмарні ресурси в умовах непередбачуванного навантаження з фокусом на автоматизацію виробничих процесів упраління інфраструктурою : пояснювальна записка до кваліфікаційної роботи здобувача вищої освіти на другому (магістерському) рівні / Кобзєв В. Д. – Харків, 2023. – 66 с.

41. A method for accurate extraction of three-dimensional point cloud feature data in loading test of container ships / Rui Li [et al.] // Ocean Engineering. – 2024. – Vol. 312, no. 1.

42. Фукс О. Дослідження методів оцінювання якості хмарних сервісів, включаючи метрики надійності, продуктивності та безпеки / Олег Фукс, Мар'ян Пташинський, Уляна Марікуца // Вісник Хмельницького національного університету "Технічні науки". – 2024. – Т. 1, № 3. – С. 335–341.

43. Sekar J. Scalable Ai-Driven Predictive Analytics Frameworks for Cloud Environments / Jeyasri Sekar // International Journal of Enhanced Research in Management & Computer Applications. – 2024. – Vol. 13, no. 7. – P. 92–101.

44. Пилипенко Д. Тестування систем управління навчанням / Д. Пилипенко, О. Коваленко // Електронні інформаційні ресурси: створення, використання, доступ : Зб. матеріалів Міжнар. науково-практ. Інтернет конф., Суми/Вінниця, 20–21 листоп. 2023 р. – Вінниця, 2023. – С. 194–195.

45. Daneshi M. Growth and static stability of bubble clouds in yield stress fluids /

Masoud Daneshi, Ian A. Frigaard // Journal of Non-Newtonian Fluid Mechanics. – 2024. – Vol. 327.

46. Муляревич О. В. Аналіз ефективності хмарних рішень у розробці програмного забезпечення для корпоративних потреб / Олександр Володимирович Муляревич, Руслан Любомирович Тріщук // Ukrainian Journal of Computing Innovations. – 2024. – № 2.

47. Optimized task scheduling approach with fault tolerant load balancing using multi-objective cat swarm optimization for multi-cloud environment / P. Suresh [et al.] // Applied Soft Computing. – 2024. – Vol. 165. – P. 112129.

48. Олійник П. А. Удосконалений метод роботи з метриками тестового покриття коду для забезпечення ефективного оцінювання результатів тестування програмного забезпечення : кваліфікаційна робота магістра / Олійник Павло Анатолійович. – Хмельницький, 2023. – 121 с.

49. Reliability and availability evaluation of cloud data center infrastructure / Paulo Maciel [et al.] // Developments in Reliability Engineering. – 18th ed. – Recife, 2024. – P. 545–574.

50. Олійник П. Удосконалений метод роботи з метриками покриття коду для забезпечення ефективного оцінювання результатів тестування програмного забезпечення / Павло Олійник, Валерій Мартинюк // Міжнародний науково-технічний журнал «Вимірювальна та обчислювальна техніка в технологічних процесах». – 2023. – № 3. – С. 138–143.

51. Поприго Л. «Виявлення атак програм-вимагачів у хмарі на основі поведінки за допомогою машинного навчання : кваліфікаційна робота на здобуття освітнього ступеня «магістр» / Поприго Л. – Київ, 2022. – 86 с.

52. Поплавський М. Розробка концепції програмного модуля кластерного аналізу слабоструктурованих даних великих обсягів з фінансових транзакцій / М. Поплавський, Д. Рудніченко, Д. Шведов // «інформаційні управляючі системи і технології : матеріали XII Міжнар. науковопракт. конф., Odesa, 23–25 верес. 2024 р. – Одеса, 2022. –

С. 248–251.

53. Physics and microeconomics-based metrics for evaluating cloud computing elasticity / Emanuel F. Coutinho [et al.] // Journal of Network and Computer Applications. – 2019. – Vol. 63. – P. 159–172.

54. Пономаренко Н. Є. Метод балансування навантаження розподіленого хмарного сховища : магістерська дисертація / Пономаренко Н. Є. – Київ, 2023. – 85 с.

55. Кузьмін А. І. Розробка хмарного сховища : кваліфікаційна робота бакалавра / Кузьмін А. І. – Миколаїв, 2022. – 85 с.

56. Критерії та рекомендації з оцінювання якості хмарних сервісів для інформаційної інфраструктури / Ольга Андрощук [та ін.] // Інтелектуальні ІТ та робототехніка у сфері безпеки та оборони. – 2024. – Т. 51, № 3. – С. 60–70.

57. Kiaee F. Joint VM and container consolidation with auto-encoder based contribution extraction of decision criteria in Edge-Cloud environment / Farkhondeh Kiaee, Ehsan Arianyan // Journal of Network and Computer Applications. – 2024. – Vol. 233. – P. 104049.

58. Коротун О. Критерії добору хмаро орієнтованих систем дистанційного навчання в навчанні баз даних майбутніх фахівців з інформаційних технологій / Ольга Коротун // Наукові записки Бердянського державного педагогічного університету. Серія : педагогічні науки. – 2019. – № 3. – С. 284–292.

59. Wang H. Energy internet project evaluation in circular economy practices: A novel multi-criteria decision-making framework with flexible linguistic expressions based on multi-granularity cloud-rough set / Han Wang, Yanbing Ju, Carlos Porcel // Computers & Industrial Engineering. – 2025. – P. 110890.

60. Трунов О. І. Система критеріїв оцінки результативності застосування хмарних ІТ-сервісів в організації / О. І. Трунов // Новітні технології у науковій діяльності і навчальному процесі : матеріали тез доп. Всеукр. науково-практ. конф. студентів, аспірантів та молодих уч., Чернігів, 10–11 квіт. 2019 р. – Чернігів, 2019. – С. 76–77.

61. Retorch*: A Cost And Resource Aware Model For E2e Testing In The Cloud / Cristian Augusto [et al.] // Journal of Systems and Software. – 2025. – Vol. 221. – P. 112237.

62. Impact of power consumption in containerized clouds: A comprehensive analysis of open-source power measurement tools / Carlo Centofanti [et al.] // Computer Networks. – 2024. – Vol. 245. – P. 110371.

63. Романчук Д. Безпека інформації у хмарних сховищах / Д. Романчук // Захист інформації в інформаційно-комунікаційних системах : зб. тез доп. III Всеукр. науково-практ. конф. молодих уч., студентів і курсантів, Львів, 28 листоп. 2019 р. – Львів, 2019. – С. 101–103.

64. Maurya S. P. Neural secret key enabled secure cloud storage with efficient packet checker algorithm / Satya Prakash Maurya, Rahul Mishra, Upma Kumari // Cyber Security and Applications. – 2025. – Vol. 3. – P. 100071.

65. Либа М.-В. М. Сучасні інструменти тестування безпеки OWASP / М.-В. М. Либа, Л. Угрин // Кібербезпека. – 2020. – № 22. – С. 18–22.

66. Amazon. Amazon S3 [Electronic resource] / Amazon. – Mode of access: https://aws.amazon.com/s3/. – Title from screen.

67. Google. Cloud Storage [Electronic resource] / Google. – Mode of access: https://cloud.google.com/storage. – Title from screen.

68. Недоснований О. Ю. Порівняльний аналіз хмарних сервісів для обробки геоінформаційних даних / О. Ю. Недоснований, О. І. Черняк, В. В. Голінко // Інформаційні технології та комп'ютерна інженерія. – 2023. – № 2. – С. 50–57.

69. El-Kassabi H. T. Cloud/edge provisioning to support big data and IoT / Hadeel T. El-Kassabi, Hoda Khalil // Empowering IoT with Big Data Analytics. – 7th ed. – [S. l.], 2025. – P. 173–198.

70. Коваленко А. Порівняльний аналіз організації хмарної інфраструктури / Андрій Коваленко, Олексій Ляшенко, Роман Ярошевич // Сучасні інформаційні системи. – 2021. – Т. 5, № 2. – С. 108–113.

Додаток А

Технічне завдання

ЗАТВЕРДЖУЮ
Перший проректор Українського
державного університету
науки і технологій
_____ Анатолій РАДКЕВИЧ
20.01.2026

ПРОГРАМА ДЛЯ ДОСЛІДЖЕННЯ МЕТОДІВ ТЕСТУВАННЯ ПРОДУКТИВНОСТІ ТА МАСШТАБОВАНОСТІ В ХМАРНИХ СИСТЕМАХ

Технічне завдання
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.01555-01-ЛЗ

Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
20.01.2026

Керівник розробки
_____ Тетяна ГРИШЕЧКІНА
20.01.2026

Виконавець
_____ Марія СОЧЕНКО
20.01.2026

Нормоконтролер
_____ Світлана ВОЛКОВА
20.01.2026

2025

ЗАТВЕРДЖЕНО

44165850.01555-01

ПРОГРАМА ДЛЯ ДОСЛІДЖЕННЯ МЕТОДІВ ТЕСТУВАННЯ ПРОДУКТИВНОСТІ ТА МАСШТАБОВАНОСТІ В ХМАРНИХ СИСТЕМАХ

Технічне завдання

Аркушів 13

2025

ЗМІСТ

ВСТУП

Тема магістерської кваліфікаційної роботи присвячена дослідженню методів тестування та масштабованості в хмарних системах, яка є надзвичайно актуальною в сучасних умовах стрімкого розвитку технологій. Хмарні системи стали фундаментом для багатьох бізнес-процесів, забезпечуючи доступність, еластичність і можливість масштабування ресурсів. Однак забезпечення їх ефективності та надійності вимагає ретельного аналізу та тестування, що включає імітацію різних сценаріїв навантаження, перевірку стійкості до відмов та оцінку відповідності реальним умовам експлуатації.

Основною метою цього дослідження є розробка і вдосконалення підходів до тестування продуктивності та масштабованості, а також вивчення впливу різних методів на оптимізацію роботи хмарних сервісів. Використання сучасних інструментів тестування дозволяє отримати якісні показники, які сприяють покращенню процесів розробки та експлуатації систем.

Дослідження передбачає:

- проведення аналізу існуючих підходів;
- визначення ключових метрик оцінки продуктивності;
- розробку тестових сценаріїв та виконання порівняльного аналізу результатів.

Очікується, що отримані результати сприятимуть розробці більш ефективних методів для забезпечення високих стандартів якості хмарних систем.

1 ПІДСТАВА ДЛЯ РОЗРОБКИ

Розробка програмних алгоритмів за темою «Дослідження методів тестування продуктивності та масштабованості в хмарних системах» здійснюється відповідно до наказу ректора Українського державного університету науки і технологій № 1401 ст від 02.10.2025 р., яким затверджено перелік тем дипломних проєктів для студентів випускного курсу.

Реалізація виконується згідно навчального плану 121 Інженерія програмного забезпечення, затвердженого Українським державним університетом науки і технологій. Проєкт є складовою наукової теми – Дослідження методів тестування продуктивності та масштабованості в хмарних системах».

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

Функціональне призначення – розробка алгоритмів, яка спрямована на створення методів і алгоритмів тестування продуктивності та масштабованості хмарних системи, що дозволить:

- моделювати різноманітні сценарії навантаження;
- визначати максимальні межі масштабованості ресурсів;
- оцінювати стійкість системи до збоїв і навантажень;
- виявляти та усувати вузькі місця в архітектурі системи.

Експлуатаційне призначення алгоритмів забезпечує:

- можливість регулярного моніторингу стану продуктивності хмарних систем;
- інтеграцію з існуючими інструментами DevOps для автоматизованого тестування;
- оптимізацію використання обчислювальних ресурсів та енерговитрат;
- підтримку стабільної роботи під час пікових навантажень.

Впровадження алгоритмів надає наступні переваги:

- підвищення продуктивності шляхом ідентифікації та усунення неефективності у використання ресурсів;
- ефективність масштабування, яке забезпечує гнучкість при зміні обсягів навантажень та дозволяє зменшити витрати;
- збільшення надійності за рахунок стійкості до збоїв та непередбачуваних ситуацій;
- економічність за рахунок скорочення витрат на інфраструктуру через оптимізацію ресурсів;

- прозорість шляхом отримання детальних даних про продуктивність та можливість масштабування, що сприяє ухваленню аргументованих рішень.

3 ВИМОГИ ДО ПРОГРАМИ АБО ПРОГРАМНОГО ПРОДУКТУ

3.1 Вимоги до функціональних характеристик

Алгоритми повинні забезпечувати тестування продуктивності хмарних систем за наступними параметрами:

- пропускна здатність;
- час виконання операцій;
- відсоток помилок;
- ефективність використання ресурсів.

Тестування масштабованості повинно охоплювати сценарії горизонтального та вертикального масштабування. Алгоритми повинні підтримувати симуляцію навантаження та стрес-тестування для оцінки стійкості системи. Забезпечення інтеграції з популярними інструментами DevOps.

3.2 Вимоги до надійності

Тестові сценарії повинні працювати коректно в умовах різних навантажень і несправностей. Алгоритми повинні фіксувати помилки та надавати звіти для їх аналізу. Надійність оцінки продуктивності повинна бути не менше 95% за основними метриками. Система повинна бути стійкою до збоїв та забезпечувати безперервну роботу під час тестування.

3.3 Умови експлуатації

Система повинна функціонувати в умовах хмарної інфраструктури з наступними параметрами:

- температура середовища від 10 до 35 °C;

- вологість не повинна перевищувати 80% при температурі 25 °C.

Платформа повинна підтримувати роботу на операційних системах Windows, Linux, MacOS. Також необхідне забезпечення стабільного Інтернет з’єднання зі швидкістю передачі даних не менше 100 Мбіт/с.

3.4 Вимоги до складу й параметрів технічних засобів

Наявність серверів із такими характеристиками:

- процесор не менше восьми ядерний;
- оперативна пам’ять не менше 16 ГБ;
- сховище, бажано, SSD від 500 ГБ.

3.5 Вимоги до інформаційної та програмної сумісності

Алгоритми повинні бути сумісними з основними мовами програмування, такими як Java, Python, C#. Забезпечення інтеграції з популярними API для збору даних про продуктивність. Використання форматів даних JSON, XML, CSV для обміну інформації.

3.6 Вимоги до маркування й упаковки

Кожна одиниця програмного забезпечення повинна мати унікальний ідентифікаційний номер. Маркування повинно містити назву продукту, версію програмного забезпечення, дату випуску, QR-код для доступу до документації. Упаковка повинна забезпечувати захист від механічних пошкоджень та впливу вологи.

3.7 Вимоги до транспортування й зберігання

Транспортування має здійснюватися у герметичних контейнерах при температурі 5-40 °C. Зберігання повинно бути організоване в сухих, вентильованих приміщеннях з

температурою 10-25°C. Упаковка має забезпечувати цілісність програмного забезпечення протягом усього строку зберігання.

3.8 Спеціальні вимоги

Алгоритми повинні відповідати стандартам безпеки ISO/IEC 27001 для захисту даних. Забезпечення підтримки багатомовного інтерфейсу для використання в міжнародних проєктах. Надання технічної документації англійською та українською мовами.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Технічне завдання, яке визначає мету, функціональні можливості, вимоги до алгоритмів.

Специфікації, які надають список усіх необхідних документів за вказаними ідентифікаторами.

Текст програми, що забезпечує логіку реалізації для програмістів.

Опис програми містить детальний опис алгоритмів, структур даних та архітектури.

Керівництво користувача, яке містить інструкції з використання програми, опис функціоналу та можливі помилки.

Керівництво розробника, що містить інструкції з підключення необхідних програмних компонентів для вірної роботи алгоритмів.

Спеціальні вимоги, за якими документація створюється в електронному вигляді українською та англійською мовами.

5 ТЕХНІКО-ЕКОНОМІЧНІ ПОКАЗНИКИ

Техніко-економічні показники наведені в таблиці А.1.

Таблиця А.1 – Техніко-економічні показники проєкту

Показник	Характеристика
1	2
Технічні показники	
Продуктивність	Максимальна обробка запитів до 1000 операцій за секунду, час відгуку системи не більше 50 мілісекунд
Масштабованість	Підтримка горизонтального масштабування для збільшення обчислюваної потужності, лінійне зростання продуктивності при додаванні ресурсів із похибкою в 5%
Надійність	Гарантована стійкість до збоїв за рівнем доступності системи не менше 99,9%, автоматичне відновлення роботи протягом двох хвилин
Сумісність	Підтримка інтеграції з DevOps-інструментами та базами даних
Енергоспоживання	Оптимізоване використання ресурсів для зниження енергоспоживання серверів до 20% порівняно зі стандартними сценаріями
Економічні показники	
Зниження витрат на інфраструктуру	Оптимізація використання ресурсів дозволяє зменшити витрати на хмарну інфраструктуру до 15-25%
Зменшення часу на розробку	Використання автоматизованих тестових сценаріїв скорочує час розробки та тестування на 30%
Зменшення витрат на технічне обслуговування	Автоматизація процесів моніторингу та тестування знижує витрати на підтримку системи до 20%
Ефективність впровадження	Швидке впровадження алгоритмів забезпечує повернення інвестицій протягом 6-12 місяців
Соціально-економічні показники	
Підвищення конкурентоспромож-ності	Оптимізація роботи хмарних систем забезпечує кращу якість послуг для кінцевих користувачів
Екологічність	Зниження енергоспоживання та оптимізація ресурсів сприяють скороченню викидів CO_2
Покращення бізнес-процесів	Надійна система забезпечує безперебійну роботу бізнесу та знижує ризики фінансових втрат

6 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Етап 1. Аналіз теоретичних та дослідницьких матеріалів. Містить у собі збір та аналізування вимог, розробку завдання та затвердження виконаної роботи. Термін виконання оцінюється в один місяць.

Етап 2. Проєктування алгоритмів. Містить розробку архітектури у вигляді діаграм. Термін виконання оцінюється в один місяць.

Етап 3. Програмна реалізація алгоритмів. Дослідження та розробка алгоритмів тестування, проєктування відповідних блок-схем. Термін виконання оцінюється в три місяці.

Етап 4. Тестування алгоритмів та їх налагодження. Проведення тестування за обраними метриками. Виправлення несправностей. Термін виконання оцінюється в пів місяця.

Етап 5. Упровадження програмних алгоритмів. Передача програмних алгоритмів та всього пакету замовникові. Навчання робітників. Термін виконання оцінюється в пів місяця.

7 ПОРЯДОК КОНТРОЛЮ І ПРИЙМАННЯ

Контроль і приймання розроблених алгоритмів для тестування продуктивності та масштабованості хмарних систем здійснюються у відповідності до технічного завдання, вимог нормативних документів та погодженої технічної документації.

Вхідний контроль:

- перевірка відповідності технічної документації встановленим стандартам;
- оцінка повноти і коректності вихідних даних;
- контроль наявності необхідних технічних ресурсів для проведення випробувань.

Проміжний контроль:

- перевірка коректності виконання окремих етапів розробки (розробка сценаріїв тестування, моделювання навантажень, інтеграція алгоритмів із системами);
- тестування компонентів у лабораторних умовах із моделюванням реальних сценаріїв;
- аналіз відповідності отриманих проміжних результатів запланованим показникам.

Фінальний контроль:

- проведення комплексних випробувань алгоритмів у реальних умовах експлуатації;
- аналіз отриманих метрик (продуктивність, масштабованість, стійкість до збоїв) та порівняння їх із заданими нормативами;
- виявлення недоліків та їх усунення перед передачею алгоритмів у промислову експлуатацію.

Вимоги до випробувань:

- випробування повинні проводитися відповідно до затверджених програм та методів;

- випробування проводяться у два етапи: лабораторне тестування та тестування в умовах експлуатації.

Умови приймання:

- алгоритми вважаються прийнятими, якщо всі заявлені характеристики (функціональні, технічні та експлуатаційні) відповідають вимогам технічного завдання;

- результати приймання оформлюються актом виконаних робіт, підписаним усіма зацікавленими сторонами.

Під час контролю повинні оформитися всі протоколи випробувань, звіти тестування та акти приймання. Уся технічна документація передається замовнику для подальшого використання та зберігання.

Склад комісії, що здійснює приймання:

- представник замовника або керівник проєкту;

- технічний експерт, який аналізує результати розробки та виконує перевірку алгоритмів на відповідність до вимог;

- керівник розробки, який представляє розроблені алгоритми та пояснює всі нюанси з їх роботою;

- інженер з якості, який контролює відповідність розробки до стандартів якості;

- фахівець із тестування, який контролює коректність проведення тестування;

- секретар комісії, який забезпечує ведення документації, оформлення протоколів випробувань та актів приймання.

Додаток Б

Специфікації

ЗАТВЕРДЖУЮ
Перший проректор Українського
державного університету
науки і технологій
_____ Анатолій РАДКЕВИЧ
20.01.2026

ПРОГРАМА ДЛЯ ДОСЛІДЖЕННЯ МЕТОДІВ ТЕСТУВАННЯ ПРОДУКТИВНОСТІ ТА МАСШТАБОВАНОСТІ В ХМАРНИХ СИСТЕМАХ

Специфікації
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.01555-01-ЛЗ

Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
20.01.2026

Керівник розробки
_____ Тетяна ГРИШЕЧКІНА
20.01.2026

Виконавець
_____ Марія СОЧЕНКО
20.01.2026

Нормоконтролер
_____ Світлана ВОЛКОВА
20.01.2026

2025

ЗАТВЕРДЖЕНО

44165850.01555-01

ПРОГРАМА ДЛЯ ДОСЛІДЖЕННЯ МЕТОДІВ ТЕСТУВАННЯ ПРОДУКТИВНОСТІ ТА МАСШТАБОВАНОСТІ В ХМАРНИХ СИСТЕМАХ

Специфікації

Аркушів 3

2025

ЗМІСТ

1 ДОКУМЕНТАЦІЯ

Документація наведена в таблиці Б.1.

Таблиця Б.1 – Специфікація документації

Позначення	Найменування	Примітка
44165850.01555-01-ЛЗ	Лист затвердження	
44165850.01555-12-ЛЗ	Лист затвердження	
44165850.01555-12	Текст програми	
44165850.01555-01-ЛЗ	Лист затвердження	
44165850.01555-13	Опис програми	
44165850.01555-І3-ЛЗ	Лист затвердження	
44165850.01555-І3	Керівництво користувача	
44165850.01555-33-ЛЗ	Лист затвердження	
44165850.01555-33	Керівництво програміста	

2 КОМПЛЕКСИ

Комплекси наведені в таблиці Б.2

Таблиця Б.2 – Специфікації комплексів

Позначення	Найменування	Примітка
44165850.01555-12	Текст програми	
44165850.01555-13	Опис програми	

3 КОМПОНЕНТИ

Компоненти наведені в таблиці Б.3

Таблиця Б.3 – Специфікації компонентів

Позначення	Найменування	Примітка
44165850.01555-12	Текст програми	
44165850.01555-13	Опис програми	

Додаток В

Текст програми

ЗАТВЕРДЖУЮ
Перший проректор Українського
державного університету
науки і технологій
_____ Анатолій РАДКЕВИЧ
20.01.2026

ПРОГРАМА ДЛЯ ДОСЛІДЖЕННЯ МЕТОДІВ ТЕСТУВАННЯ ПРОДУКТИВНОСТІ ТА МАСШТАБОВАНОСТІ В ХМАРНИХ СИСТЕМАХ
Текст програми
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.01555-12-ЛЗ

Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
20.01.2026

Керівник розробки
_____ Тетяна ГРИШЕЧКІНА
20.01.2026

Виконавець
_____ Марія СОЧЕНКО
20.01.2026

Нормоконтролер
_____ Світлана ВОЛКОВА
20.01.2026

2025

ЗАТВЕРДЖЕНО

44165850.01555-12

ПРОГРАМА ДЛЯ ДОСЛІДЖЕННЯ МЕТОДІВ ТЕСТУВАННЯ ПРОДУКТИВНОСТІ ТА МАСШТАБОВАНОСТІ В ХМАРНИХ СИСТЕМАХ

Текст програми

Аркушів 4

2025

Текст scaling_aws_deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: s3-test-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: s3-test
  template:
    metadata:
      labels:
        app: s3-test
    spec:
      containers:
      - name: s3-test-container
        image: amazon/aws-cli
        command: [ "/bin/sh", "-c", "while true; do aws s3 cp testfile.txt s3://mashabucket/$(date +%s)-testfile.txt; done" ]
        resources:
          requests:
            cpu: "100m"
          limits:
            cpu: "500m"
        env:
        - name: AWS_ACCESS_KEY_ID
          valueFrom:
            secretKeyRef:
              name: aws-secret
              key: access_key
        - name: AWS_SECRET_ACCESS_KEY
          valueFrom:
            secretKeyRef:
              name: aws-secret
              key: secret_key
```

Текст scaling_aws_hpa.yaml

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: s3-test-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: s3-test-app
  minReplicas: 1
  maxReplicas: 5
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```

Текст scaling_gcp_deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gcs-test-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: gcs-test
  template:
    metadata:
      labels:
        app: gcs-test
    spec:
      containers:
      - name: gcs-test-container
        image: google/cloud-sdk
        command: [ "/bin/sh", "-c", "while true; do dd if=/dev/zero of=testfile.txt bs=1M count=10; gsutil cp testfile.txt gs://masha_bucket/$(date +%s)-testfile.txt; done" ]
        resources:
          requests:
            cpu: "100m"
          limits:
            cpu: "500m"
```

Текст scaling_gcp_hpa.yaml

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: gcs-test-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: gcs-test-app
  minReplicas: 1
  maxReplicas: 5
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```

Текст failover_depl_aws_test.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: s3-test-app
spec:
  replicas: 1
  selector:
```

```
    matchLabels:
      app: s3-test
  template:
    metadata:
      labels:
        app: s3-test
    spec:
      containers:
      - name: s3-tester
        image: amazon/aws-cli
        command: ["/bin/sh", "-c"]
        args:
          - "while true; do for i in {1..100}; do aws s3 cp test-file.txt s3://s3-test-bucket/ & done; sleep 1; done"
        env:
        - name: AWS_ACCESS_KEY_ID
          valueFrom:
            secretKeyRef:
              name: aws-credentials
              key: AWS_ACCESS_KEY_ID
        - name: AWS_SECRET_ACCESS_KEY
          valueFrom:
            secretKeyRef:
              name: aws-credentials
              key: AWS_SECRET_ACCESS_KEY
        resources:
          requests:
            cpu: "100m"
          limits:
            cpu: "500m"
```

Текст failover_hpa_aws_test.yaml

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: s3-test-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: s3-test-app
  minReplicas: 1
  maxReplicas: 15
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```

Текст failover_depl_gcp_test.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gcp-storage-app
```

```
  labels:
    app: gcp-storage-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: gcp-storage-app
  template:
    metadata:
      labels:
        app: gcp-storage-app
    spec:
      containers:
      - name: gcp-storage-app
        image: masha_container
        ports:
        - containerPort: 8080
        resources:
          requests:
            cpu: 200m
            memory: 128Mi
          limits:
            cpu: 1
            memory: 256Mi
        env:
        - name: GCP_BUCKET_NAME
          value: "masha_bucket"
        - name: GOOGLE_APPLICATION_CREDENTIALS
          value: "Desktop/hip-weaver-447911-n2-d7450865ab68.json"
        volumeMounts:
        - name: secret-volume
          mountPath: /etc/secret-volume
          readOnly: true
      volumes:
      - name: secret-volume
        secret:
          secretName: gcp-service-account-key  # Название секрета с ключом
```

Текст failover_hpa_gcp_test.yaml

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: gcp-storage-hpa
  namespace: default
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: gcp-storage-app
  minReplicas: 3
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization       averageUtilization: 50
```

Додаток Г

Опис програми

ЗАТВЕРДЖУЮ
Перший проректор Українського
державного університету
науки і технологій
_____ Анатолій РАДКЕВИЧ
20.01.2026

ПРОГРАМА ДЛЯ ДОСЛІДЖЕННЯ МЕТОДІВ ТЕСТУВАННЯ ПРОДУКТИВНОСТІ ТА МАСШТАБОВАНОСТІ В ХМАРНИХ СИСТЕМАХ

Опис програми
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.01555-13-ЛЗ

Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
20.01.2026

Керівник розробки
_____ Тетяна ГРИШЕЧКІНА
20.01.2026

Виконавець
_____ Марія СОЧЕНКО
20.01.2026

Нормоконтролер
_____ Світлана ВОЛКОВА
20.01.2026

2025

ЗАТВЕРДЖЕНО

44165850.01555-13

ПРОГРАМА ДЛЯ ДОСЛІДЖЕННЯ МЕТОДІВ ТЕСТУВАННЯ ПРОДУКТИВНОСТІ ТА МАСШТАБОВАНОСТІ В ХМАРНИХ СИСТЕМАХ

Опис програми

Аркушів 7

2025

ЗМІСТ

1 ЗАГАЛЬНІ ВІДОМОСТІ

Розроблена програма представляє собою набір конфігураційних файлів для оркестрації контейнеризованих додатків у хмарному середовищі. Вона призначена для автоматизації тестування продуктивності та масштабованості інфраструктури, розгорнутої на платформах AWS та GCP.

2 ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Програма забезпечує:

- автоматичне розгортання додатків у середовищі Kubernetes;
- динамічне масштабування за рівнем навантаження;
- автоматичне відновлення у разі відмови компонентів;
- налаштування тестових сценаріїв для вимірювання продуктивності системи.

3 ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Програма складається з таких основних конфігураційних файлів:

- файли розгортання (Deployment) – визначають параметри запуску контейнеризованих додатків, зокрема використання ресурсів процесору та пам’яті;

- файли авто-масштабування (HorizontalPodAutoscaler, HPA) – забезпечують автоматичне збільшення або зменшення кількості реплік залежно від навантаження;

- файли конфігурації середовища (Secrets, ConfigMaps) – містять ключі доступу та параметри взаємодії з хмарними сервісами;

- скрипти автоматизації – команди для виконання стрес-тестів шляхом моделювання навантаження.

4 ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Технічними засобами є:

- Kubernetes – система оркестрації контейнерів;
- AWS (S3, EC2) та Google Cloud (GCP Storage) – хмарні платформи для зберігання та обчислень;
- Docker – контейнеризація додатків;
- Prometheus/Grafana – моніторинг продуктивності.

5 ВИКЛИК І ЗАВАТАЖЕННЯ

Програма завантажується у середовищі Kubernetes шляхом застосування конфігураційних файлів: kubectl apply -f deployment.yaml та kubectl apply -f hpa.yaml.

Для AWS та GCP використовуються відповідні команди CLI.

6 ВХІДНІ ДАНІ

Вхідними даними є:

- кількість початкових та максимальних реплік сервісу;
- обмеження використання процесору та пам’яті;
- тестові файли для завантаження у хмарне сховище.

7 ВИХІДНІ ДАНІ

Вихідними файлами є:

- логи виконання тестів;
- графіки завантаження ресурсів;
- звіти про ефективність масштабування.

Додаток Д

Керівництво користувача

ЗАТВЕРДЖУЮ
Перший проректор Українського
державного університету
науки і технологій
_____ Анатолій РАДКЕВИЧ
20.01.2026

ПРОГРАМА ДЛЯ ДОСЛІДЖЕННЯ МЕТОДІВ ТЕСТУВАННЯ ПРОДУКТИВНОСТІ ТА МАСШТАБОВАНОСТІ В ХМАРНИХ СИСТЕМАХ

Керівництво користувача
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.01555-І3-ЛЗ

Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
20.01.2026

Керівник розробки
_____ Тетяна ГРИШЕЧКІНА
20.01.2026

Виконавець
_____ Марія СОЧЕНКО
20.01.2026

Нормоконтролер
_____ Світлана ВОЛКОВА
20.01.2026

2025

ЗАТВЕРДЖЕНО

44165850.01555-І3

ПРОГРАМА ДЛЯ ДОСЛІДЖЕННЯ МЕТОДІВ ТЕСТУВАННЯ ПРОДУКТИВНОСТІ ТА МАСШТАБОВАНОСТІ В ХМАРНИХ СИСТЕМАХ

Керівництво користувача

Аркушів 11

2025

ЗМІСТ

ВСТУП

Дана інструкція користувача призначена для забезпечення детальних покрокових рекомендацій щодо встановлення, налаштування та використання інструментів для тестування продуктивності та масштабованості хмарних сховищ. Вона охоплює інструкції для встановлення такого ПЗ, як Apache JMeter та Kubernetes HPA.

Метою даної інструкції є:

- спрощення процесу налаштування інструментів для користувачів-початківців;
- забезпечення точних технічних рекомендацій для досвідчених користувачів;
- оптимізація роботи з інструментами шляхом надання перевірених практик і команд.

Інструкція структурована таким чином, щоб бути максимально зрозумілою для користувачів різного рівня технічної підготовки.

1 ПРИЗНАЧЕННЯ ТА УМОВИ ЗАСТОСУВАННЯ

Інструкція користувача призначена для забезпечення підтримки фахівців у сфері тестування ПЗ, адміністраторів систем та розробників, які здійснюють:

- тестування продуктивності та навантаження таким інструментом як Apache JMeter;
- масштабування хмар таким інструментом як Kubernetes HPA;
- стрес-тестування хмар таким інструментом як Apache JMeter;
- симуляція збоїв та перевірка відмовостійкості таким інструментом як Kubernetes HPA.

Системними вимогами є:

- наявність операційної системи macOS або іншої ОС із сумісними інструментами;
- установлені основні утиліти: Java, Docker, Minikube або інші, залежно від потреб, що вказані в даній інструкції.

2 ПІДГОТОВКА ДО РОБОТИ

Дана інструкція буде вказана для інсталювання ПЗ на macOS. Перш ніж розпочати використання інструментів, зазначених у цій інструкції, необхідно виконати кілька важливих підготовчих кроків, які забезпечать коректність їхньої роботи та ефективність тестування.

Першим кроком є перевірка системних вимог:

- операційна система: macOS або інша із підтримкою необхідних утиліт;
- об’єм пам’яті: щонайменше 8 ГБ оперативної пам’яті;
- процесор: мульти-ядерний процесор (від 4х ядер);
- стабільне Інтернет-з’єднання для завантаження необхідних компонентів.

Установлення необхідного ПЗ: Java (версія 8 або вище), Docker, Maven, kubectl та інші доповнення, що вказані в даній інструкції.

3 ОПИС ОПЕРАЦІЙ

3.1 Інсталювання Apache JMeter

Apache JMeter потребує Java для роботи. Для цього можна встановити Java через Homebrew в терміналі за командою: brew install openjdk, як показано на рисунку Д.1.

Рисунок Д.1 – Установлення Java на macOS

Після того, як Java буде встановлена, можна встановити Apache JMeter за допомогою Homebrew. У терміналі потрібно виконати команду: brew install jmeter, як показано на рисунку Д.2.

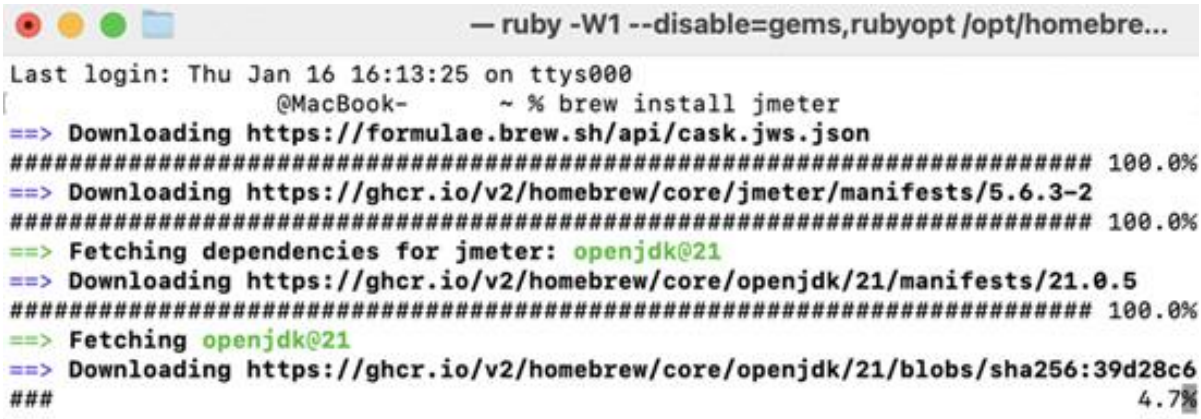


Рисунок Д.2 – Установлення Apache JMeter на macOS

Після завершення установки можна запустити Apache JMeter за допомогою команди: jmeter в терміналі. Головне вікно програми наведено на рисунку Д.3.

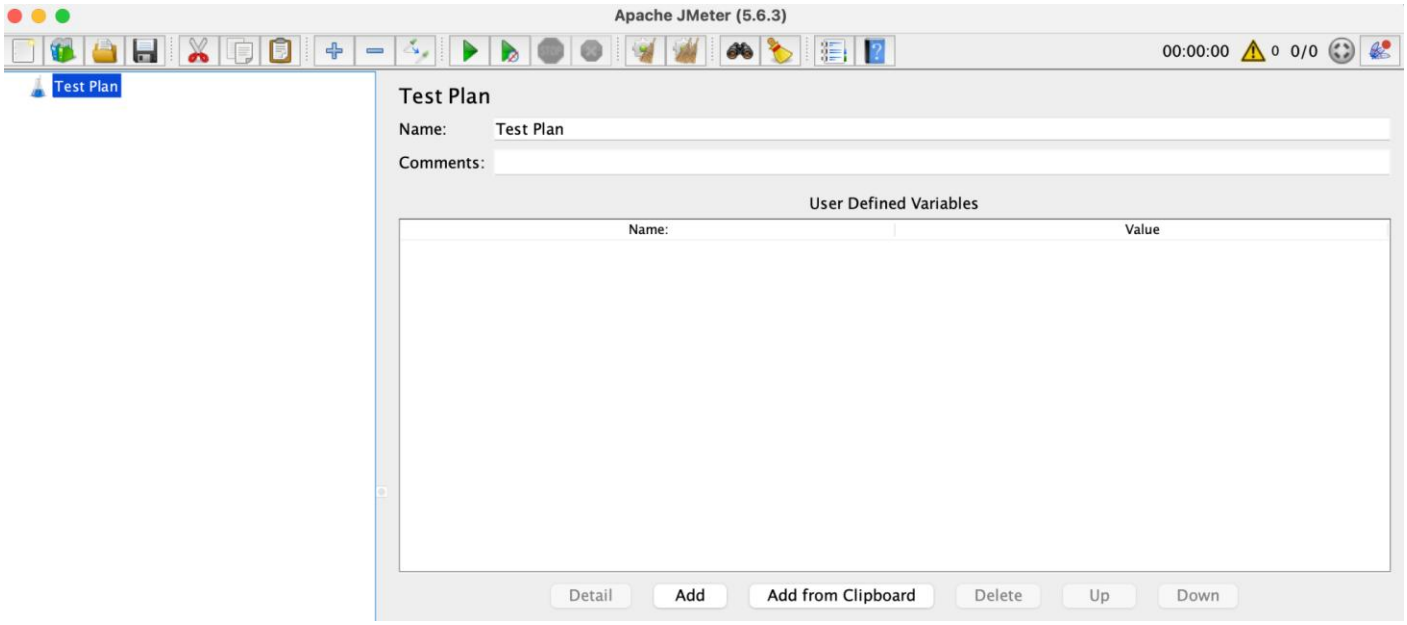


Рисунок Д.3 – Установлена Apache JMeter на macOS

За необхідності можна встановити додаткові плагіни для роботи із звітністю. Для цього потрібно перейти для плагін-менеджеру, який знаходиться в налаштуваннях. Наприклад, для покращення аналізування результатів тестувань були інстальовані такі плагіни, як результат у графах, додаткові вигляди звітів, таймер, створення кастомних груп потоків для тестувань і сценаріїв тощо.

Такі додаткові налаштування допоможуть більш глибоко та точно проаналізувати роботу хмарних сховищ, їх можливості до тестування навантаження та реагування на стрес-тестування.

3.2 Інсталювання Kubernetes HPA

Спочатку необхідно інсталювати Docker для macOS, завантаживши інсталятор із офіційного вебсайту. Установлена програма наведена на рисунку Д.4.

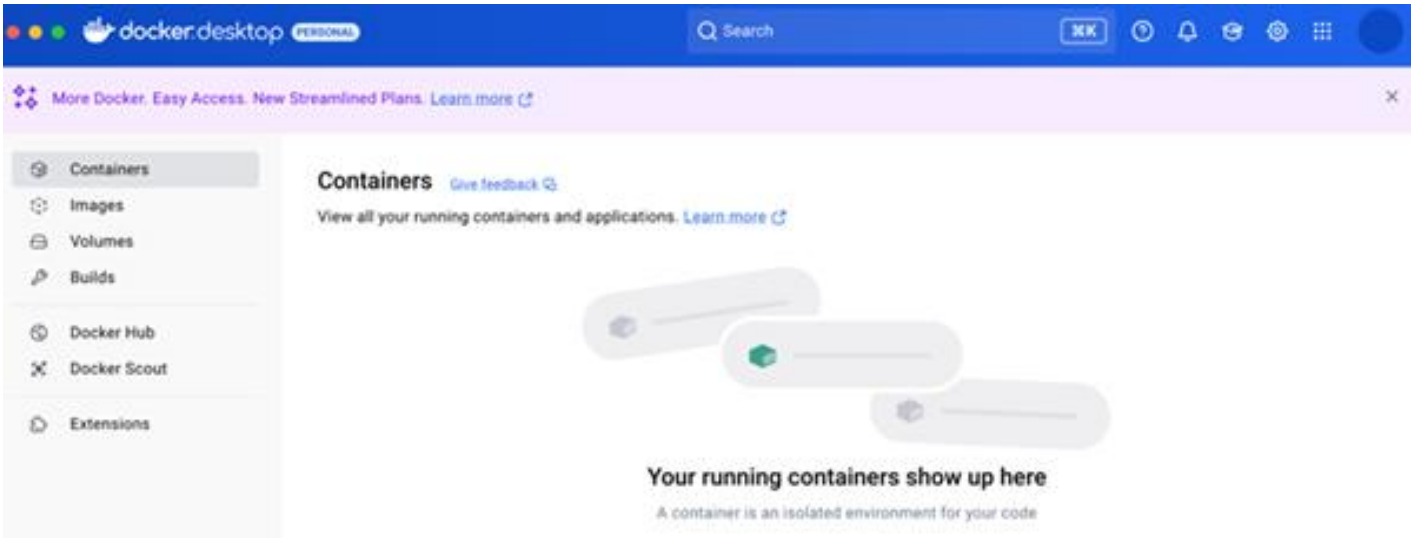


Рисунок Д.4 – Установлена Docker на macOS

Далі потрібно встановити kubectl для взаємодії Kubernetes HPA з кластером за такою командою в терміналі: brew install kubectl, як показано на рисунку Д.5.

Рисунок Д.5 – Процес інсталювання kubectl на macOS

Далі треба встановити Minikube, який дозволяє запускати локальний кластер на macOS за допомогою команди: brew install minikube, як наведено на рисунку Д.6.

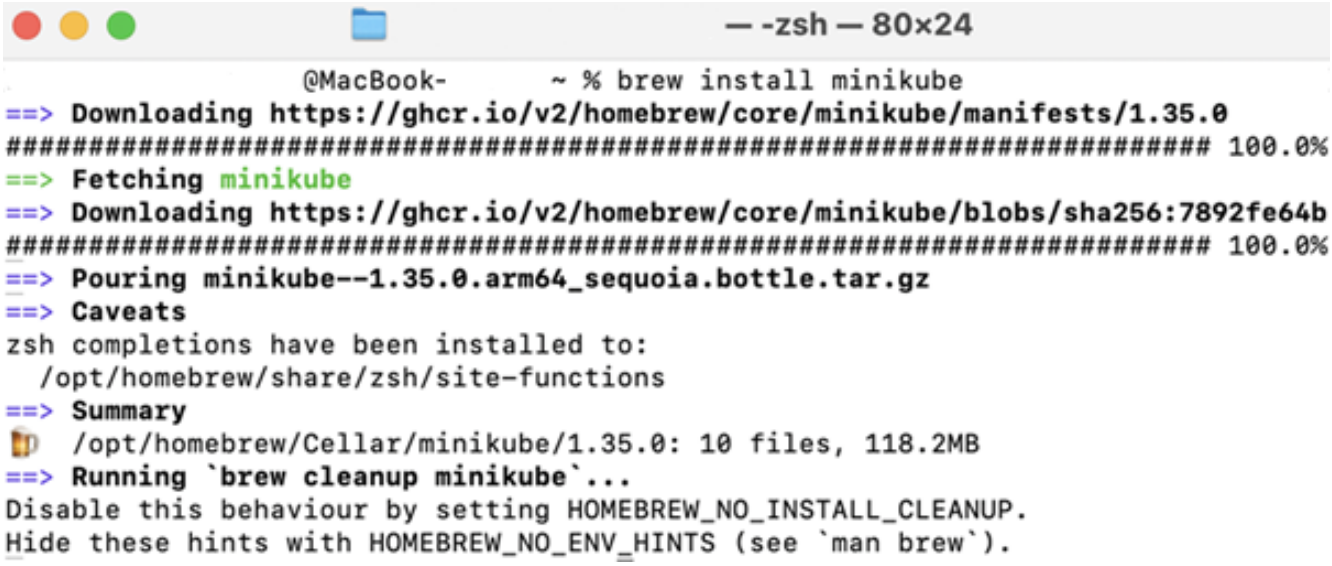


Рисунок Д.6 – Процес інсталювання Minikube на macOS

Далі, потрібно запустити Minikube для створення локального кластеру та його перевірки за командою: minikube start, як показано на рисунку Д.7.

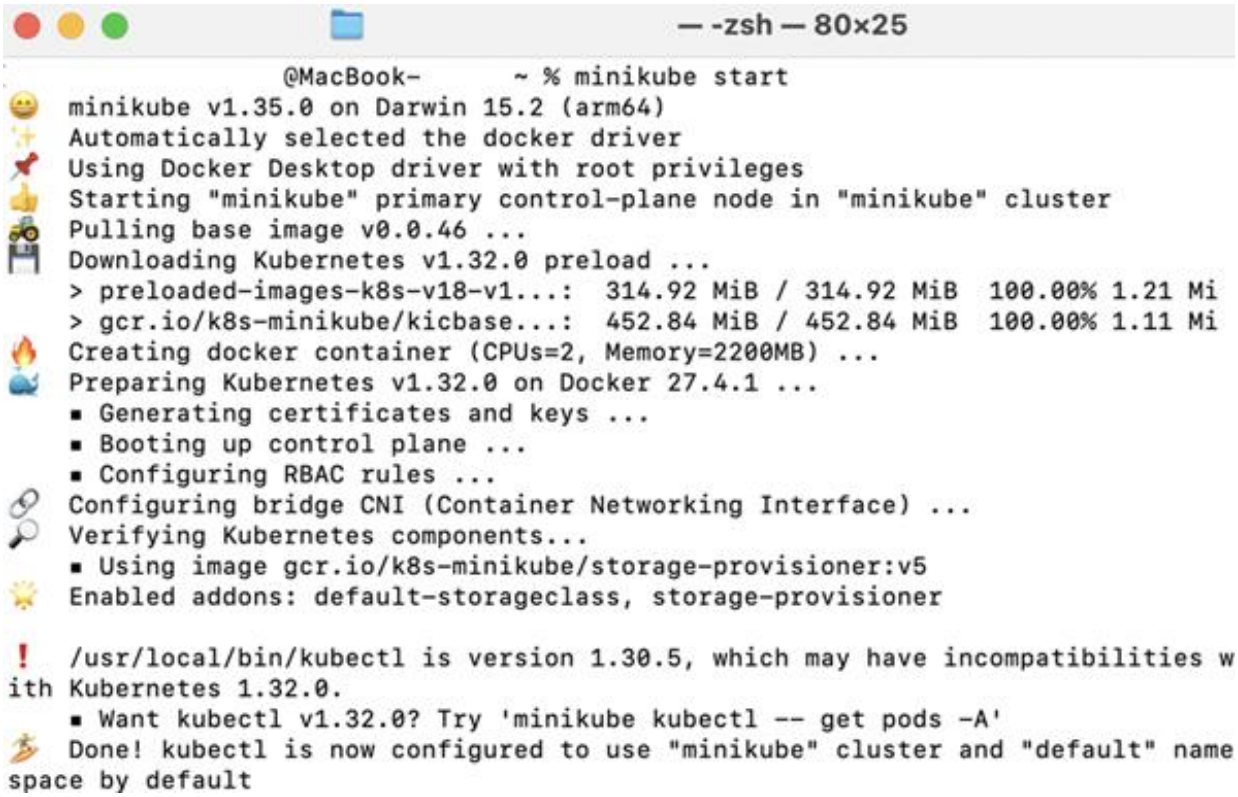


Рисунок Д.7 – Запуск Minikube на macOS

Далі потрібно ввімкнути HPA. Якщо його немає, потрібно встановити за командою: kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml.

Для перевірки Metrics Server: kubectl get deployment metrics-server -n kube-system, як наведено на рисунку Д.8.

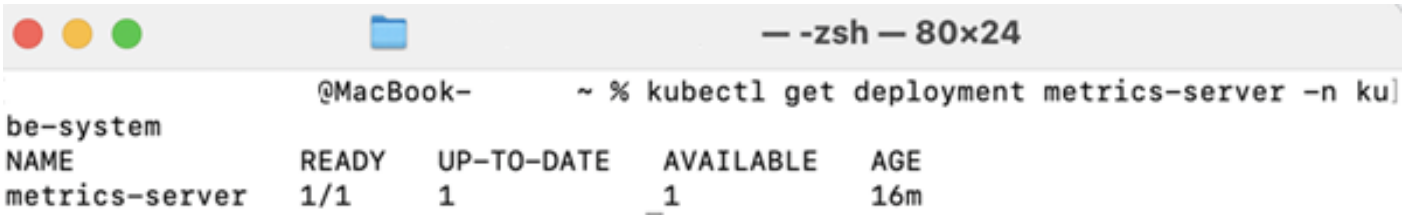


Рисунок Д.8 – Перевірка Metrics Server на macOS

Далі потрібно встановити початкові налаштування, щоб перевірити працездатність HPA. Налаштування здійснюються за командою: kubectl autoscale deployment <ім'я_розгортання> --cpu-percent=50 --min=1 --max=10. Для перевірки роботи використовувати команду: kubectl get hpa, як показано на рисунку Д.9.

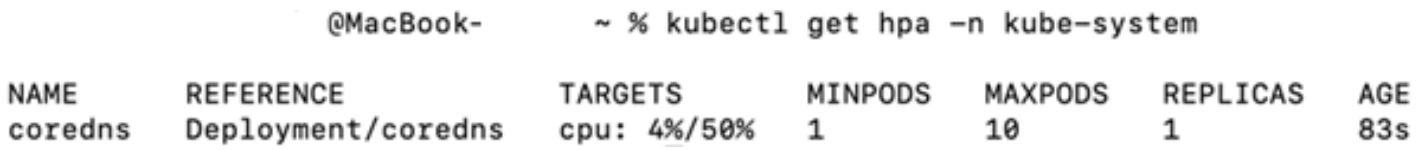


Рисунок Д.9 – Перевірка працездатності HPA на macOS

Додатково можна інсталювати інструменти Grafana та Prometheus для моніторингу результатів та спрощеного аналізу тестування масштабованості та відмовостійкості.

4 АВАРІЙНІ СИТУАЦІЇ

У процесі роботи з інструментами можуть виникати аварійні ситуації або помилки.

Перша причина – інструмент не запускається. Можливими причинами можуть бути:

- відсутність необхідного програмного забезпечення (Java, Docker тощо);
- некоректно встановлені або пошкоджені файли інструменту;
- невірний шлях до виконуваного файлу.

Способами усунення є повторення тих же команд та встановлення шляхів до інструментів змінної PATH.

Друга причина – тестування зависає або виникає помилка. Можливими причинами є:

- нестача ресурсів (пам’ять, процесор);
- некоректно налаштований тестовий сценарій;
- проблеми зі з’єднанням із сервером або базою даних.

Способами усунення є такі варіанти:

- перевірка системних ресурсів за допомогою моніторингу (наприклад, Activity Monitor);
- оптимізація тестових сценаріїв, зменшивши кількість потоків або обсяг навантаження;
- переконання, що цільовий сервер доступний: ping <IP-адреса_сервера>;
- перевірка лог-файли тесту для діагностики.

Третя можлива помилка – Docker-контейнер не запускається. Можливі причини:

- Docker не працює або не встановлено;
- помилка в Dockerfile або docker-compose.yml;
- недостатньо ресурсів для запуску контейнера.

Способами усунення є наступні варіанти:

- переконання, що Docker запущений: docker ps;
- перевірка валідності Dockerfile: docker build;
- перезапуск Docker і звільнення ресурсів.

Четвертою можливою помилкою є помилки в Kubernetes. Можливі причини:

- неправильна конфігурація кластеру;
- відсутність необхідних компонентів (наприклад, Metrics Server).

Способами усунення є:

- перевірити статус кластера: kubectl get nodes;
- увімкнення відсутніх компонентів (наприклад, Metrics Server): kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml;
- перевірити лог-файли подів: kubectl logs <ім'я_пода>.

5 РЕКОМЕНДАЦІЇ ЩОДО ЗАСВОЄННЯ

Рекомендаціями щодо засвоєння є такі:

- передбачається вивчення документації інструментів, таких як JMeter, Kubernetes, на офіційних сайтах;

- рекомендується виконувати інструкції на основі реальних проєктів для глибшого розуміння функціональності;

- рекомендується створювати тестові сценарії, поступово збільшуючи їх складність;

- пропонується використовувати відеоуроки та статті для додаткового вивчення;

- із метою поступового освоєння інструментів рекомендовано починати з простих конфігурацій;

- передбачається детальний розбір результатів тестування та помилок для вдосконалення навичок.

Додаток Е

Керівництво програміста

ЗАТВЕРДЖУЮ
Перший проректор Українського
державного університету
науки і технологій
_____ Анатолій РАДКЕВИЧ
20.01.2026

ПРОГРАМА ДЛЯ ДОСЛІДЖЕННЯ МЕТОДІВ ТЕСТУВАННЯ ПРОДУКТИВНОСТІ ТА МАСШТАБОВАНОСТІ В ХМАРНИХ СИСТЕМАХ

Керівництво програміста
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.01555-33-ЛЗ

Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
20.01.2026

Керівник розробки
_____ Тетяна ГРИШЕЧКІНА
20.01.2026

Виконавець
_____ Марія СОЧЕНКО
20.01.2026

Нормоконтролер
_____ Світлана ВОЛКОВА
20.01.2026

2025

ЗАТВЕРДЖЕНО

44165850.01555-33

ПРОГРАМА ДЛЯ ДОСЛІДЖЕННЯ МЕТОДІВ ТЕСТУВАННЯ ПРОДУКТИВНОСТІ ТА МАСШТАБОВАНОСТІ В ХМАРНИХ СИСТЕМАХ

Керівництво програміста

Аркушів 7

2025

ЗМІСТ

1 ПРИЗНАЧЕННЯ Й УМОВИ ЗАСТОСУВАННЯ ПРОГРАМИ

Програма призначена для автоматизації тестування продуктивності та масштабованості хмарних сервісів, розгорнутих у середовищах AWS, GCP, Apache JMeter та Kubernetes. Вони використовуються для оцінки швидкодії хмарних сховищ, аналізу поведінки сервісів під навантаженням та перевірки механізмів автомасштабування.

Програми можуть застосовуватися для:

- тестування продуктивності хмарних сховищ (S3, GCP Storage) через Apache JMeter;
- аналізу здатності систем до масштабування за допомогою Kubernetes HPA;
- моделювання навантаження та стрес-тестування сервісів;
- оцінки відмовостійкості в умовах пікових навантажень.

2 ХАРАКТЕРИСТИКИ ПРОГРАМИ

Програма складається з таких компонентів:

- конфігураційні файли Kubernetes (deployment.yaml, hpa.yaml) – визначають параметри розгортання додатків;

- тестові сценарії Apache JMeter (jmeter_test.jmx) – виконують навантажувальне тестування хмарних сховищ;

- інструменти моніторингу (Prometheus, Grafana) – аналізують продуктивність системи;

- CLI-утиліти (kubectl, aws-cli, gcloud) – використовуються для управління кластером та хмарними ресурсами.

3 ЗВЕРНЕННЯ ДО ПРОГРАМИ

3.1 Тестування в Apache JMeter

Для створення тестового плану в JMeter необхідно слідувати наступним крокам:

- відкрити JMeter;
- додати Thread Group (Test Plan → Add → Threads (Users) → Thread Group), як наведено на рисунку Е.1;
- установити: Number of Threads (Users): 50, Ramp-Up Period: 10 сек, Loop Count: Forever або інші параметри на розсуд спеціаліста, як наведено на рисунку Е.1;
- додати HTTP Request: Protocol: https, Server Name: my-test-bucket.s3.amazonaws.com (для AWS) або storage.googleapis.com (для GCP), Method: GET, Path: <найменування файлу із сховища>, як наведено на рисунку Е.2;
- додати ліснерів для перегляду результатів тестування;
- запустити тест та переглянути результати.

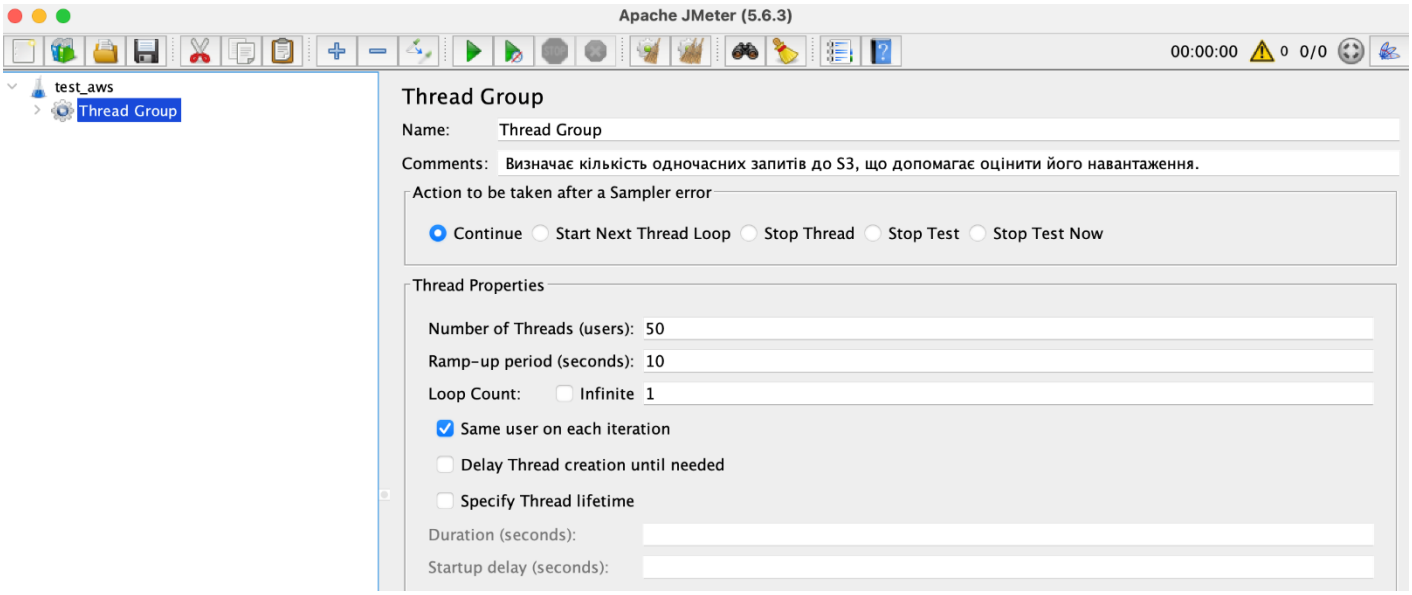


Рисунок Е.1 – Додавання тестового сценарію для тестування в Apache JMeter

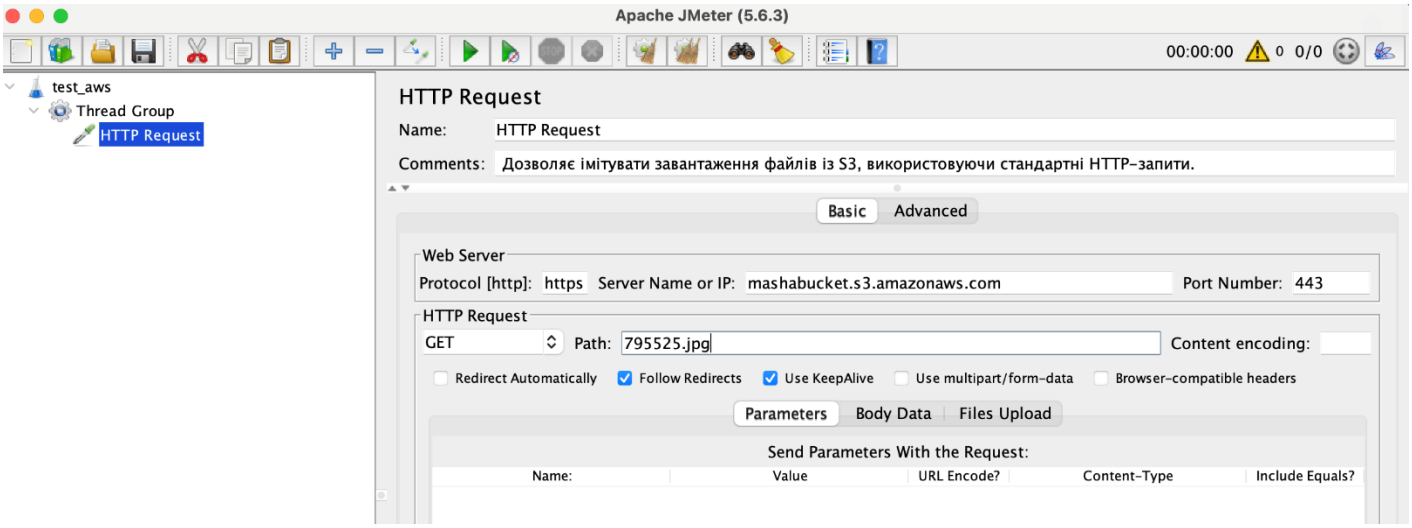


Рисунок Е.2 – Додавання запиту тестового сценарію для тестування в Apache JMeter

3.2 Тестування в Kubernetes HPA

Тестування за допомогою Kubernetes HPA виконується за наступними кроками:

- запуск Kubernetes-кластера за допомогою команд minikube start та minikube addons enable metrics-server, як наведено на рисунку Е.3;

- розгорнути сервіс за допомогою kubectl apply -f deployment.yaml та kubectl apply -f hpa.yaml – розпочнеться масштабування, як наведено на рисунку Е.4;

- аналіз результатів: kubectl top pods та kubectl get hpa, як наведено на рисунку Е.4.

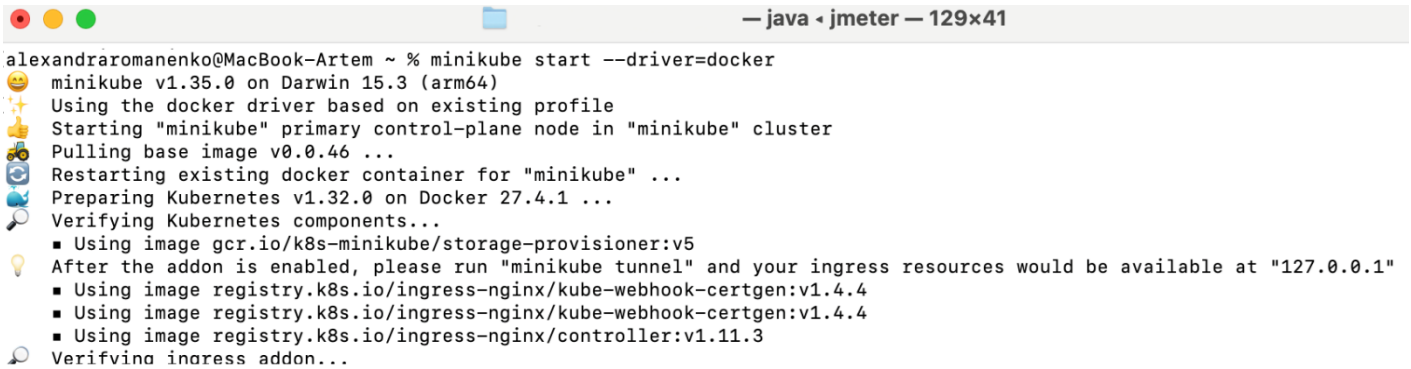


Рисунок Е.3 – Розгортання minikube в Docker для Kubernetes HPA

```
Last login: Tue Feb  4 11:42:17 on ttys000
                @MacBook-      ~ % kubectl apply -f Desktop/deployment.yaml
deployment.apps/s3-test-app configured
                @MacBook-      ~ % kubectl get pods
NAME                           READY   STATUS        RESTARTS   AGE
s3-test-app-59d9779fd7-46rcz   1/1     Running       0          5s
s3-test-app-7b7b685484-2k8xz   1/1     Terminating   0          5m15s
s3-test-app-7b7b685484-5mjt8   1/1     Terminating   0          10m
s3-test-app-7b7b685484-njlc9   1/1     Terminating   0          5m
s3-test-app-7b7b685484-sghdc   1/1     Terminating   0          5m15s
s3-test-app-7b7b685484-v59f9   1/1     Terminating   0          5m15s
                @MacBook-      ~ % kubectl apply -f Desktop/hpa.yaml
horizontalpodautoscaler.autoscaling/s3-test-hpa unchanged
                @MacBook-      ~ % kubectl apply -f Desktop/deployment.yaml
deployment.apps/s3-test-app configured
                @MacBook-      ~ % kubectl get hpa
NAME          REFERENCE                 TARGETS              MINPODS   MAXPODS   REPLICAS   AGE
s3-test-hpa   Deployment/s3-test-app   cpu: <unknown>/50%   1         5         1          10m
                @MacBook-      ~ % kubectl top pods
NAME                           CPU(cores)   MEMORY(bytes)
s3-test-app-648c7f6587-9wbfg   475m         13Mi
                @MacBook-      ~ % kubectl get pods -n kube-system | grep metrics-server
metrics-server-7fbb699795-zlclr    1/1     Running   0             15m
                @MacBook-      ~ % kubectl apply -f Desktop/deployment.yaml
deployment.apps/s3-test-app configured
                @MacBook-      ~ % kubectl get hpa
NAME          REFERENCE                 TARGETS         MINPODS   MAXPODS   REPLICAS   AGE
s3-test-hpa   Deployment/s3-test-app   cpu: 403%/50%   1         5         1          12m
```

Рисунок Е.4 – Виконання масштабування за допомогою Kubernetes HPA

4 ВХІДНІ Й ВИХІДНІ ДАНІ

Вхідні дані:

1) конфігурація JMeter (jmeter_test.jmx):

- users – кількість одночасних користувачів;
- ramp-up – час поступового навантаження;
- loop – кількість повторень тесту;

2) параметри Kubernetes Deployment (deployment.yaml):

- replicas – кількість початкових реплік;
- resources.limits.cpu/memory – обмеження використання ресурсів.

Вихідні дані:

- час відгуку сервера (Response Time);
- максимальне навантаження (Requests Per Second, RPS);
- використання ресурсів (CPU, RAM);
- динаміка масштабування HPA (kubectl get hpa).

5 ПОВІДОМЛЕННЯ

Помилками в JMeter можуть бути:

- java.net.SocketException: Connection reset – сервер не витримує навантаження, а рішення: зменшити users у JMeter;

- 429 Too Many Requests – обмеження трафіку AWS/GCP, а рішення: збільшити таймаут у jmeter_test.jmx.

Помилками в Kubernetes можуть бути:

- No metrics available – HPA не отримує дані, а рішення: перевірити Metrics Server: kubectl get deployment metrics-server -n kube-system

- Insufficient CPU – брак ресурсів, а рішення: збільшити resources.requests.cpu у deployment.yaml.

Додаток Ж

Тези конференції

Дослідження методів тестування продуктивності та масштабованості в хмарних системах

Соченко М.О., Гришечкіна Т.С., Український державний університет науки та технологій, Україна

У сучасних умовах стрімкого розвитку інформаційних технологій хмарні системи стали основою для побудови програмних рішень у різних галузях - від електронної комерції та фінансового сектору до освіти й державного управління. Використання хмарних платформ забезпечує гнучкість, доступність ресурсів і можливість динамічного масштабування. Водночас зростання кількості користувачів і навантажень висуває підвищені вимоги до продуктивності, масштабованості та надійності таких систем. Це зумовлює актуальність досліджень, спрямованих на вдосконалення методів тестування хмарних рішень.

Актуальність даної роботи полягає у необхідності комплексної оцінки ефективності хмарних систем в умовах змінного та пікового навантаження. Недостатньо протестовані системи можуть призводити до зниження якості сервісів, втрати даних та фінансових збитків. Тому застосування сучасних методів навантажувального тестування, тестування масштабованості, стрес-тестування та тестування відмовостійкості є критично важливим етапом життєвого циклу програмного забезпечення.

Об’єктом дослідження є хмарні системи та їх характеристики продуктивності і масштабованості. Предметом дослідження є методи тестування продуктивності та масштабованості в хмарних середовищах. Метою роботи є дослідження та аналіз сучасних підходів до тестування хмарних систем, розробка експериментальних сценаріїв і визначення ефективних способів оптимізації їх роботи.

У роботі проаналізовано сучасні тенденції розвитку хмарних технологій, зокрема автоматизацію тестування, контейнеризацію, використання оркестрації та механізмів автоматичного масштабування. Особливу увагу приділено дослідженню основних видів тестування: навантажувального тестування, яке дозволяє оцінити стабільність системи при типовому навантаженні; тестування масштабованості, спрямованого на аналіз ефективності використання додаткових ресурсів; стрес-тестування, що визначає граничні можливості системи; а також тестування відмовостійкості, яке перевіряє здатність системи відновлювати роботу після збоїв.

У практичній частині роботи розроблено тестові сценарії та алгоритми тестування продуктивності й масштабованості хмарних систем. Для проведення експериментів використано сучасні інструментальні засоби, зокрема Apache JMeter, K6, Locust, а також засоби моніторингу AWS CloudWatch і Google Cloud Monitoring. Проведено аналіз ключових метрик ефективності, таких як час відгуку, пропускна здатність, використання ресурсів, еластичність і лінійність масштабування.

За результатами дослідження виконано порівняльний аналіз методів тестування та сформульовано рекомендації щодо оптимізації хмарних систем. Отримані результати підтверджують, що використання автоматизованого тестування у поєднанні з контейнеризацією та механізмами автоматичного масштабування дозволяє підвищити стабільність і продуктивність хмарних сервісів, а також зменшити кількість помилок при високих навантаженнях.

Практична цінність роботи полягає у можливості використання запропонованих методик і тестових сценаріїв під час проєктування та експлуатації реальних хмарних систем з метою підвищення їх надійності та ефективності.