

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Інформаційних управляючих систем
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти другий (магістерський)
Дослідження моделей управління вартістю ІТ-проектів з розробки інтернет
магазинів комп'ютерних ігор при використанні безсерверних
технологій
(тема)

Виконав:

здобувач 2 року навчання,
групи УПГІТМ-24-1

Євгеній МАТЮШКО

(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

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

Освітня програма Управління проектами в
галузі інформаційних технологій

Керівник доц. каф. ІУС Олена МІХНОВА
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ІУС



(підпис)

Костянтин ПЕТРОВ

(власне ім'я, прізвище)

2026 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
 Кафедра Інформаційних управляючих систем
 Рівень вищої освіти другий (магістерський)
 Спеціальність 122 Комп'ютерні науки
 (код і повна назва)
 Тип програми освітньо-наукова
 (освітньо-професійна або освітньо-наукова)
 Освітня програма Управління проектами в галузі інформаційних технологій
 (повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри


 (підпис)

« 06 » квітня 20 26 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Матюшку Євгенію Дмитровичу
 (прізвище, ім'я, по батькові)

1. Тема роботи Дослідження моделей управління вартістю ІТ-проектів з розробки інтернет магазинів комп'ютерних ігор при використанні безсерверних технологій

затверджена наказом університету від 23 березня 2026 р. № 231Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 20 травня 2026 р.

3. Вихідні дані до роботи Матеріали науково-дослідної практики, офіційна документація хмарного провайдера Microsoft Azure, дані навантажувального тестування Grafana k6, огляд науково-технічних джерел щодо методів управління вартістю у безсерверних архітектурах та забезпечення угод про рівень обслуговування

4. Перелік питань, що потрібно опрацювати в роботі Аналіз проблематики ціноутворення та управління ресурсами у безсерверних інфраструктурах. Формалізація математичної SLA-орієнтованої моделі оптимізації витрат з урахуванням фінансових ризиків. Проектування архітектури та алгоритму роботи процедури управління вартістю для інтернет-магазину. Практичне випробування розробленої моделі шляхом навантажувального тестування мікросервісу та розрахунку сукупної вартості володіння

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної області та існуючих підходів до оптимізації витрат на хмарний хостинг	06.04.2026	Виконано
2	Постановка мети і завдань дослідження	09.04.2026 – 11.04.2026	Виконано
3	Дослідження теоретично-методологічних основ управління вартістю Serverless інфраструктури	12.04.2026 – 17.04.2026	Виконано
4	Розробка та впровадження процедури управління вартістю безсерверного інтернет-магазину комп'ютерних ігор	19.04.2026 – 25.04.2026	Виконано
5	Практична реалізація моделі управління вартістю, проведення тестування та аналіз результатів	26.04.2026 – 30.04.2026	Виконано
6	Оформлення пояснювальної записки	05.04.2026 – 06.04.2026	Виконано
7	Оформлення графічної частини та презентаційних матеріалів	07.05.2026 – 10.05.2026	Виконано
8	Представлення на рецензування	13.05.2026	Виконано
9	Проходження перевірки на плагіат та нормоконтроль	15.05.2026	Виконано
10	Подання кваліфікаційної роботи на розгляд ЕК	20.05.2026	Виконано

Дата видачі завдання 06 квітня 2026 р.

Здобувач _____


(підпис)

Керівник роботи _____


(підпис)

доц. каф. ІУС Олена МІХНОВА
(посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 75с., 38 рис., 3табл., 27 джерел.

AZURE, FINOPS, SERVERLESS, SLA, ВАРТІСТЬ ІНФРАСТРУКТУРИ, ЕЛЕКТРОННА КОМЕРЦІЯ, ОПТИМІЗАЦІЯ ВИТРАТ, ХМАРНІ ОБЧИСЛЕННЯ.

Об'єктом дослідження кваліфікаційної роботи є процеси управління вартістю у безсерверних архітектурах для систем електронної комерції.

Метою роботи є підвищення економічної ефективності функціонування інтернет-магазину шляхом розроблення та впровадження моделі управління вартістю на основі гібридної моделі, орієнтованої на SLA.

Методи дослідження – навантажувальне тестування за допомогою інструментарію Grafana k6, математичне моделювання сукупної вартості володіння, статистичний аналіз хмарної телеметрії.

У ході виконання роботи розроблено процедуру оптимізації витрат, яка, на відміну від класичних підходів, інтегрує показники швидкодії та фінансові ризики від порушення SLA в єдину цільову функцію мінімізації сукупних витрат. Визначено вплив обсягу оперативної пам'яті на швидкість та витрати в Azure Container App, встановлено оптимальні параметри мінімізації збитків.

На основі результатів виконаних досліджень розроблено модель управління вартістю безсерверного інтернет-магазину комп'ютерних ігор. Упровадження розробленої процедури уможливорює проактивне управління бюджетами високонавантажених систем на наявних хмарних потужностях, усуває явища надлишкового та недостатнього забезпечення ресурсами та підвищує рентабельність ІТ-проектів у межах методології FinOps.

ABSTRACT

The explanatory note of the qualification thesis: 75 pages, 38 figures, 3 tables, 27 sources.

AZURE, FINOPS, SERVERLESS, SLA, INFRASTRUCTURE COST, E-COMMERCE, COST OPTIMIZATION, CLOUD COMPUTING.

The object of the study is the process of cost management and dynamic scaling of computing resources in serverless architectures for e-commerce systems.

The aim of the thesis is to improve the economic efficiency of online stores functioning by developing and implementing a cost management model based on a hybrid SLA-oriented model.

Research methods – load testing using Grafana k6 toolset, mathematic modeling of the total cost of ownership, statistical analysis of telemetry.

In the course of the work, a cost optimization procedure was developed, which, unlike classical approaches, integrates performance metrics and financial risks from SLA violations into a single objective function for minimizing total costs.

The impact of RAM capacity on performance speed and costs in Azure Container Apps was determined, and optimal parameters for minimizing losses were established.

Based on results of the conducted research, a cost management model for a serverless online computer game store was developed.

The implementation of the developed procedure enables proactive budget management mechanism for high-load systems on existing cloud capacities, eliminates the phenomena of resource overprovisioning and underprovisioning, and increases the profitability of IT-projects within the FinOps methodology.

ЗМІСТ

Вступ	9
1. Аналіз предметної області та існуючих підходів до оптимізації витрат на хмарний хостинг	11
1.1 Аналіз специфіки ІТ-проектів з розробки інтернет-магазинів комп'ютерних ігор та робіт, що здійснюються під час їх реалізації	11
1.2 Аналіз процесу управління вартістю та класифікація моделей ціноутворення у безсерверних архітектурах	12
1.3 Аналіз підходів, які можуть бути застосовані для оцінювання та управління бюджетом Serverless-проектів	17
1.3.1 Прогнозування та оптимізація витрат на базі машинного навчання	17
1.3.2 Миттєве оцінювання вартості за допомогою користувацького метричного моніторингу	18
1.3.3 Забезпечення прозорості витрат на рівні програмного коду (Cost-Transparent Cloud Programming)	19
1.3.4 Комплексне застосування підходів для розробки інтернет-магазинів комп'ютерних ігор	20
1.4 Особливості та проблеми існуючих методів управління вартістю, виходячи зі специфіки безсерверних технологій	21
1.5 Постановка задачі дослідження	25
2. Дослідження теоретично-методологічних основ управління вартістю serverless інфраструктури	27
2.1 Підхід оцінювання та оптимізації витрат на хмарний хостинг	27
2.1.1 Профілювання навантаження та аналіз трафіку	29
2.1.2 Проектування та розгортання безсерверної архітектури	29
2.1.3 Навантажувальне тестування та моніторинг метрик	30

2.1.4 Математичне моделювання та пошук оптимальних налаштувань	30
2.1.5 Формування управлінських рішень та налаштування Azure Cost Management	31
2.2 Модель визначення оптимального співвідношення вартості та продуктивності Serverless-мікросервісів	31
2.3 Вплив проблеми «холодних стартів» на економічну ефективність Serverless-додатків	35
2.4 Комплексна (гібридна) SLA-орієнтована модель управління вартістю з урахуванням бізнес-ризиків	36
3. Розробка та впровадження процедури управління вартістю безсерверного інтернет-магазину комп'ютерних ігор	39
3.1 Архітектура та послідовність управління вартістю Serverless-ресурсів	39
3.2 Проектування архітектури хмарної інфраструктури та процесу агрегації телеметричних даних	42
3.3 Порядок впровадження розробленої процедури в управлінські процеси ІТ-проекту	44
4. Практична реалізація моделі управління вартістю	46
4.1. Мета та умови проведення експерименту	46
4.2 Агрегація метрик та визначення 95-го перцентиля	48
4.3 Технічна фільтрація конфігурацій за критеріями SLA	51
4.4 Знаходження глобального економічного оптимуму та аналіз сукупної вартості володіння	52
Висновки	55
Перелік джерел посилання	57
Додаток А Графічний матеріал кваліфікаційної роботи	61

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

Agile – гнучка методологія розробки програмного забезпечення

ARIMA – (Autoregressive Integrated Moving Average) авторегресійна інтегрована ковзна середня

DDoS (Distributed Denial of Service) – розподілена атака типу «відмова в обслуговуванні»

DevOps (Development and Operations) – набір практик, що поєднує розробку програмного забезпечення та ІТ-експлуатацію

FaaS (Function-as-a-Service) – модель хмарних обчислень «Функція як послуга».

FinOps (Cloud Financial Operations) – методологія управління фінансами та оптимізації витрат у хмарних середовищах

GiB (Gibibyte) – гібібайт

IaaS/PaaS (Infrastructure-as-a-Service / Platform-as-a-Service) – інфраструктура як послуга / платформа як послуга

LSTM (Long Short-Term Memory) – архітектура рекурентної нейронної мережі довгої короткострокової пам'яті

Pay-as-you-go – модель тарифікації з оплатою виключно за фактично спожиті хмарні ресурси

Provisioned Concurrency – зарезервована потужність

RAM (Random Access Memory) – оперативний запам'ятовуючий пристрій

Scrum – фреймворк для управління процесом розробки програмного забезпечення

Serverless – безсерверні обчислення

SLA (Service Level Agreement) – угода про рівень надання послуг

vCPU (Virtual Central Processing Unit) – віртуальний центральний процесор

ВСТУП

Стрімка цифровізація світової економіки та перехід до хмарних обчислень призвели до широкого впровадження безсерверних архітектур. Парадигма Serverless надає розробникам можливість зосередитися на бізнес-логіці, делегуючи управління інфраструктурою провайдеру. Проте, не зважаючи на очевидні переваги моделі виставлення рахунків «Pay-as-you-go», управління витратами стає критичним викликом для менеджменту. Динамічне масштабування та мілісекундна тарифікація часто призводять до непередбачуваного зростання бюджетів.

Для інформаційних систем електронної комерції, зокрема інтернет-магазинів комп'ютерних ігор, проблема оптимізації тісно пов'язана з продуктивністю. Кожна затримка в обробці запиту має пряму залежність з відтоком користувачів та втратою прибутку. Традиційні підходи до управління витратами часто ігнорують вплив якості обслуговування на загальну рентабельність проєкту. Відсутність моделей, які поєднували б принципи FinOps із технічними вимогами до швидкодії, робить розроблення та впровадження спеціалізованих процедур управління вартістю актуальним завданням для сучасного управління ІТ-проєктами.

Метою роботи є підвищення економічної ефективності функціонування хмарної інфраструктури інтернет-магазину шляхом розроблення та впровадження гібридної моделі управління вартості, що орієнтована на SLA. Для досягнення поставленої мети необхідно вирішити завдання щодо аналізу особливостей тарифікації та масштабування в сучасних Serverless-платформах, дослідження впливу технічних конфігурацій ресурсів на час виконання транзакції та загальну вартість володіння. Передбачено обґрунтування та формулювання математичної моделі оптимізації витрат, що інтегрує фінансові ризики від порушення SLA, а також проведення експериментальної перевірки

працездатності розроблених процедур за допомогою навантажувального тестування та аналізу економічної ефективності.

Об'єктом дослідження виступають процеси управління вартістю та динамічного масштабування обчислювальних ресурсів у безсерверних архітектурах для систем електронної комерції. Предметом дослідження є моделі, методи та інформаційні технології оптимізації інфраструктурних витрат ІТ-проектів з урахуванням показників швидкодії та бізнес-ризиків.

Для розв'язання поставлених задач використано методи навантажувального тестування за допомогою інструменту Grafana k6 для збору емпіричних даних, математичне моделювання сукупної вартості володіння, статистичний аналіз телеметрії, а також концепції FinOps для оцінки результатів.

Практичне значення роботи полягає у створенні моделі, яка дозволяє ІТ-командам завбачливо управляти хмарними бюджетами, визначати оптимальні конфігурації ресурсів для усунення явищ Overprovisioning, Underprovisioning та мінімізувати бізнес-збитки від затримок обробки запитів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ ПІДХОДІВ ДО ОПТИМІЗАЦІЇ ВИТРАТ НА ХМАРНИЙ ХОСТИНГ

1.1 Аналіз специфіки ІТ-проектів з розробки інтернет-магазинів комп'ютерних ігор та робіт, що здійснюються під час їх реалізації

Електронна комерція, завдяки стрімкому розвитку комп'ютерних та інтернет-технологій, зазнала значного зростання в усіх сферах та суттєво змінила правила ведення бізнесу [1]. Розробка інтернет-магазинів комп'ютерних ігор є специфічним і дуже вимогливим напрямком електронної комерції. Головна особливість таких ІТ-проектів полягає в необхідності створення систем, які здатні витримувати раптові та колосальні піки навантаження, що виникають під час релізів нових популярних ігор або сезонних розпродажів.

Для сучасних вебзастосунків критично важливо гарантувати безперервність надання послуг навіть за умов дуже високих обчислювальних навантажень [2]. Саме тому під час реалізації подібних проектів хмарні обчислення та веб-інтеграція відіграють ключову роль, забезпечуючи необхідну масштабованість та зниження операційних витрат [3].

Аналіз робіт, що здійснюються під час розробки сучасних платформ для продажу ігор, дозволяє виділити такі ключові архітектурні та управлінські етапи:

- перехід до мікросервісної архітектури: Оскільки монолітні застосунки важко масштабувати, розробники все частіше обирають мікросервіси. Мікросервісна архітектура забезпечує значну горизонтальну масштабованість системи під високим навантаженням [4]. Крім того, розділення системи на невеликі незалежні функціональні одиниці (сервіси) дозволяє командам розробників набагато легше створювати, тестувати та розгортати код [4];

- налаштування хмарної ІТ-інфраструктури: Проектні роботи включають аналіз та впровадження передових технологій для побудови ІТ-інфраструктури,

яка ефективно надає сервіси користувачам [2]. Використання хмарних технологій та засобів автоматизації трансформує електронну комерцію, дозволяючи системі динамічно виділяти ресурси лише тоді, коли це потрібно [3];

– реалізація інтелектуальних та безпечних транзакцій: Платформи електронної комерції наступного покоління вимагають впровадження персоналізованих порталів для користувачів. Це дозволяє здійснювати інтелектуальний підбір (matching) пропозицій, уникати ізольованих інформаційних островів завдяки взаємопов'язаним системам та підтримувати паралельне оброблення даних. Такий підхід робить процес покупки ігор безпечнішим і прозорішим [1].

Отже, специфіка ІТ-проектів зі створення ігрових інтернет-магазинів полягає у фокусі на відмовостійкості. Основні роботи розробників та DevOps-інженерів спрямовані на міграцію логіки в хмару [3], оптимізацію розгортання мікросервісів [4] та автоматизацію інфраструктури [2], що є ідеальним підґрунтям для застосування Serverless-технологій.

1.2 Аналіз процесу управління вартістю та класифікація моделей ціноутворення у безсерверних архітектурах

Перехід від традиційних локальних серверів до хмарних інфраструктур докорінно змінив парадигму фінансового планування ІТ-проектів. Процес управління вартістю у хмарному середовищі є складним завданням через еластичність ресурсів та комплексність моделей тарифікації (білінгу), які пропонують хмарні провайдери [5]. Відсутність ефективного контролю за споживанням ресурсів часто призводить до непередбачуваних фінансових витрат. Тому для сучасних проектів критично важливо впроваджувати практики

безперервного моніторингу, прогнозування (зокрема, з використанням методів машинного навчання) та оптимізації витрат, що об'єднуються концепцією FinOps [5].

Основою для управління бюджетом є розуміння моделей ціноутворення. Аналіз існуючих підходів до тарифікації хмарних обчислень дозволяє класифікувати їх на кілька основних категорій: моделі з оплатою за фактом споживання (Pay-as-you-go), моделі передплати (Subscription) та гібридні схеми [6]. Розуміння того, як саме утилізуються ресурси в рамках кожної моделі, є ключовим фактором для вибору найбільш економічно вигідної стратегії [6].

Однак Serverless архітектури запроваджують принципово нову, більш гранулярну модель ціноутворення, яка суттєво відрізняється від традиційних IaaS або PaaS. Класифікація моделей ціноутворення для Serverless-рішень базується на таких принципах:

- модель мікробілінгу за виконання: У традиційних хмарних моделях віртуальні машини тарифікуються за час їх роботи (наприклад, погодинно), незалежно від того, чи виконують вони корисну роботу. Натомість Serverless-платформи не стягують плату за час простою [7]. Рахунок формується виключно на основі кількості викликів функцій та тривалості їх виконання, яка вимірюється з точністю до мілісекунд (наприклад, порціями по 100 мс) у поєднанні з обсягом виділеної оперативної пам'яті [7]. Це робить цю модель ідеальною для вебпроектів із нерівномірним трафіком;

- модель зарезервованої потужності (Provisioned Concurrency): Оскільки мікробілінг має недолік у вигляді затримок під час запуску нових екземплярів функцій («холодний старт»), провайдери пропонують гібридну модель. Вона передбачає фіксовану оплату за підтримку певної кількості функцій у стані постійної готовності, що дещо повертає процес управління вартістю до класичного планування ресурсів, але гарантує високу продуктивність.

Економічний та архітектурний вплив переходу на Serverless є значним. Як показують дослідження, для певних типів навантажень (наприклад, обробка

нерегулярних запитів користувачів) перехід на мікробілінг дозволяє скоротити витрати на хостинг від двох до десяти разів порівняно зі звичайними хмарними серверами [7]. Разом з тим, управління вартістю в таких умовах вимагає від менеджерів проєктів зміни мислення: бюджет тепер напряду залежить від якості та ефективності написаного програмістами коду, оскільки кожна зайва мілісекунда виконання алгоритму безпосередньо збільшує фінансові витрати [7].

Для наочного розуміння різниці між існуючими підходами до ціноутворення, було сформовано зведену порівняльну характеристику. У таблиці 1.1 наведено порівняльний аналіз фінансових та управлінських аспектів традиційних хмарних інфраструктур (IaaS/PaaS) та безсерверних архітектур .

Таблиця 1.1 – Порівняльний аналіз управлінських та фінансових моделей традиційних і безсерверних хмарних інфраструктур

Критерій порівняння	Традиційна хмарна модель (IaaS/PaaS)	Безсерверні контейнери (Azure Container Apps)	FaaS (Azure Functions / AWS Lambda)
Одиниця білінгу (тарифікації)	Години або хвилини роботи виділеної віртуальної машини.	Кількість запитів, активні vCPU-секунди та RAM-секунди.	Кількість викликів, мілісекунди часу виконання та обсяг оперативної пам'яті.

Продовження таблиці 1.1

Критерій порівняння	Традиційна хмарна модель (IaaS/PaaS)	Безсерверна модель (Serverless)	FaaS (Azure Functions / AWS Lambda)
Оплата за час простою	Оплачується повною мірою.	Дорівнює нулю при масштабуванні до 0 реплік.	Дорівнює нулю.
Прозорість інфраструктури	Висока	Середня (керування образом контейнера та ресурсами)	Низька (інфраструктура повністю прихована).
Вектор оптимізації витрат	Налаштування конфігурації сервера силами DevOps-інженерів.	Оптимізація коду та підбір профілю vCPU/RAM..	Оптимізація алгоритмів у коді розробниками.
Масштабування (Scaling)	Вимагає налаштування правил авто-масштабування; має затримки.	Швидко, автоматичне, на основі HTTP-запитів або подій.	Миттєве, автоматичне, на рівні окремих паралельних запитів.

Продовження таблиці 1.1

Критерій порівняння	Традиційна хмарна модель (IaaS/PaaS)	Безсерверна модель (Serverless)	FaaS (Azure Functions / AWS Lambda)
Головний фінансовий ризик	Перевитрати на утримання надлишкових потужностей.	Поєднання ризику Overprovisioning та «bill shock» від навантаження.	Екстремальні перевитрати через архітектурні помилки, зациклення або DDoS-атаки.

Результати порівняльного аналізу свідчать, що перехід до безсерверних архітектур, зокрема у формі контейнеризованих рішень типу Azure Container Apps, докорінно змінює парадигму фінансового менеджменту ІТ-проектів. Відповідальність за економічну ефективність переміщується від адміністраторів інфраструктури до розробників та ІТ-менеджерів, оскільки бюджет тепер напряму корелює з обраною конфігурацією обчислювальних ресурсів та якістю програмного коду. Це актуалізує потребу в застосуванні аналітичних методик та практик FinOps для забезпечення прозорості та проактивного контролю витрат в умовах динамічного масштабування.

1.3 Аналіз підходів, які можуть бути застосовані для оцінювання та управління бюджетом Serverless-проектів

Стрімкий розвиток хмарних обчислень загалом та безсерверних архітектур зокрема докорінно змінив парадигму розгортання інформаційних систем. Завдяки еластичності хмарних платформ розробники отримали можливість динамічно масштабувати ресурси відповідно до поточних потреб користувачів. Однак ця еластичність та складність сучасних моделей білінгу створюють значні виклики для ефективного управління бюджетом проекту [8]. У традиційних архітектурах витрати на інфраструктуру є здебільшого фіксованими (капітальні витрати), тоді як у Serverless-середовищі вони є повністю змінними (операційні витрати) і залежать від кількості викликів функцій, часу їх виконання та обсягу спожитої оперативної пам'яті.

Аналіз сучасної науково-практичної літератури дозволяє виділити три фундаментальні підходи до оцінювання та управління бюджетом хмарних та безсерверних проектів, які можуть бути імплементовані під час розробки високонавантажених інтернет-магазинів: підхід на базі машинного навчання (макрорівень), підхід миттєвого метричного моніторингу (мезорівень) та підхід прозорості витрат на рівні коду (мікрорівень).

1.3.1 Прогнозування та оптимізація витрат на базі машинного навчання

Перший підхід базується на інтеграції фінансових операцій у процеси хмарної розробки (концепція FinOps) з використанням передових аналітичних інструментів. Оскільки хмарні моделі тарифікації є надзвичайно складними, традиційні методи бюджетування часто призводять або до перевитрат, або до

недостатнього виділення ресурсів (що критично для інтернет-магазинів під час розпродажів) [8].

Для вирішення цієї проблеми застосовуються гібридні моделі прогнозування, що поєднують статистичні методи та машинне навчання. Наприклад, дослідження показують високу ефективність використання авторегресійних інтегрованих моделей ковзного середнього у поєднанні з нейронними мережами довгої короткострокової пам'яті для аналізу історичних даних про витрати [8]. ARIMA забезпечує точне моделювання лінійних тенденцій та сезонності (наприклад, щомісячних піків активності покупців), тоді як алгоритми LSTM ефективно виявляють нелінійні, складні патерни споживання ресурсів, які важко передбачити традиційними методами [8].

Більше того, управління бюджетом може бути автоматизоване за допомогою методів навчання з підкріпленням (Reinforcement Learning), зокрема алгоритму DDPG (Deep Deterministic Policy Gradient). Цей метод дозволяє системі безперервно навчатися на основі взаємодії з хмарним середовищем та приймати оптимальні рішення щодо виділення ресурсів, мінімізуючи витрати при збереженні заданого рівня продуктивності [8]. Використання таких комплексних систем дозволяє покращити точність прогнозування витрат на 15-25% порівняно з базовими методами [8].

1.3.2 Миттєве оцінювання вартості за допомогою користувацького метричного моніторингу

Другий підхід вирішує проблему затримки білінгової інформації від хмарних провайдерів. Зазвичай клієнти хмарних сервісів бажають миттєво коригувати політику використання ресурсів на основі їх вартості. Однак миттєва реакція часто є неможливою, оскільки хмарним провайдерам

(наприклад, AWS, Azure) потрібні години або навіть кілька днів для агрегації та оновлення білінгової інформації на панелі управління клієнта [9]. У контексті Serverless-проектів, де тисячі функцій можуть бути викликані одночасно (наприклад, під час релізу нової гри), така затримка може призвести до колосальних фінансових втрат до того моменту, як менеджер проекту помітить аномалію.

Для розв'язання цієї проблеми застосовується модель миттєвого оцінювання вартості. Суть підходу полягає у створенні спеціалізованих демонів моніторингу (metering daemons) на боці користувача, які в реальному часі відстежують використання ресурсів (процесорного часу, оперативної пам'яті, мережевого трафіку) від імені віртуальних екземплярів або безсерверних функцій [9].

Цей підхід дозволяє створювати високоточні локальні калькулятори вартості, алгоритми яких максимально наближені до тих, що використовують самі хмарні провайдери. Завдяки миттєвому збору метрик на боці клієнта, менеджери проектів отримують можливість встановлювати автоматичні ліміти (alerts and triggers) на виконання певних функцій ще до того, як офіційний рахунок від провайдера буде сформовано [9]. Це забезпечує жорсткий контроль над бюджетом на стадії розгортання та експлуатації системи.

1.3.3 Забезпечення прозорості витрат на рівні програмного коду (Cost-Transparent Cloud Programming)

Третій підхід є найбільш інноваційним і фокусується безпосередньо на процесі розробки програмного забезпечення. У традиційних моделях розробники часто не усвідомлюють фінансових наслідків написаного ними коду. У Serverless-архітектурі кожна мілісекунда виконання функції та кожен

виділений мегабайт пам'яті мають свою ціну. Отже, архітектурні або алгоритмічні неточності безпосередньо призводять до збільшення фінансових витрат.

Для мінімізації таких ризиків застосовується підхід прозорості витрат (Cost-Transparent Cloud Programming), що передбачає інтеграцію механізмів оцінювання вартості безпосередньо в Інтегровані середовища розробки [10]. Цей метод базується на концепції «живого програмування» де код безперервно аналізується та виконується у фоновому режимі під час його написання розробником.

Відповідно до цієї парадигми (відомої як «A Penny a Function»), середовище розробки не лише перевіряє код на наявність синтаксичних помилок, але й надає миттєвий зворотний зв'язок щодо орієнтовної вартості виконання кожної окремої хмарної функції [10]. Розробники отримують можливість порівнювати різні варіанти імплементації алгоритму не лише за критеріями швидкості та споживання пам'яті, але й за економічним критерієм. Візуалізація вартості на рівні вихідного коду докорінно змінює процес прийняття рішень: оптимізація бюджету починається не на етапі адміністрування хмари (DevOps), а безпосередньо під час написання бізнес-логіки розробниками [10].

1.3.4 Комплексне застосування підходів для розробки інтернет-магазинів комп'ютерних ігор

Застосування вищезазначених підходів є критично важливим для проєктів з розробки інтернет-магазинів комп'ютерних ігор. Такі платформи характеризуються нерівномірним розподілом навантаження. Інтеграція моделей машинного навчання (ARIMA/LSTM) дозволить глобально прогнозувати

бюджет на основі маркетингового календаря (релізи, акції) [8]. Використання миттєвого метричного моніторингу захистить проєкт від непередбачуваних фінансових втрат у разі раптових сплесків трафіку або DDoS-атак [8]. Водночас забезпечення прозорості витрат на рівні коду дозволить розробникам оптимізувати найчастіше викликані мікросервіси (наприклад, сервіс кошика або каталог товарів), гарантуючи, що їх щоденне виконання залишатиметься економічно рентабельним [10].

Таким чином, управління бюджетом Serverless-проєктів вимагає переходу від класичних бухгалтерських методів до інженерно-аналітичних інструментів, що охоплюють увесь життєвий цикл розробки – від написання коду до предиктивного масштабування інфраструктури.

1.4 Особливості та проблеми існуючих методів управління вартістю, виходячи зі специфіки безсерверних технологій

Управління вартістю IT-проєктів загалом є надзвичайно складним завданням, оскільки ця сфера об'єктивно характеризується високим ступенем невизначеності та схильністю до фінансових ризиків. Емпіричні дослідження великих масивів даних доводять, що перевитрати бюджету в IT-галузі не підпорядковуються нормальному (гауссовому) розподілу, де більшість проєктів мають лише незначні відхилення від плану. Натомість такі відхилення демонструють розподіл за степеневим законом (Power-Law Distribution), що формує так звані «товсті хвости» ймовірностей [11]. Це означає, що існує статистично висока ймовірність виникнення екстремальних перевитрат (явища, які часто називають «чорними лебедями» або «королями драконів»), коли фінальний бюджет проєкту може перевищити початкові оцінки в кілька разів [11]. Традиційні інструменти управління проєктами та методи оцінки вартості

системно ігнорують цю емпіричну реальність, помилково моделюючи ризики як помірні величини, що зрештою призводить до недооцінки ймовірності катастрофічних фінансових наслідків під час розробки високонавантажених систем.

Перехід підприємств до хмарних обчислень та модернізація існуючих додатків мали б стати рішенням проблеми оптимізації витрат. Проте на практиці багато спроб міграції у хмару завершуються невдачею або суттєвими затримками саме через перевитрати та недостатнє початкове планування [12]. Для прийняття стратегічних рішень щодо міграції архітектури необхідно мати попередню оцінку орієнтовної вартості експлуатації у хмарі (Rough Order of Magnitude – ROM), яка слугує базовим вхідним параметром для ІТ-управління [12]. Однак існуючі підходи до оцінки ROM, які були розроблені для традиційних інфраструктурних моделей (IaaS або PaaS), виявляються абсолютно нерелевантними під час впровадження моделі Function-as-a-Service (FaaS) або Serverless. Відсутність адаптованих інструментів для профілювання та автоматичного виявлення залежностей у безсерверному середовищі унеможливорює точне прогнозування бюджету на початкових етапах життєвого циклу проєкту [12].

Головна специфіка безсерверних технологій полягає у тому, що розробники повністю абстрагуються від управління серверами, делегуючи масштабування інфраструктури хмарному провайдеру. З одного боку, це забезпечує вбудовану еластичність і високу доступність, але з іншого – породжує критичну проблему: фундаментальну відсутність прозорості інфраструктури [13]. Оскільки команди розробників не мають доступу до базових метрик фізичних або віртуальних серверів, існуючим методам управління бракує формальних інструментів для детального моделювання продуктивності та вартості [13]. Менеджерам проєктів вкрай важко знайти оптимальний компроміс між швидкістю відгуку системи та витратами на її утримання. Як результат, точне передбачення метрик, таких як час виконання,

вартість, статус завершення та їхній розподіл для складних робочих процесів додатку, залишається нерозв'язаною проблемою в рамках стандартних управлінських фреймворків [13].

Ще однією вкрай специфічною проблемою існуючих підходів є ігнорування нелінійної залежності між виділеними обчислювальними ресурсами та загальною вартістю виконання коду. У таких FaaS-середовищах, як AWS Lambda, білінг формується на основі кількості викликів функції та тривалості її виконання, яка напряду залежить від виділеного обсягу оперативної пам'яті [14]. Технологічна особливість полягає в тому, що обчислювальна потужність процесора виділяється хмарним провайдером пропорційно до обсягу пам'яті. Відповідно, збільшення обсягу пам'яті робить кожну мілісекунду виконання функції дорожчою, але одночасно це може настільки пришвидшити обробку запиту, що загальний фінансовий рахунок за виклик парадоксально зменшиться [14]. Більшість розробників залишають конфігурації пам'яті за замовчуванням, що призводить до величезних прихованих перевитрат. Експериментальні дослідження доводять, що формалізований вибір «найбільш сприятливого розміру ресурсу» здатен оптимізувати вартість експлуатації для 80% безсерверних мікросервісів [14]. Класичні ж методи калькуляції бюджету просто не мають математичного апарату для такого динамічного налаштування конфігурацій на рівні окремих мікросервісів інтернет-магазину.

Окрім суто технічних і математичних аспектів, архітектура Serverless актуалізує низку поведінкових та соціальних проблем у процесі управління командами розробників. В умовах еластичного масштабування обчислювальні потужності можуть хибно сприйматися як невичерпний ресурс загального користування [15]. Якщо традиційні віртуальні машини розглядаються менеджерами як фіксований і безпечний актив із чіткою межею споживання, то безсерверні функції через свою «спільну» природу є вкрай вразливими до надмірної експлуатації – явища, відомого в економіці як «Трагедія спільного»

(Tragedy of the Commons) [15]. Коли розробники мають технічну можливість нескінченно генерувати виклики функцій (наприклад, через неоптимальний цикл у кодї або погано спроектовану архітектуру бази даних), відсутність жорстких інфраструктурних бар'єрів миттєво конвертується у неконтрольоване зростання операційних витрат підприємства.

Щоб подолати цю проблему, новітні дослідження пропонують застосовувати моделі, засновані на Теорії перспектив, яка враховує психологічні та поведінкові характеристики суб'єктів прийняття рішень – їхню схильність шукати вигоду або намагатися уникнути втрат (loss-averse behavior) [15].

Традиційні статичні підходи до ризик-менеджменту та ціноутворення не здатні адекватно працювати в таких розподілених хмарних екосистемах, оскільки вони не враховують соціальну взаємодію між учасниками проєкту (розробниками, DevOps-інженерами, фінансовим відділом) та їхню реакцію на невизначеність навантаження [15]. Управління бюджетом вимагає впровадження спеціалізованих функцій корисності, які б математично формалізували сприйняття ризику під час гібридного розподілу завдань між фіксованими серверами та безсерверними потужностями.

Підсумовуючи вищезазначене, можна зробити висновок, що пряме перенесення існуючих методів оцінки вартості ІТ-проєктів на Serverless-технології є концептуально хибним та економічно небезпечним кроком. По-перше, класичні моделі ігнорують наочно доведений степеневий розподіл ризиків (Power-Law), що робить бюджети вразливими до екстремальних перевитрат [15]. По-друге, нездатність існуючих методик формувати точні попередні оцінки (ROM) в умовах обмеженої прозорості інфраструктури гальмує ефективну модернізацію корпоративних застосунків [13], [14]. По-третє, ігнорування нелінійної проблеми конфігурування співвідношення пам'яті до швидкодії [14] у поєднанні з відсутністю контролю за поведінковими ризиками «Трагедії спільного» [15] позбавляє проєктного

менеджера можливості ефективно оптимізувати операційний бюджет на стадії впровадження вебпроєкту.

1.5 Постановка задачі дослідження

Аналіз предметної області показав, що існуючі методи оцінки вартості ІТ-проєктів не підходять для сучасних безсерверних технологій. У системах електронної комерції, які постійно стикаються з раптовими сплесками активності, критично важливо знайти баланс між швидкістю роботи та витратами на хостинг. Тому головною задачею дослідження є вибір та обґрунтування моделі оптимізації витрат на хмарну інфраструктуру, яка б забезпечила цей баланс на етапі впровадження.

Складність задачі полягає в тому, що у безсерверних системах фінансові витрати формуються динамічно. Рахунок залежить від багатьох змінних: кількості запитів, часу обробки та обсягу виділеної пам'яті. Без контролю цих процесів проєкт ризикує отримати непередбачувані перевитрати. Отже, стандартних калькуляторів провайдерів недостатньо – потрібен глибокий аналітичний підхід.

Для досягнення поставленої мети необхідно вирішити такі конкретні задачі:

- розробити модель, яка імітуватиме реальну поведінку користувачів;
- розгорнути базову безсерверну архітектуру для типового компонента інтернет-магазину на основі сучасної хмарної платформи;
- здійснити навантажувальне тестування з різними конфігураціями виділеної пам'яті. Це дозволить зібрати дані про зміну часу відгуку та вартість обробки кожної транзакції;

- на основі отриманих результатів обчислити найбільш вигідний обсяг ресурсів та визначити точку беззбитковості (межу, після якої прискорювати роботу економічно недоцільно);

- сформулювати алгоритм, який допоможе ІТ-менеджерам приймати обґрунтовані рішення при виборі тарифних планів, враховуючи наявний бюджет і вимоги до якості.

Таким чином, виконання цих завдань зробить процес управління витратами більш передбачуваним і дозволить оптимізувати фінанси ще до повноцінної експлуатації системи.

2 ДОСЛІДЖЕННЯ ТЕОРЕТИЧНО-МЕТОДОЛОГІЧНИХ ОСНОВ УПРАВЛІННЯ ВАРТІСТЮ SERVERLESS ІНФРАСТРУКТУРИ

2.1 Підхід оцінювання та оптимізації витрат на хмарний хостинг

Проекти з Serverless архітектурами характеризуються динамічною, нелінійною природою ціноутворення, тому ефективне управління є актуальною проблемою. Оптимізація витрат у такому випадку не обмежується лише вибором тарифного плану або базовим налаштуванням хмарних ресурсів. Для гарантії стабільного функціонування високонавантаженої системи і уникнення надлишкових фінансових витрат, розробникам та менеджерам необхідно слідувати чітким системним підходам. З огляду на поставлені цілі дослідження, було розроблено п'яти кроковий алгоритм дій, що дозволить знайти рівновагу між операційною вартістю і продуктивністю хмарного хостингу. Загальна структурно-логічна схема запропонованого підходу наведена на рисунку 2.1.



Рисунок 2.1 – Схема підходу до оптимізації вартості безсерверної інфраструктури

2.1.1 Профілювання навантаження та аналіз трафіку

Перший етап передбачає проведення аналізу активності користувачів. Кількість запитів у системах з безсерверною архітектурою безпосередньо впливає на грошові витрати. Саме тому необхідно розробити профіль навантаження для системи. Він має враховувати як звичайні дні експлуатації, так і періоди піків трафіку. В результаті такого аналізу можливо підготувати реалістичні сценарії для подальшого тестування інфраструктури.

2.1.2 Проектування та розгортання безсерверної архітектури

Наступним кроком є технічна підготовка до тесту. На даному етапі відбувається розробка базового мікросервісу мовою C#, що виступить в ролі піддослідного, виконуючи типові для проекту інтернет-магазину завдання. Тестовий додаток розгортається у хмарі Microsoft у контейнері Azure Container Apps. Так як Azure – це найкраща хмара для розробників .NET, оскільки вона була створена з урахуванням потреб розробників .NET [16].

Тестувати потрібно реальний програмний код, адже швидкість його виконання напряму залежить від специфіки інтеграції .NET-додатків з інфраструктурою Azure.

2.1.3 Навантажувальне тестування та моніторинг метрик

Наступний, третій крок полягає у зборі практичних даних. Для цього до мікросервісу, що розгорнутий у хмарі, направляється штучний трафік за сценаріями, розробленими в першому кроці. В процесі тестування поступово змінюється конфігурація обчислювальних ресурсів (потужність процесора vCPU та обсяг оперативної пам'яті), виділених для контейнера. У цей час вбудована система телеметрії Azure Application Insights детально фіксує всі необхідні метрики: час відгуку, тривалість холодних стартів та споживання ресурсів [17]. Ці системні логи стануть надійним фундаментом для фінансового аналізу.

2.1.4 Математичне моделювання та пошук оптимальних налаштувань

Зібрані метрики проходять аналітичну обробку на четвертому етапі. Зв'язок між обраною конфігурацією ресурсів, ціною та часом роботи є нелінійним. З цієї причини просте порівняння базових тарифів не дасть бажаного точного результату. Буде застосовано математичну модель, яка шукає точку беззбитковості – межі, після якої виділення додаткових ресурсів для прискорення роботи коду призводить до невиправданих витрат і стає фінансово не вигідним рішенням.

2.1.5 Формування управлінських рішень та налаштування Azure Cost Management

Фінальний етап перетворює математичні розрахунки у площину управління проєктами. На основі знайденої оптимальної конфігурації ресурсів керівник проєкту ухвалює рішення щодо вибору конкретної моделі білінгу. Після цього в панелі Azure Cost Management встановлюються жорсткі фінансові ліміти та налаштовуються сповіщення про можливі перевитрати. Це допомагає проактивно контролювати бюджет під час реальної роботи вебпроєкту.

2.2 Модель визначення оптимального співвідношення вартості та продуктивності Serverless-мікросервісів

Розробка ефективного методу управління фінансами у Serverless-архітектурах вимагає детального розуміння базових принципів роботи хмарних платформ. Тому запропонована аналітична модель базується на послідовному дослідженні механізмів ціноутворення та формалізації критеріїв оптимізації.

2.2.1 Аналіз особливостей ціноутворення та масштабування ресурсів

У середовищі Azure Container Apps фінансова вартість виконання мікросервісу залежить від конфігурації ресурсів та часу їх споживання. Першим параметром є налаштування обчислювальних ресурсів, які дозволяють

розробнику явним чином налаштувати як потужність процесора так і обсяг оперативної пам'яті. Другий параметр – фактичний час виконання коду, який вимірюється в мілісекундах [18].

Для точного фінансового прогнозування прямі витрати на хмарну інфраструктуру C_{cloud} за розрахунковий період (наприклад, місяць) можна формалізувати наступним чином:

$$C_{cloud} = (vCPU \times P_{vCPU} + RAM \times P_{RAM}) \times T_{month}, \quad (2.1)$$

де $vCPU$ – кількість виділених ядер процесора;

P_{vCPU} – базова вартість однієї секунди роботи процесора;

RAM – обсяг виділеної оперативної пам'яті (ГБ);

P_{RAM} – базова вартість однієї секунди роботи пам'яті;

T_{month} – сумарний активний час роботи у секундах за розрахунковий місяць.

2.2.2 Дослідження нелінійної залежності між швидкодією і вартістю

Тариф за кожен мілісекунду роботи стає дорожчим зі збільшенням обсягу виділених потужностей $vCPU$ та оперативної пам'яті [7]. Оскільки в Azure Container Apps розробник має змогу власноруч виділяти більше процесорного часу та пам'яті для контейнера, це створює наступну залежність: інколи вибір потужнішої та дорожчої конфігурації ресурсів призводить до зменшення загальної вартості, так як мікросервіс відпрацьовує набагато швидше.

З іншого ж боку, існуюча межа перешкоджає прискоренню системи при подальшому додаванні ресурсів. Зокрема, якщо затримка виконання функції пов'язана із синхронним очікуванням відповіді від зовнішніх сервісів, таких як бази даних чи об'єктні сховища, додатково виділена потужність процесора не

здатна вплинути на цей процес [19]. В цьому випадку виникнуть фінансові перевитрати, через оплату дорогого тарифу без виграшу у швидкодії [19, 20].

2.2.3 Формалізація математичної моделі та критеріїв оптимізації

Модель пошуку точки беззбитковості (Break-even point) розробляється для уникнення перевитрат. Це конкретний обсяг пам'яті, при якому досягається найнижча загальна вартість виконання операції.

Цільову функцію для оптимізації можна представити у вигляді моделі мінімізації витрат. Загальна вартість розраховується як добуток ключових змінних: загальної кількості викликів, тривалості виконання функції, обсягу виділеної пам'яті та базової ціни хмарного провайдера за одиницю ресурсів (ГБ-секунду). Формалізовано це рівняння має такий вигляд:

$$C(M) = N \times T(M) \times M \times P_{res}, \quad (2.2)$$

де $C(M)$ – загальна вартість виконання функції;

N – загальна кількість викликів (транзакцій) за розрахунковий період;

$T(M)$ – тривалість виконання функції, що нелінійно залежить від обсягу пам'яті;

M – обсяг оперативної пам'яті для одного екземпляра функції;

P_{res} – базова ціна хмарного провайдера за одиницю ресурсів.

При цьому час виконання функції не повинен перевищувати максимально допустиму норму (SLA), яка є критичною для якості обслуговування користувачів [20]. У вигляді математичної нерівності це обмеження записується так:

$$T(M) \leq T_{SLA}, \quad (2.3)$$

де T_{SLA} – максимально допустимий час відгуку системи.

Отримуємо цільову функцію знаходження оптимального обсягу ресурсів для мінімізації витрат (2.3).

$$M_{opt} = \arg \min_M C(M), \quad (2.4)$$

де M_{opt} – економічно оптимальний обсяг ресурсів (точка беззбитковості), що забезпечує мінімум функції $C(M)$ за умови виконання системних обмежень.

2.2.4 Практичне застосування алгоритму пошуку оптимуму

Запропонована математична модель працює на основі експериментальних даних, зібраних під час навантажувального тестування мікросервісу. Алгоритм порівнює зміну часу виконання при кожному ітеративному збільшенні виділених ресурсів. Коли відсоток прискорення коду стає меншим за відсоток подорожчання базового тарифу, алгоритм фіксує проходження точки беззбитковості. Знайдені у такий спосіб налаштування ресурсів вважаються економічно оптимальними і затверджуються для використання у промисловому середовищі вебпроєкту.

2.3 Вплив проблеми «холодних стартів» на економічну ефективність Serverless-додатків

Модель оптимізації, розроблена в попередньому підрозділі прекрасно функціонує в умовах стабільного навантаження. Проте, використання безсерверних архітектур в реальних умовах супроводжується специфічними інфраструктурними явищами, які здатні значно викривляти результати розрахунків. Таким явищем є затримка ініціалізації, або так званий Cold Start. Основна перевага Serverless-архітектури, зокрема плану Azure Container Apps Consumption, полягає у масштабуванні до нуля. У випадку якщо до мікросервісу не надходить запитів, хмарний провайдер повністю звільняє обчислювальні ресурси і проєкт не платить за простій [21]. Однак, коли надходить новий запит після періоду простою, інфраструктура потребує додаткового часу на виділення нового контейнера, запуск середовища та завантаження коду в оперативну пам'ять [22]. Для компільованих мов програмування зі складним середовищем виконання, таких як C#, тривалість холодного старту традиційно є вищою в порівнянні зі скриптовими мовами і може навіть досягати кількох секунд.

Наявність холодних стартів створює значний ризик для оптимізаційної моделі (формула 2.1). Змінна $T(M)$, яка позначає тривалість виконання функції, перестає бути стабільною. Коли контейнер вже працює функція може виконуватися за мілісекунди, тоді як під час холодного старту цей час зростає в десятки разів. Така різниця у часі виконання призводить до двох негативних наслідків. По-перше це штучно збільшує загальну вартість експлуатації системи $C(M)$, оскільки провайдер тарифікує час холодної ініціалізації як звичайний час роботи обчислювальних ресурсів [21]. По-друге, виникає миттєве порушення обмеження (формула 2.2). Якщо максимальний час відповіді для вебсистеми сягає 2 секунди, то тривалий холодний старт призведе до втрати клієнта. З огляду на це, під час практичних розрахунків оптимального обсягу пам'яті

необхідно орієнтуватися не на середній час виконання, а на 95-й або 99-й перцентиль (P95/P99) зібраних під час тестування метрик [22].

Щоб досягти коректної роботи математичної моделі та дотримання умов SLA потрібно застосовувати архітектурні або програмні методи мінімізації холодних стартів. В екосистемі .NET для контейнеризованих одним із найефективніших програмних підходів є використання технології компіляції Ahead-of-Time (Native AOT). Вона дозволяє перетворити C# код на нативний машинний код, що повністю усуває необхідність запуску «важкого» середовища CLR під час старту контейнера і скорочує затримку у кілька разів [23]. Альтернативним інфраструктурним рішенням в Azure Container Apps є перехід на налаштування кількості реплік ($\text{minReplicas} > 0$), або використання профілю Dedicated. Це дозволяє тримати заздалегідь визначену кількість контейнерів завжди активними (Pre-warmed instances), що повністю виключає явище холодних стартів [24]. Варто зазначити, що такий перехід повністю змінює модель ціноутворення: замість оплати за час роботи проєкт змушений платити фіксовану базову ставку за зарезервовані ресурси. Це потребує окремого економічного обґрунтування, яке виходить за межі запропонованої базової моделі оптимізації мікробілінгу.

2.4 Комплексна (гібридна) SLA-орієнтована модель управління вартістю з урахуванням бізнес-ризиків

Базова математична модель оптимізації, що описана формулами (2.1)-(2.3), вирішує задачу мінімізації прямих інфраструктурних витрат. Проте цей підхід є недостатнім для систем електронної комерції, яким є інтернет-магазин комп'ютерних ігор.

У сфері електронної комерції кожна затримка при завантаженні або обробці сторінки призводить до відтоку клієнтів. Порушення технічного ліміту T_{SLA} (формула 2.2) або високі показники 95-го перцентиля через холодні старти, генерують приховані фінансові збитки, які затьмарюють економію на дешевих хмарних тарифах.

З огляду на вище згадану проблему було розроблено гібридну модель управління вартістю, що адаптує методи фінансових операцій до специфіки безсерверних систем шляхом монетизації швидкодії. Запропонована функція (формула 2.4) зміщує фокус з простого пошуку найдешевшого ресурсу на пошук мінімуму сукупних витрат для підприємства.

$$C_{\Sigma}(M) = C_{cloud}(M) + P_{loss}(T_{p95}(M)) \rightarrow \min, \quad (2.5)$$

де $C_{\Sigma}(M)$ – сукупна вартість володіння мікросервісом при виділеному обсязі пам'яті M ;

$C_{cloud}(M)$ – прямі витрати на хмарну інфраструктуру, які розраховуються за формулою (2.1)

$P_{loss}(T_{p95})$ – фінансовий еквівалент втраченої вигоди. Відображає прогнозовану втрату прибутку магазину через зниження конверсії, що напряду залежить від 95-го перцентиля часу виконання транзакції.

Для математичного розрахунку фінансового еквівалента втраченої вигоди P_{loss} використовується наступна формула (2.6):

$$P_{loss} = \max(0, T_{p95} - T_{SLA}) \times C_{delay} \times N_{peak}, \quad (2.6)$$

де T_{p95} – 95-й процентиль часу виконання транзакції;

T_{SLA} – максимально допустимий час відповіді, встановлений бізнес-вмогою;

C_{delay} – фінансовий штраф за кожен секунду затримки понад SLA;

N_{peak} – кількість пікових годин на місяць, під час яких застосовується цей штраф.

Застосування моделі, представленої на формулі (2.5), дозволить математично довести, що для обчислювально-навантажених процесів штучне збільшення інфраструктурних витрат C_{cloud} шляхом переходу на дорожчий тариф є економічно вигідним рішенням, якщо це призведе до падіння показнику штрафу P_{loss} і зменшення загальної суми C_{Σ} . Для I/O-залежних процесів, де обсяг пам'яті не впливає на T_{p95} модель блокує виділення надлишкових ресурсів.

3 РОЗРОБКА ТА ВПРОВАДЖЕННЯ ПРОЦЕДУРИ УПРАВЛІННЯ ВАРТІСТЮ БЕЗСЕРВЕРНОГО ІНТЕРНЕТ- МАГАЗИНУ КОМП'ЮТЕРНИХ ІГОР

3.1 Архітектура та послідовність управління вартістю Serverless-ресурсів

Запропонована процедура призначена для автоматизації управлінського процесу прийняття рішень щодо масштабування ресурсів безсерверної інфраструктури інтернет-магазину комп'ютерних ігор. Головна мета якої – знаходження оптимального обсягу виділеної оперативної пам'яті M_{opt} . Цей параметр має зменшити фінансові витрати на хмарний хостинг за умови дотримання SLA під час пікових навантажень – наприклад, під час релізів довгоочікуваних проєктів чи великих сезонних розпродажів.

На початковому етапі отримуємо набір телеметричних даних з моніторингової платформи Azure Application Insights. Оскільки специфіка безсерверних систем передбачає наявність холодних стартів, алгоритм має виконувати статистичну очистку вибірки від цих інфраструктурних явищ. Для гарантії високої якості обслуговування користувачів, базовою метрикою продуктивності має виступати не середній час, а 95-перцентиль часу виконання запиту. Математично це представлено у формулі 3.1.

$$T_{p95}(M) = percentile(\{t_1, t_2, t_3, \dots, t_n\}, 95), \quad (3.1)$$

де t_i – тривалість обробки окремої транзакції при заданому обсязі пам'яті M .

Наступним кроком алгоритм виконує ітеративний розрахунок прогнозованої вартості інфраструктури для всіх налаштувань пам'яті M , що підтримуються провайдером. Результатом цього розрахунку є формування матриці альтернативних рішень, в якій містяться зведені дані щодо продуктивності та вартості для кожної опції масштабування.

Фінальна стадія зводиться до багатокритеріального відбору оптимальних налаштувань з отриманої матриці. Застосовується технічний фільтр: відсіюються всі конфігурації, що не забезпечують необхідної швидкості роботи. Так формується підмножина допустимих значень Ω_{valid} :

$$\Omega_{valid} = \{M \in \Omega \mid T_{p95}(M) \leq T_{SLA}\}, \quad (3.2)$$

де Ω – загальна початкова множина доступних рівнів оперативної пам'яті;

T_{SLA} – максимально допустимий час обробки запиту.

Серед конфігурацій, що увійдуть до Ω_{valid} , алгоритм шукатиме глобальний економічний оптимум. Кінцеве рішення M_{opt} формула (3.3) відповідатиме конфігурації, яка забезпечує найнижчу сукупну вартість володіння з урахуванням бізнес-ризиків $C_{\Sigma}(M)$ згідно з моделлю (2.4).

$$M_{opt} = \arg \min_{M \in \Omega_{valid}} C_{\Sigma}(M) \quad (3.3)$$

Всю послідовність, починаючи від збору метрик і закінчуючи затвердженням оптимальної конфігурації, зображено на блок-схемі (рисунок 3.1)

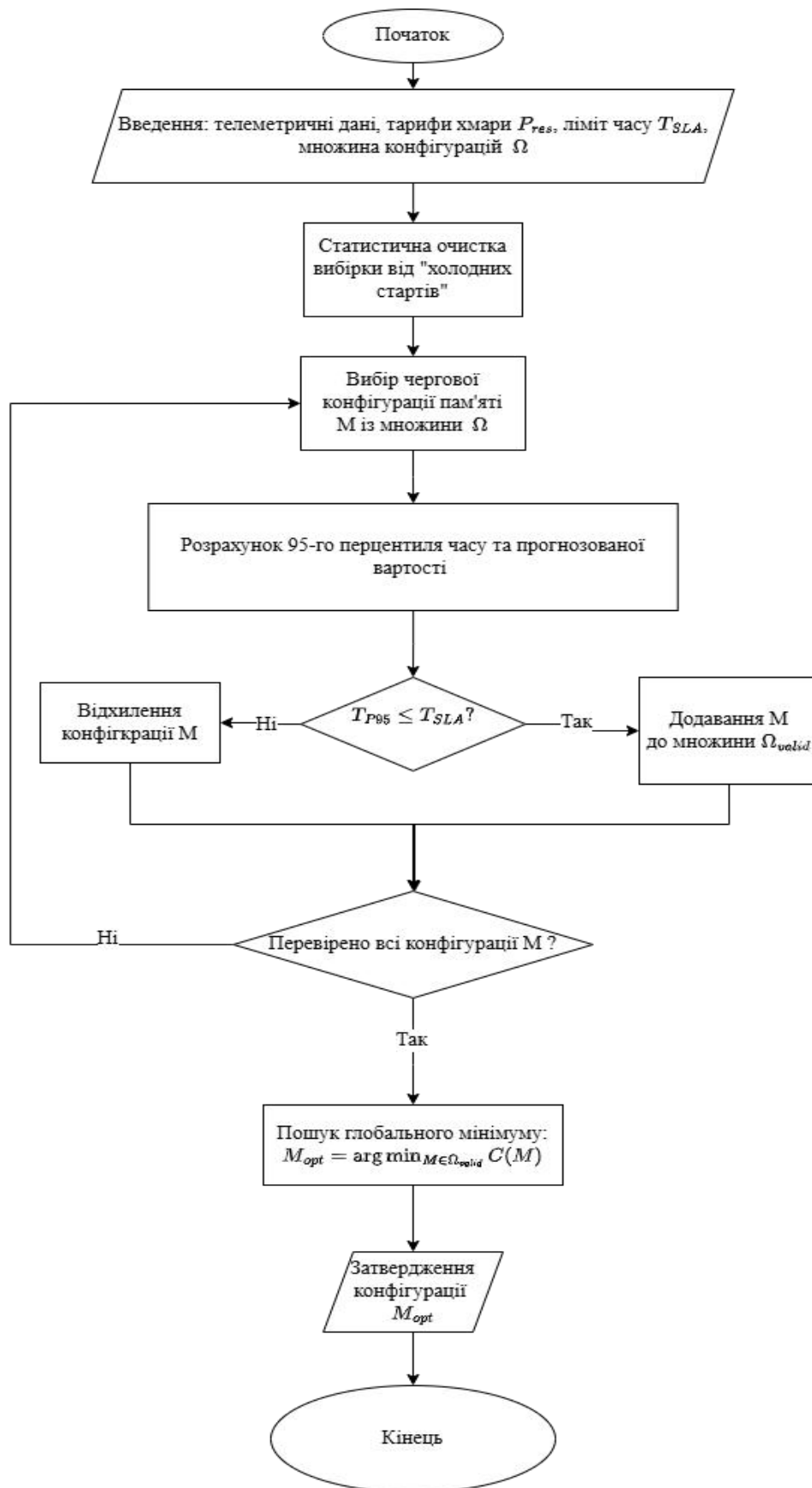


Рисунок 3.1 – Узагальнений алгоритм управління вартістю

3.2 Проєктування архітектури хмарної інфраструктури та процесу агрегації телеметричних даних

Для практичного застосування та валідації моделі управління вартістю, формалізованої у підрозділі 2.4, виникає необхідність у проєктуванні відповідної інфраструктури. Оскільки цільовою платформою дослідження було обрано екосистему Microsoft Azure, архітектура системи будується з використанням парадигми керованих сервісів. Такий підхід дозволяє усунути операційні витрати на адміністрування та конфігурацію віртуальних серверів, зосередившись виключно на процесах фінансової оптимізації.

Спроектowana інфраструктурна модель логічно розділена на три рівні, що взаємодіють між собою. Кожен з них виконує ізольовану функцію в загальному конвеєрі обслуговування трафіку інтернет-магазину та управління обчислювальними ресурсами.

Головним виконавчим ядром обчислювального рівня виступає Azure Container Apps, розгорнутий у рамках тарифу Consumption. На цьому рівні функціонує ізольований мікросервіс предметної області інтернет-магазину комп'ютерних ігор (модулі розрахунку регіональних знижок кошика та генерації унікальних ключів активації). Використання Azure Container Apps гарантує автоматичне горизонтальне масштабування ресурсів під час пікових навантажень, таких як сезонні розпродажі, а також дозволяє гнучко налаштовувати комбінації vCPU та оперативної пам'яті для кожної репліки контейнера.

До обчислювального рівня інтегровано сервіс моніторингу телеметрії Azure Application Insights для забезпечення процесів прийняття рішень достовірним масивом емпіричних даних. Цей інфраструктурний модуль виконує роль сховища логів та інструментарію профілювання продуктивності. Його ключовим завданням в рамках дослідження є безперервний збір даних

щодо тривалості обробки кожної HTTP-транзакції, ідентифікація холодних стартів та відстеження фактичного споживання оперативної пам'яті. Вилучення та структурування сирих даних здійснюється за допомогою аналітичної мови KQL (Kusto Query Language), що дозволяє формувати високоточні вибірки для програмної обробки.

Рівень прийняття управлінських рішень базується на застосування математичної моделі управління вартістю. Замість окремого програмного забезпечення, цей етап реалізується як аналітична процедура обробки даних. Алгоритм використовує експортовані метрики з рівня моніторингу, ключовою з яких є 95-перцентиль часу виконання транзакцій (T_{95}).

До отриманого масиву даних програма автоматично застосовує розроблену цільову функцію сукупної вартості з формули (2.4), циклічно перебираючи доступні конфігурації для програмного знаходження глобального економічного оптимуму.

Процес інформаційної взаємодії та маршрутизації потоків даних між рівнями має поетапний, циклічний характер. Ініціатором є вхідний клієнтський запит, який змушує обчислювальний рівень виділити ізольоване середовище із поточною конфігурацією обсягу пам'яті та vCPU. Метрики тривалості виконання обчислень транслуються на рівень телеметрії. Після того, як накопичиться статистично значуща вибірка, дані переносяться до консольної програми. Якщо результати розрахунку вказують на те, що поточні налаштування генерують фінансові збитки P_{loss} через погіршення швидкодії та порушення SLA, алгоритм інформує про необхідність коригувальної дії – зміни обсягу оперативної пам'яті для досягнення балансу продуктивності. Загальну організацію інфраструктури зображено на рисунку 3.2.

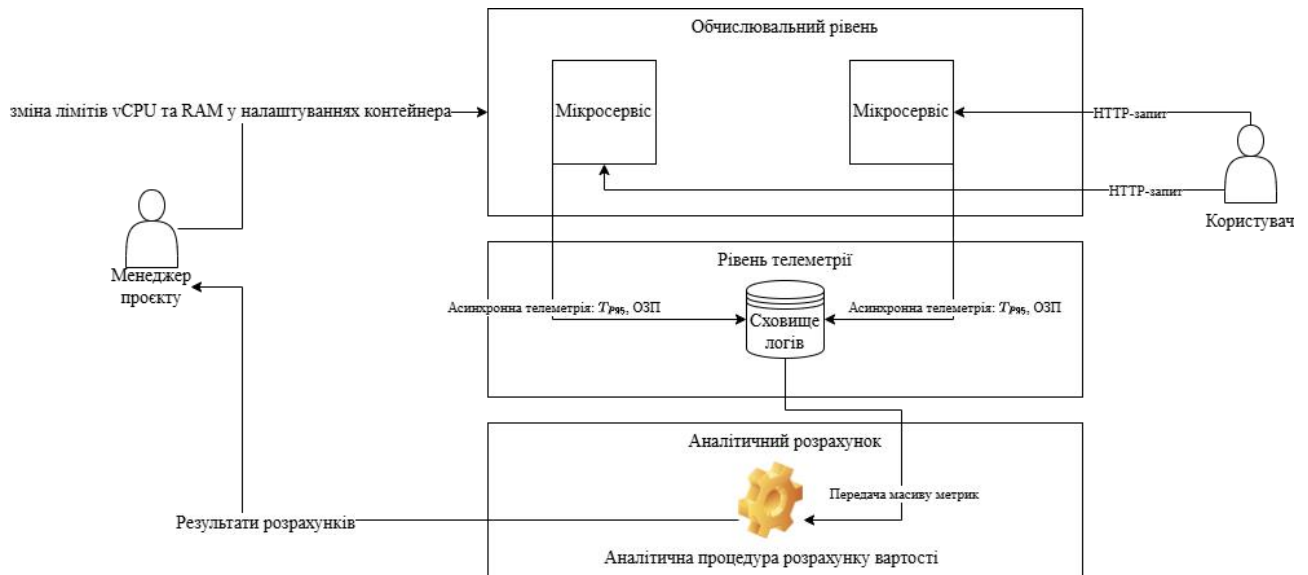


Рисунок 3.2 — Структурна схема хмарної інфраструктури та процесу фінансового контролю

Впровадження такого підходу дозволить відділити шлях проходження користувацького трафіку від зовнішніх фонових обчислень оптимізаційного модуля. Це формує надійне й повністю ізольоване середовище для проведення експериментального навантаження для дослідження.

3.3 Порядок впровадження розробленої процедури в управлінські процеси ІТ-проекту

Розроблена процедура створює інструментальний фундамент для фінансової оптимізації хмарних ресурсів. Проте, для досягнення максимальної економічної ефективності, модель управління має бути органічно інтегрована в життєвий цикл розробки програмного забезпечення інтернет-магазину та стати складовою регулярних процесів управління.

У контексті сучасних методологій управління, таких як Agile чи Scrum, процедуру оптимізації доцільно впроваджувати на етапі тестування та

підготовки до релізу. Особливість безсерверних архітектур полягає у високому впливі змін в програмному коді на час виконання транзакцій, що може суттєво впливати на 95-й перцентиль T_{P95} та сукупну вартість C_{Σ} .

Процедура впроваджується у пайплайн безперервної інтеграції та розгортання у формі послідовного алгоритму. Після випуску версії мікросервісу у тестове середовище викликається автоматизований скрипт навантажувального тестування для симуляції пікового користувацького трафіку. Далі зібрані телеметричні дані аналізуються за гібридною моделлю для формування рекомендації щодо оптимальної потужності vCPU та обсягу пам'яті. На фінальному етапі менеджер проєкту або спеціаліст з фінансових операцій аналізує отримані результати розрахунків. У випадку, якщо запропонована конфігурація призведе до дотримання SLA без перевитрат бюджету, ухвалює управлінське рішення про затвердження її для розгортання у виробниче середовище.

При цьому роль менеджера IT-проєкту переходить з пасивного спостерігача за хмарними витратами в активного учасника процесу оптимізації. Менеджер встановлює бізнес-обмеження: визначає максимально допустимий час обробки транзакції T_{SLA} залежно від критичності функціонального модуля та формує економічний розмір штрафу P_{loss} на основі аналізу показників конферсії магазину ігор.

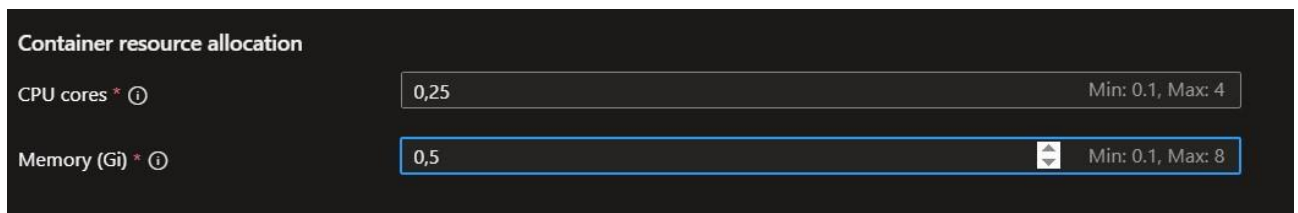
У підсумку розроблена процедура перетворюється з ізольованого математичного розрахунку на безперервний управлінський процес, тобто реалізує практику FinOps. Це забезпечує попередній контроль над бюджетом проєкту, запобігаючи виникненню непередбачуваних фінансових збитків після запуску нових функцій інтернет-магазину.

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ МОДЕЛІ УПРАВЛІННЯ ВАРТІСТЮ

4.1. Мета та умови проведення експерименту

Метою дослідження є практична перевірка розробленої гібридної SLA-орієнтованої моделі управління вартістю, що була розроблена у попередніх розділах на базі реальної хмарної інфраструктури. Для проведення експерименту було розгорнуто ізольований мікросервісний додаток API у середовищі Microsoft Container Apps на регіоні північна Європа.

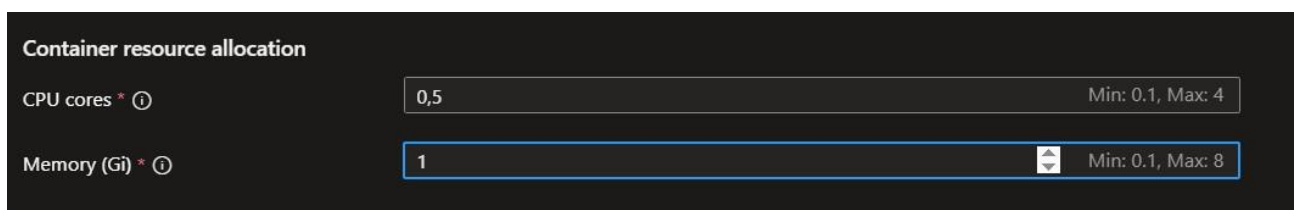
Для оцінки ефективності використання ресурсів було обрано чотири конфігурації контейнера Ω . В рамках роботи конфігурації називаються: M_1 (Слабка), M_2 (Базова), M_3 (Потужна) та M_4 (Надмірна). Найслабші налаштування включають 0.25 vCPU, 0.5 GiB RAM (рисунок 4.1).



The screenshot shows a dark-themed interface titled "Container resource allocation". It contains two input fields. The first is labeled "CPU cores * ⓘ" and has a value of "0,25" with a range of "Min: 0.1, Max: 4". The second is labeled "Memory (Gi) * ⓘ" and has a value of "0,5" with a range of "Min: 0.1, Max: 8".

Рисунок 4.1 – Мінімальна конфігурація vCPU і RAM

Базова конфігурація має 0.5 vCPU, 1.0 GiB RAM (рисунок 4.2)



The screenshot shows the same "Container resource allocation" interface. The "CPU cores" field now has a value of "0,5" with a range of "Min: 0.1, Max: 4". The "Memory (Gi)" field now has a value of "1" with a range of "Min: 0.1, Max: 8".

Рисунок 4.2 – Базова конфігурація vCPU і RAM

Потужна конфігурація: 2.0 vCPU, 4.0 GiB RAM (рисунок 4.3).

Container resource allocation

CPU cores * ⓘ Min: 0.1, Max: 4

Memory (Gi) * ⓘ Min: 0.1, Max: 8

Рисунок 4.3 – Потужна конфігурація vCPU і RAM

Надлишкова конфігурація контейнера має ж максимально можливі значення 4.0 vCPU, 8.0 GiB RAM (рисунок 4.4).

Container resource allocation

CPU cores * ⓘ Min: 0.1, Max: 4

Memory (Gi) * ⓘ Min: 0.1, Max: 8

Рисунок 4.4 – Надлишкова конфігурація vCPU і RAM

Згідно з бізнес-вимогами, максимально допустимий час обробки запиту (SLA) було встановлено на рівні $T_{SLA} = 1.0$ с (1000мс). Для збору телеметричних даних застосовувався інструмент навантажувального тестування Grafana k6. Тестовий сценарій імітував паралельні запити 25 користувачів, що робили сумарно 1000 транзакцій для кожної з конфігурацій

Лістинг 4.1 – Скрипт навантажувального тестування у Grafana k6

```
import http from "k6/http";
import { check, sleep } from "k6";
export const options = {
  vus: 25,
  iterations: 1000,
};
export default function () {
```

```

const url = "https://gamestore-api.ashydesert-
94f7f95c.northeurope.azurecontainerapps.io/api/CalculateRegion
alBundlePrice";

const res = http.get(url);
check(res, {
  "is status 200": (r) => r.status === 200,
});
sleep(0.1);
}

```

4.2 Агрегація метрик та визначення 95-го процентилля

Базовою метрикою для оцінки швидкодії є 95-й перцентиль часу виконання запиту T_{p95} . Під час практичного дослідження цей показник не обчислювався вручну. Збір тисяч сирих телеметричних даних та розрахунок фінального значення T_{p95} виконувалися автоматично за допомогою вбудованих аналітичних алгоритмів інструменту навантажувального тестування Grafana k6

Після завершення кожної ітерації тестування k6 генерував консольний звіт, з якого бралось готове значення 95-го перцентилля для використання у розробленій моделі. Результати тестування кожної конфігурації зафіксували закономірну стрімку нелінійну деградацію швидкодії при зменшенні виділених ресурсів. Результати навантаження представлені на рисунках (4.5) – (4.8).

```

TOTAL RESULTS

checks_total.....: 1000    17.958114/s
checks_succeeded...: 100.00% 1000 out of 1000
checks_failed.....: 0.00%   0 out of 1000

✓ is status 200

HTTP
http_req_duration.....: avg=1.25s min=162.94ms med=480.76ms max=9.83s p(90)=3.14s p(95)=5.18s
{ expected_response:true }...: avg=1.25s min=162.94ms med=480.76ms max=9.83s p(90)=3.14s p(95)=5.18s
http_req_failed.....: 0.00%   0 out of 1000
http_reqs.....: 1000    17.958114/s

EXECUTION
iteration_duration.....: avg=1.35s min=263.83ms med=582.27ms max=9.93s p(90)=3.3s p(95)=5.28s
iterations.....: 1000    17.958114/s
vus.....: 9      min=9      max=25
vus_max.....: 25    min=25    max=25

NETWORK
data_received.....: 262 kB 4.7 kB/s
data_sent.....: 82 kB 1.5 kB/s

```

Рисунок 4.5 – Результати тестування конфігурації M_1 : $T_{p95} = 5.18$ с

```

TOTAL RESULTS

checks_total.....: 1000    33.654886/s
checks_succeeded...: 100.00% 1000 out of 1000
checks_failed.....: 0.00%   0 out of 1000

✓ is status 200

HTTP
http_req_duration.....: avg=614.72ms min=93.46ms med=200.84ms max=4.6s p(90)=1.75s p(95)=3.09s
{ expected_response:true }...: avg=614.72ms min=93.46ms med=200.84ms max=4.6s p(90)=1.75s p(95)=3.09s
http_req_failed.....: 0.00%   0 out of 1000
http_reqs.....: 1000    33.654886/s

EXECUTION
iteration_duration.....: avg=719.04ms min=194.45ms med=301.44ms max=4.7s p(90)=1.85s p(95)=3.19s
iterations.....: 1000    33.654886/s
vus.....: 11     min=11     max=25
vus_max.....: 25    min=25    max=25

NETWORK
data_received.....: 257 kB 8.7 kB/s
data_sent.....: 82 kB 2.8 kB/s

```

Рисунок 4.6 – Результати тестування конфігурації M_2 : $T_{p95} = 3.09$ с

```

TOTAL RESULTS

checks_total.....: 1000    93.404529/s
checks_succeeded...: 100.00% 1000 out of 1000
checks_failed.....: 0.00%   0 out of 1000

✓ is status 200

HTTP
http_req_duration.....: avg=155.69ms min=93.42ms med=130.61ms max=679.51ms p(90)=238.22ms p(95)=293.49ms
{ expected_response:true }...: avg=155.69ms min=93.42ms med=130.61ms max=679.51ms p(90)=238.22ms p(95)=293.49ms
http_req_failed.....: 0.00%   0 out of 1000
http_reqs.....: 1000    93.404529/s

EXECUTION
iteration_duration.....: avg=265.14ms min=194.02ms med=231.1ms max=859.27ms p(90)=353.49ms p(95)=445.82ms
iterations.....: 1000    93.404529/s
vus.....: 25      min=25      max=25
vus_max.....: 25      min=25      max=25

NETWORK
data_received.....: 251 kB 24 kB/s
data_sent.....: 82 kB 7.7 kB/s

```

Рисунок 4.7 – Результати тестування конфігурації M_3 : $T_{p95} = 0.29$ с

```

TOTAL RESULTS

checks_total.....: 1000    107.679353/s
checks_succeeded...: 100.00% 1000 out of 1000
checks_failed.....: 0.00%   0 out of 1000

✓ is status 200

HTTP
http_req_duration.....: avg=125.45ms min=93.57ms med=114.19ms max=461.13ms p(90)=157.49ms p(95)=182.65ms
{ expected_response:true }...: avg=125.45ms min=93.57ms med=114.19ms max=461.13ms p(90)=157.49ms p(95)=182.65ms
http_req_failed.....: 0.00%   0 out of 1000
http_reqs.....: 1000    107.679353/s

EXECUTION
iteration_duration.....: avg=230.01ms min=194.12ms med=214.84ms max=561.33ms p(90)=258.47ms p(95)=285.96ms
iterations.....: 1000    107.679353/s
vus.....: 25      min=25      max=25
vus_max.....: 25      min=25      max=25

NETWORK
data_received.....: 251 kB 27 kB/s
data_sent.....: 82 kB 8.9 kB/s

```

Рисунок 4.8 – Результати тестування конфігурації M_4 : $T_{p95} = 0.18$ с

4.3 Технічна фільтрація конфігурацій за критеріями SLA

Початковий етап розробленої процедури вимагає відсіяти технічно непридатні конфігурації обчислювальних ресурсів. Для цього використовується математична умова формування підмножини допустимих значень Ω_{valid} , яку було формалізовано (3.2). Згідно з бізнес-вимогами до системи електронної комерції, максимально допустимий час відгуку T_{SLA} встановлено на рівні 1 секунди.

На основі емпіричних значень 95-го перцентилля, отриманих у ході навантажувального тестування (рисунки 4.5-4.8), перевірено відповідність кожного варіанту встановленому ліміту швидкодії. Результати фільтрування наведено в таблиці 4.1.

Таблиця 4.1 – Результати фільтрування конфігурацій

Конфігурація	Ресурси (vCPU/RAM)	T_{p95} , с	Статус
M_1 (Слабка)	0.25 vCPU / 0.5 ГБ	5.18	Відхилено
M_2 (Базова)	0.50 vCPU / 1.0 ГБ	3.09	Відхилено
M_3 (Потужна)	2.00 vCPU / 4.0 ГБ	0.29	Допущено
M_4 (Надмірна)	4.00 vCPU / 8.0 ГБ	0.18	Допущено

Як видно з результатів, конфігурації M_1 та M_2 сильно порушують обмеження Ω_{valid} , створюючи затримку, що у 3-5 разів перевищують цільовий показник SLA. Це призводить до їх виключення з подальшого розгляду. До

підмножини допустимих значень, придатних до використання, увійдуть тільки конфігурації M_3 та M_4 .

4.4 Знаходження глобального економічного оптимуму та аналіз сукупної вартості володіння

Для визначення економічно найвигіднішого варіанту налаштувань серед наявної множини виконаємо розрахунок сукупної вартості володіння C_{Σ} . Обчислення здійснюються за гібридною моделлю вартістю, що орієнтована на SLA (формула 2.5).

Для проведення розрахунків використано параметри: фінансовий еквівалент збитку $C_{delay} = \$10$ за кожну секунду затримки понад ліміт T_{SLA} (становить 1 секунду); кількість пікових годин на місяць $N_{peak} = 100$. Прямі інфраструктурні витрати розраховано за формулою (2.1) на основі офіційних тарифів Azure Container Apps ($P_{vCPU} = \$0.000034/c$, $P_{Ram} = \$0.000004/c$) для середнього календарного місяця (2628000 с).

Результати обчислення сукупної вартості показали варіативність економічної ефективності, що залежить від виділених ресурсів. Так, для конфігурації M_1 (0.25 vCPU/0.5 ГБ) прямі інфраструктурні витрати C_{cloud} становлять лише \$27.59, проте через критичну затримку відгуку (5.18 с) фінансовий штраф P_{loss} сягає \$4180.00. В результаті цього загальна сукупна вартість C_{Σ} занадто висока і дорівнює \$4207.59. Аналогічна ситуація спостерігається і для конфігурації M_2 (0.5 vCPU/1 ГБ): при вартості хмарної інфраструктури \$55.19 штраф за порушення SLA становить \$2090.00, утворюючи сукупну вартість \$2145.19.

Натомість конфігурація M_3 , навпроти, при пропорційному зростанні прямих витрат на інфраструктуру $C_{cloud} = \$220.75$, забезпечує повне

дотримання цільового SLA ($T_{p95} \leq 1.0$ с). Оскільки фінансові штрафи P_{loss} у цьому випадку не застосовуються, сукупна вартість дорівнює вартості самої інфраструктури. Використання конфігурації M_4 (4.0 vCPU/8.0 ГБ) також гарантує відсутність штрафів, однак прямі витрати на інфраструктуру C_{cloud} зростають до \$441.50, що робить її використання економічно не вигідним.

Аналіз економічної ефективності наведено в таблиці 4.2.

Таблиця 4.2 – Зведені результати FinOps-оптимізації за розробленою моделлю

Конфігурація	Прямі витрати (C_{cloud}), дол.	Штраф (P_{loss}), дол.	Сукупна вартість (C_{Σ}), дол.
Слабка (M_1)	27.59	4180.00	4207.59
Базова (M_2)	55.19	2090.00	2145.19
Потужна (M_3)	220.75	0.00	220.75
Надмірна (M_4)	441.50	0.00	441.50

Графічне представлення таблиці можна прослідкувати на графіку (рисунок 4.9)

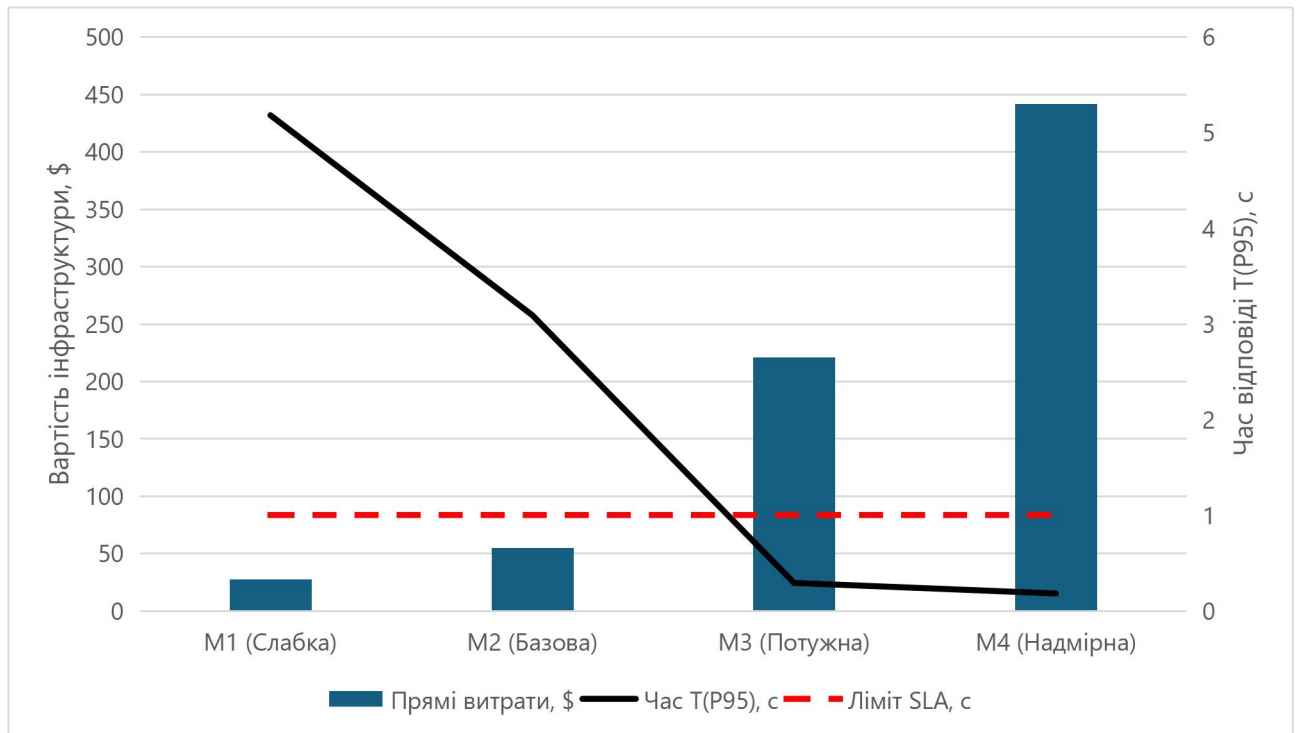


Рисунок 4.9 – Кореляція між продуктивністю та вартістю хмарних ресурсів

Застосування алгоритму пошуку глобального мінімуму згідно з формулою (3.3) математично доводить, що конфігурація M_3 (2.0 vCPU/4.0 ГБ) є точкою економічного оптимуму. Хоча прямі інфраструктурні витрати на цю конфігурацію вищі за попередні варіанти, відсутність бізнес-штрафів робить її вартість володіння найнижчою серед усіх альтернатив. Водночас подальше збільшення ресурсів до рівня M_4 веде до подвоєння бюджету без суттєвого виграшу в показниках SLA.

ВИСНОВКИ

У магістерській кваліфікаційній роботі розв'язано актуальне науково-прикладне завдання щодо підвищення ефективності хмарних Serverless архітектур систем електронної комерції шляхом розроблення та впровадження SLA-орієнтованої моделі управління вартістю. Проведений аналіз механізмів тарифікації та масштабування в сучасних Serverless-платформах дозволив виявити, що підходи до економії, які полягають у виборі найдешевших інфраструктурних конфігурацій, у високонавантажених системах призводять до значного зниження швидкодії та генерують значні приховані збитки. На основі цього було розроблено гібридну математичну FinOps-модель оптимізації витрат, наукова новизна якої полягає у переході до розрахунку сукупної вартості володіння шляхом інтеграції показників швидкодії та фінансових ризиків від порушення угод про рівень обслуговування в єдину цільову функцію мінімізації витрат.

У межах дослідження було проведено моделювання та експериментальний аналіз поведінки ізольованого мікросервісу, розгорнутого у Microsoft Azure Container Apps. Це дозволило отримати точні дані щодо споживання ресурсів та швидкодії системи без необхідності розгортання додаткових програмних модулів. Практична валідація моделі за допомогою навантажувального тестування інструментом Grafana k6 підтвердила нелінійну залежність між обсягом виділених ресурсів та часом обробки запитів. Застосування розробленого алгоритму технічної фільтрації дало змогу математично обґрунтувати відхилення конфігурацій, що порушували встановлену межу SLA у 1 секунду, та обрати найбільш ефективний варіант налаштувань обчислювальних ресурсів.

Результати проведеного моделювання дозволили виявити, що потужніша й дорожча конфігурація (2.0 vCPU) забезпечує зниження фактичних витрат на

виконання коду порівняно зі слабкими аналогами завдяки суттєвому скороченню тарифікованого часу. Практична значимість отриманих результатів полягає у створенні підґрунтя для проактивного управління хмарними бюджетами, що дозволяє ІТ-командам приймати аргументовані рішення щодо масштабування систем, усуваючи явища надлишкового та недостатнього забезпечення ресурсами в межах методології FinOps.

За результатами дипломної роботи опубліковано тези Матюшко Є.Д., Міхнова О.Д. Тестування моделі управління вартістю проєкту на безсерверних технологіях. Матеріали Всеукраїнської науково-методичної конференції «Проблеми математичного моделювання», Кам'янське, 19-21 травня 2026. Матеріали конференції розміщені на сайті Дніпровського державного технічного університету (www.dstu.dp.ua).

Пояснювальна записка до кваліфікаційної роботи оформлена згідно ДСТУ та методичними вказівками щодо розробки та оформлення кваліфікаційної роботи [25-27].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Architecture of next-generation e-commerce platform / Y. Huang et al. Tsinghua science and technology. 2019. Vol. 24, no. 1. P. 18–29. DOI: <https://doi.org/10.26599/tst.2018.9010067>.
2. Perri D., Simonetti M., Gervasi O. Deploying efficiently modern applications on cloud. Electronics. 2022. Vol. 11, no. 3. P. 450. DOI: <https://doi.org/10.3390/electronics11030450>.
3. Trends in the development of web computing in e-commerce / Н. Чаплинська та ін. Periodicals of engineering and natural sciences (PEN). 2025. Vol. 13, no. 1. P. 233–248. DOI: <https://doi.org/10.21533/pen.v13.i1.250>.
4. Braun I., Hoffmann M., Mörseburg R. Implementation of a web-based audience response system as microservice application vs. monolithic application. 18th International Conference on WWW/Internet 2019. 2019. P. 25–32.
5. FanYon K., Chua F.-F., Lim A. H.-L. Resource cost management in cloud service environment. Bulletin of electrical engineering and informatics. 2026. Vol. 15, no. 1. P. 470–483. DOI: <https://doi.org/10.11591/eei.v15i1.10352> (date of access: 08.05.2026).
6. Ramesh A., Pradhan V., Lamkuche H. Understanding and analysing resource utilization, costing strategies and pricing models in cloud computing. Journal of physics: conference series. 2021. Vol. 1964, no. 4. P. 042049. DOI: <https://doi.org/10.1088/1742-6596/1964/4/042049>.
7. Adzic G., Chatley R. Serverless computing: economic and architectural impact. Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2017). New York, NY, USA : Association for Computing Machinery, 2017. P. 884–889. DOI: <https://doi.org/10.1145/3106237.3117767>.
8. FanYon K., Chua F.-F., Lim A. H.-L. Resource cost management in cloud service environment. Bulletin of electrical engineering and informatics. 2026. Vol. 15, no. 1. P. 470–483. DOI: <https://doi.org/10.11591/eei.v15i1.10352>.

9. Cho K. W., Bahn H. A Cost Estimation Model for Cloud Services and Applying to PC Laboratory Platforms. *Processes*. 2020. Vol. 8, No. 1. Article 76.
10. Böhme L., Beckmann T., Baltes S., Hirschfeld R. A Penny a Function: Towards Cost Transparent Cloud Programming. *Proceedings of the 2nd ACM SIGPLAN International Workshop on Programming Abstractions and Interactive Notations, Tools, and Environments (PAINT 2023)*. New York, NY, USA : Association for Computing Machinery, 2023. P. 1–10. DOI: <https://doi.org/10.1145/3623504.3623566>.
11. The empirical reality of IT project cost overruns: discovering A power-law distribution / B. Flyvbjerg et al. *Journal of management information systems*. 2022. Vol. 39, no. 3. P. 607–639. DOI: <https://doi.org/10.1080/07421222.2022.2096544>.
12. Ramchand K., Baruwat Chhetri M., Kowalczyk R. Enterprise adoption of cloud computing with application portfolio profiling and application portfolio assessment. *Journal of cloud computing*. 2021. Vol. 10, no. 1. DOI: <https://doi.org/10.1186/s13677-020-00210-w>.
13. Fine-Grained performance and cost modeling and optimization for faas applications / C. Lin et al. *IEEE transactions on parallel and distributed systems*. 2022. P. 1–15. DOI: <https://doi.org/10.1109/tpds.2022.3214783>.
14. Poudel B., Pandey S. A study on cost optimization in faas computing with the most favorable resource size. *International journal of research publications*. 2024. Vol. 50, no. 1. DOI: <https://doi.org/10.47119/ijrp1001501620246653>.
15. Apostolopoulos P. A., Tsiropoulou E. E., Papavassiliou S. Risk-Aware Social Cloud Computing Based on Serverless Computing Model. *2019 IEEE Global Communications Conference (GLOBECOM)*. Waikoloa, HI, USA, 2019. P. 1–6. DOI: <https://doi.org/10.1109/GLOBECOM38437.2019.9013182>.
16. Develop modern cloud applications with .NET on Azure | .NET. Microsoft. URL: <https://dotnet.microsoft.com/en-us/apps/cloud/azure> (дата звернення: 11.04.2026).

17. Application Insights OpenTelemetry observability overview - Azure Monitor. Microsoft Learn: Build with answers in reach. URL: <https://learn.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview?tabs=webapps> (дата звернення: 11.04.2026).
18. Azure Container Apps pricing. URL: <https://azure.microsoft.com/en-us/pricing/details/container-apps/> (дата звернення: 13.04.2026).
19. Eismann S., Grohmann J., van Eyk E., Herbst N., Kounev S. Predicting the Costs of Serverless Workflows. Proceedings of the ACM/SPEC International Conference on Performance Engineering (ICPE '20). New York, NY, USA : Association for Computing Machinery, 2020. P. 265–276. DOI: <https://doi.org/10.1145/3358960.3379133>.
20. A survey of cost optimization in serverless cloud computing / Z. Li et al. Journal of physics: conference series. 2021. Vol. 1802, no. 3. P. 032070. DOI: <https://doi.org/10.1088/1742-6596/1802/3/032070>.
21. Reducing cold-start time on azure container apps. Microsoft Learn: Build with answers in reach. URL: <https://learn.microsoft.com/en-us/azure/container-apps/cold-start> (дата звернення: 13.04.2026).
22. Wang L., Li M., Zhang Y., Ristenpart T., Swift M. Peeking behind the curtains of serverless platforms. 2018 USENIX Annual Technical Conference (USENIX ATC 18). 2018. P. 133-146.
23. Guide for running C# Azure Functions in an isolated worker process. Microsoft Learn. URL: <https://learn.microsoft.com/en-us/azure/azure-functions/dotnet-isolated-process-guide> (дата звернення: 13.04.2026).
24. Azure Functions Premium plan hosting. Microsoft Learn. URL: <https://learn.microsoft.com/en-us/azure/azure-functions/functions-premium-plan> (дата звернення: 13.04.2026).
25. ДСТУ 3008:2015 "Звіти у сфері науки і техніки. Структура та правила оформлення". Київ: Держстандарт України, 2017. 31 с.

26. ДСТУ 8302:2015 «Бібліографічне посилання. Загальні положення та правила складання». Київ: Держстандарт України, 2017. 20 с.

27. Методичні вказівки щодо розробки та оформлення кваліфікаційної роботи другого (магістерського) рівня вищої освіти за освітньо-науковою програмою «Управління проектами в галузі інформаційних технологій» / упоряд.: К. Е. Петров та ін. Харків : ХНУРЕ, 2024. 24 с.