ДОДАТОК А

Слайди презентації

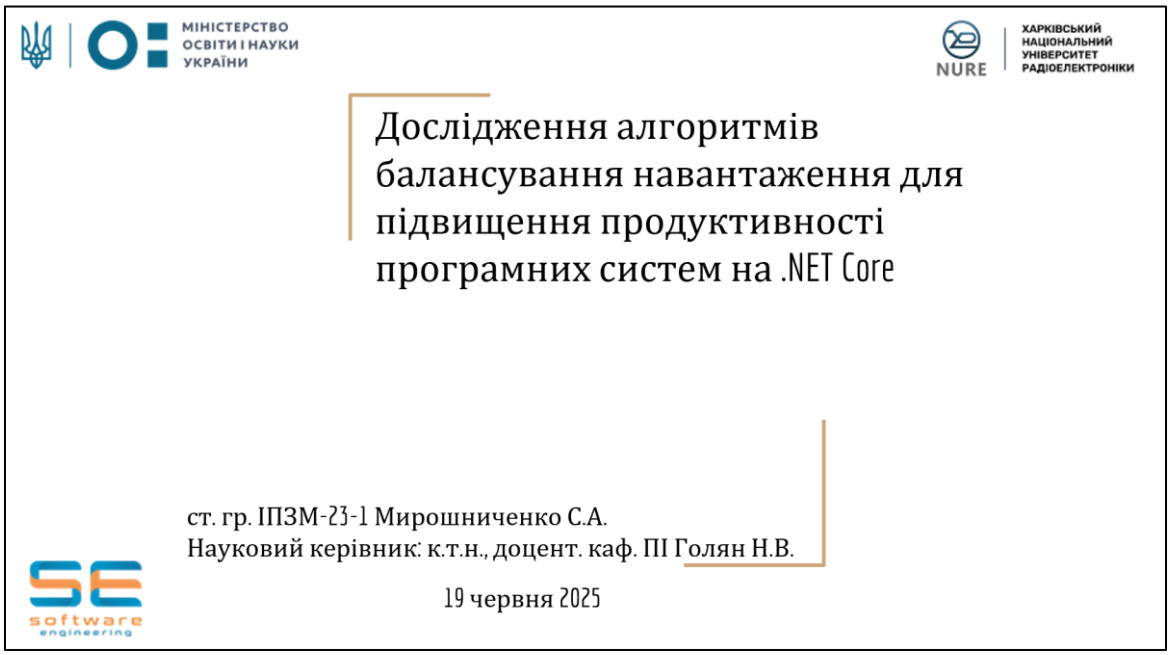


Рисунок А.1 – 1 слайд (рисунок виконано самостійно).

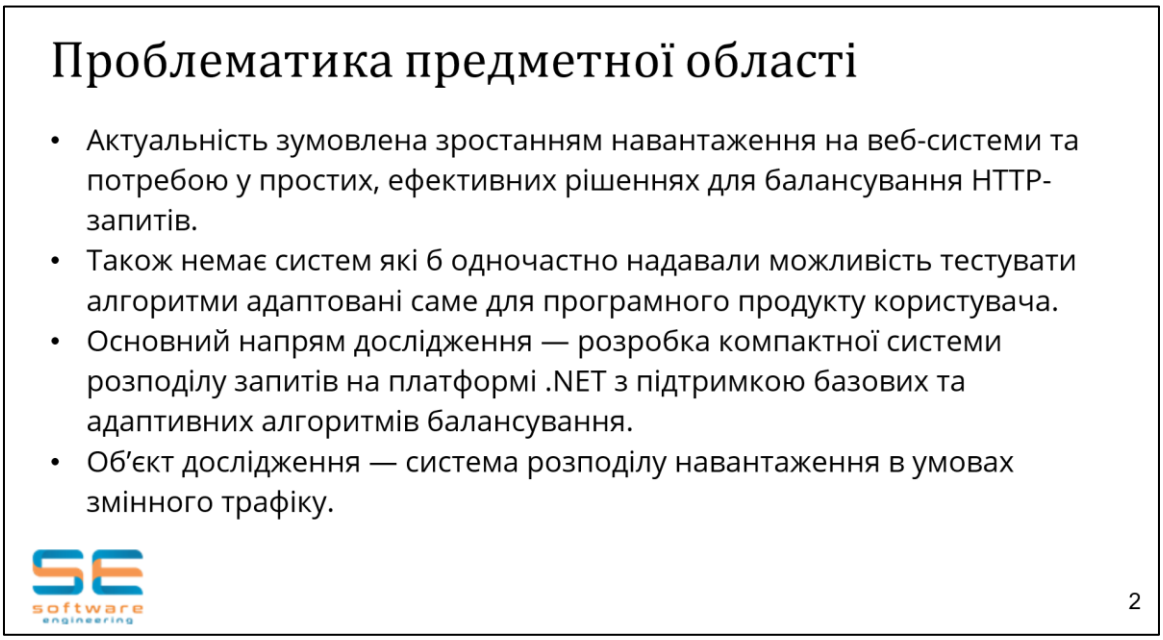


Рисунок А.2 – 2 слайд (рисунок виконано самостійно).

Рисунок А.3 – 3 слайд (рисунок виконано самостійно).

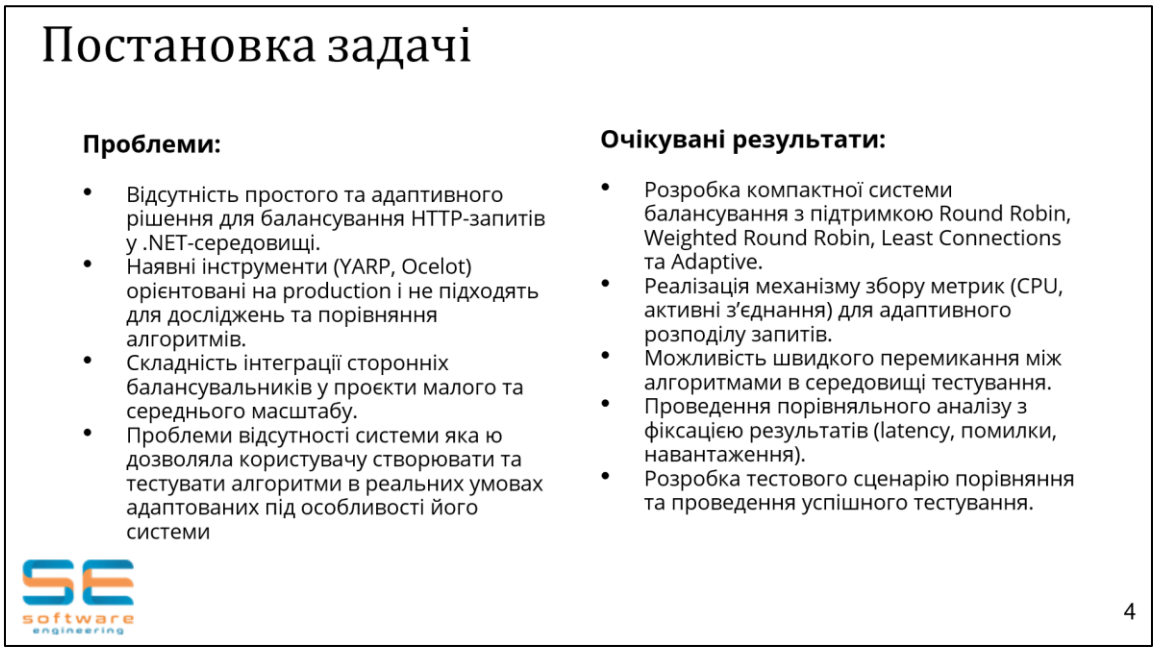


Рисунок А.4 – 4 слайд (рисунок виконано самостійно).

Методологія дослідження

Використані методи дослідження:

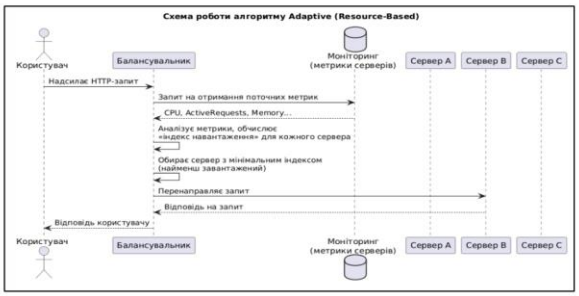
- **Порівняльний аналіз** — для оцінки ефективності алгоритмів балансування (latency, розподіл навантаження, кількість помилок).
- **Емпіричне тестування** — симуляція реального навантаження з різними сценаріями.
- **Збір та статистична обробка метрик** — для забезпечення достовірності та репрезентативності результатів.

5

Рисунок А.5 – 5 слайд (рисунок виконано самостійно).

Алгоритми

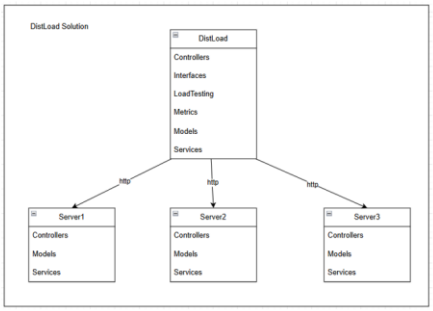
- Round Robin – рівномірний розподіл запитів між серверами у циклічному порядку, незалежно від їхнього поточного стану;
- Weighted Round Robin – модифікація попереднього методу, яка враховує вагові коефіцієнти серверів, дозволяючи розподіляти більше запитів на більш потужні вузли;
- Least Connections – підхід, який орієнтується на кількість активних з'єднань, спрямовуючи нові запити до того сервера, який має найменше активних підключень у даний момент;
- Adaptive (Resource-Based) – алгоритм, що використовує актуальні дані про стан серверів для прийняття рішень про маршрутизацію запитів, забезпечуючи динамічне та більш гнучке балансування.



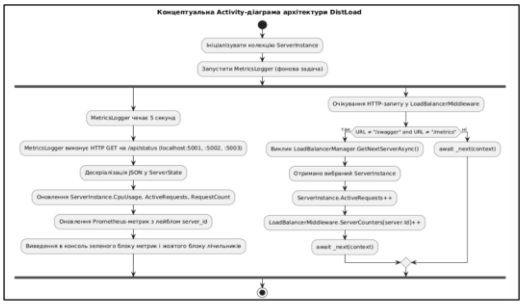
6

Рисунок А.6 – 6 слайд (рисунок виконано самостійно).

Архітектура система



Концептуальна архітектура системи



Базовий пайплайн виконання потоку

software engineering

7

Рисунок А.7 – 7 слайд (рисунок виконано самостійно).

Файлова система проекту

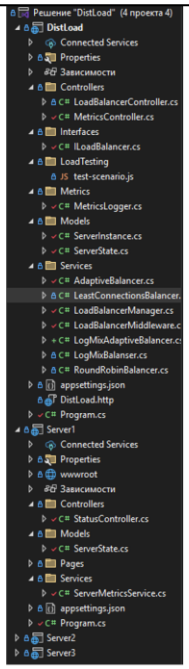
Структура проєкту побудована для забезпечення модульності, розширюваності та легкості тестування:

DistLoad — головний проєкт-балансувальник:

Controllers — обробка запитів та метрик.
Services — реалізація алгоритмів балансування.
Models — моделі стану серверів і метрик.
Interfaces — контракт для алгоритмів.
Metrics — збір метрик із серверів.
LoadTesting — скрипти для K6.

Server1 / Server2 / Server3 — імітаційні сервери з єдиною структурою:

Controllers — статус сервера.
Models — поточний стан (CPU, з'єднання).
Services — обробка логіки навантаження.



software engineering

8

Рисунок А.8 – 8 слайд (рисунок виконано самостійно).

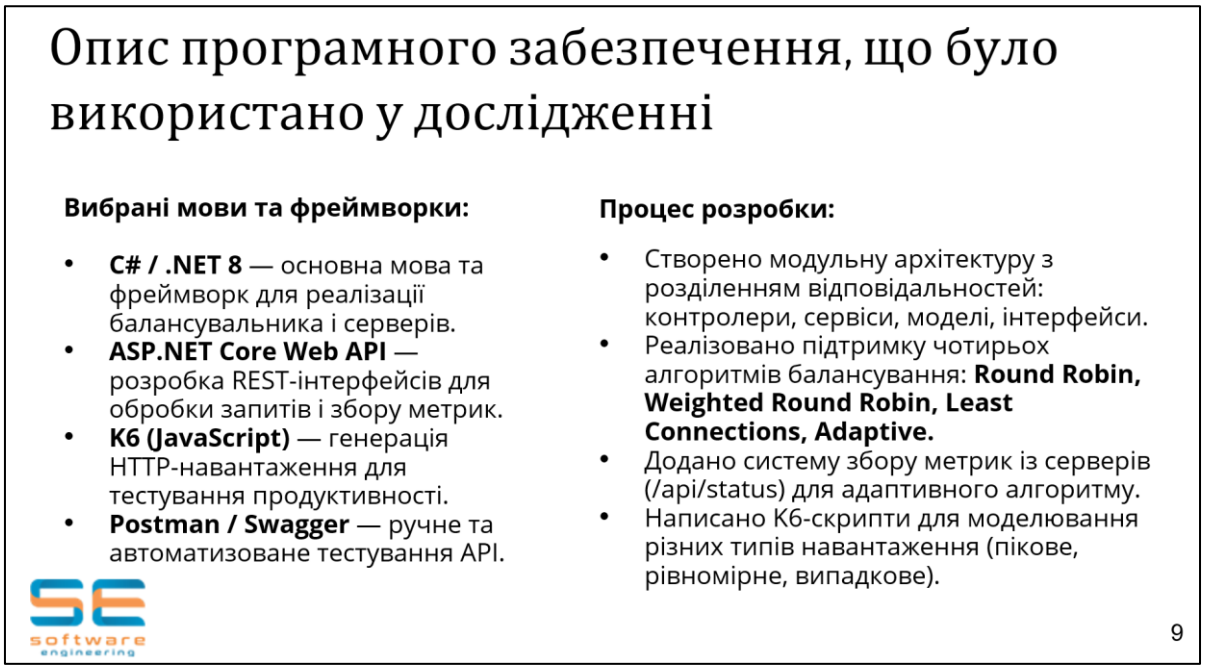


Рисунок А.9 – 9 слайд (рисунок виконано самостійно).

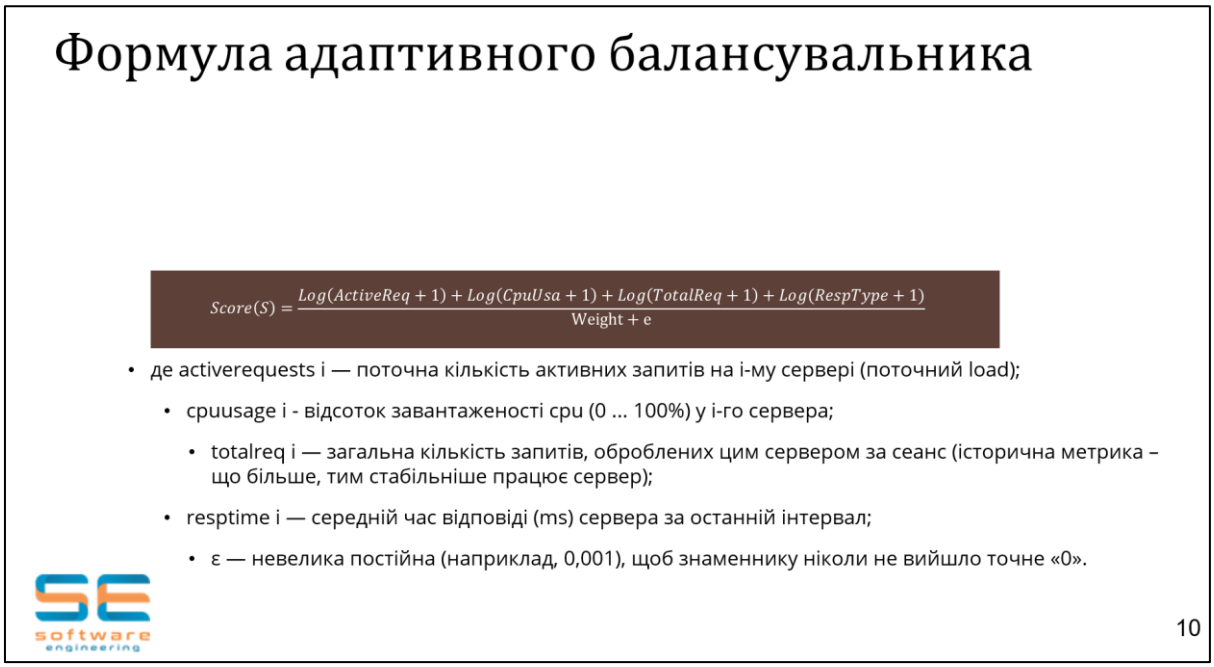


Рисунок А.10 – 10 слайд (рисунок виконано самостійно).

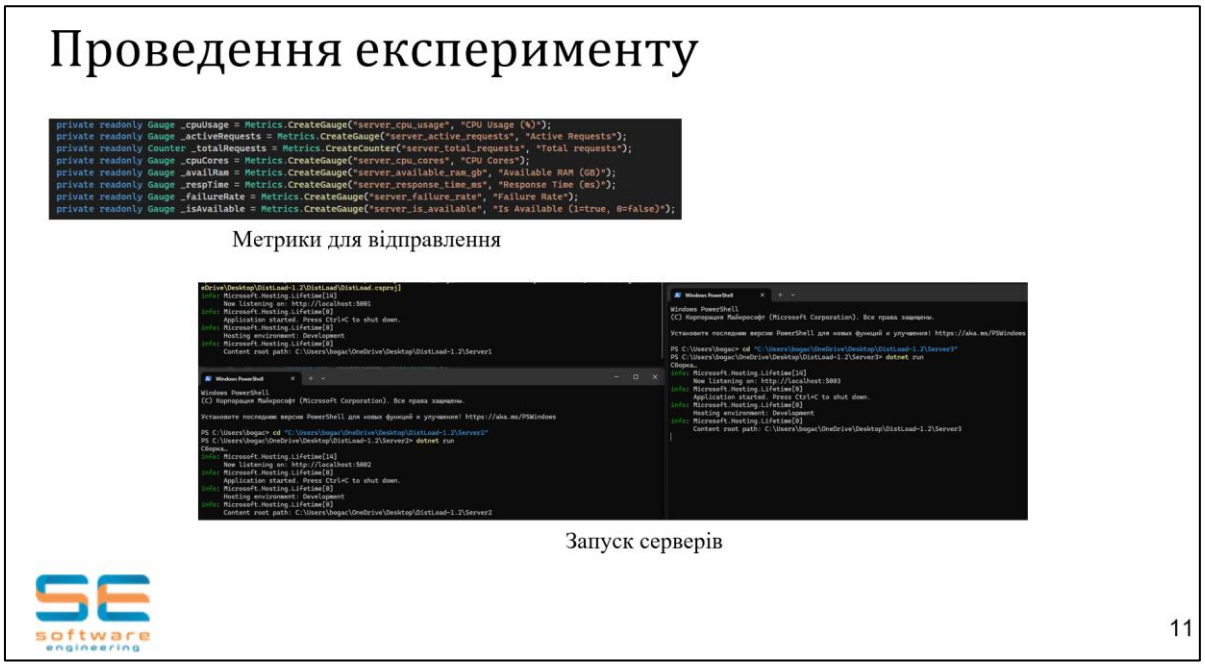


Рисунок А.11 – 11 слайд (рисунок виконано самостійно).

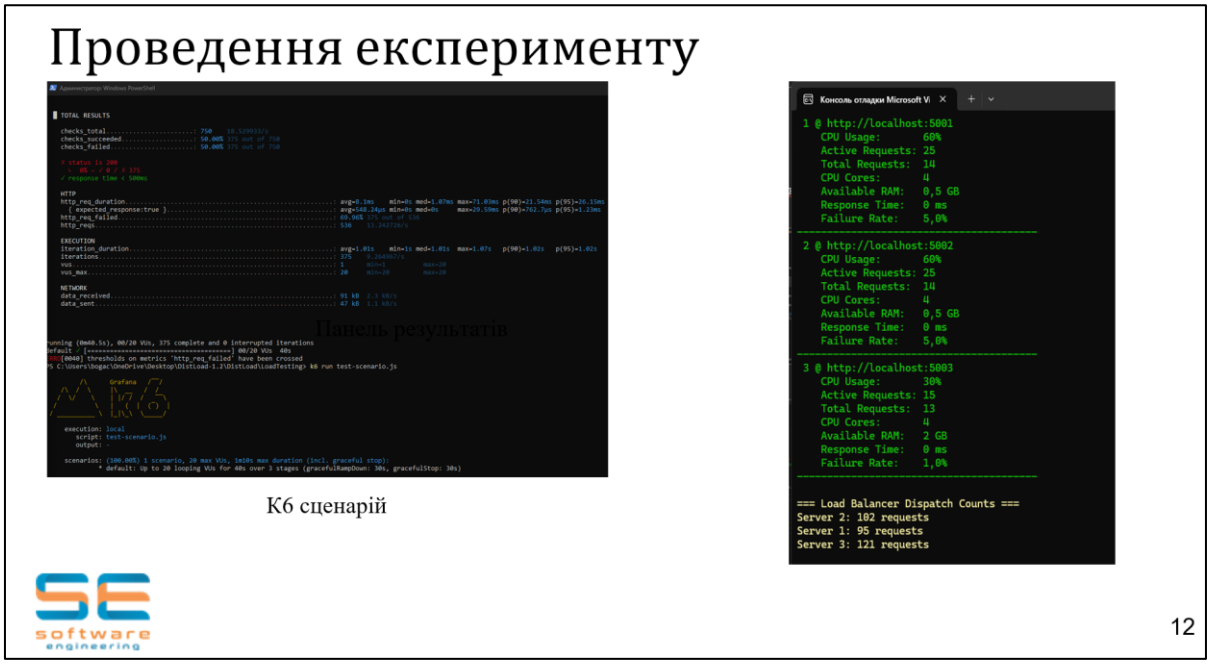


Рисунок А.12 – 12 слайд (рисунок виконано самостійно).

Аналіз отриманих результатів

Ітерація	Server 1	Server 2	Server 3
1	179	53	33
2	210	59	34
3	210	59	34
4	239	60	34
5	239	60	34
6	259	60	34
7	259	60	34

Таблиця ітерацій

Консольний вивід результатів відпрацювання LogMixAdaptiveBalancer

13

Рисунок А.13 – 13 слайд (рисунок виконано самостійно).

Аналіз результатів

Алгоритм гнучко коригує розподіл: частка запитів, що надсилається на Server 2 і 3, трохи зростає в останніх циклах, однак основне навантаження залишається за Server 1, допоки його показники залишаються найменш критичними. Саме така стратегія забезпечує найстійкішу та найефективнішу роботу всієї системи навіть у випадку довготривалого або несиметричного навантаження, оскільки LogMixAdaptiveBalancer постійно переоцінює метрики в реальному часі, мінімізує вплив вузлів з високою часткою збоїв і не допускає різких провалів у доступності жодного з активних серверів.

У підсумку, результати експерименту демонструють, що запропонований алгоритм LogMixAdaptiveBalancer здатний ефективно й гнучко розподіляти навантаження між серверами в умовах нерівномірної динаміки та поступового зростання критичних метрик, таких як CPU Usage, кількість активних запитів і частка збоїв. На відміну від класичних підходів

14

Рисунок А.14 – 14 слайд (рисунок виконано самостійно).

Публікація результатів

15

Рисунок А.15 – 15 слайд (рисунок виконано самостійно).

Підсумки

У ході виконання кваліфікаційної роботи було проведено аналіз проблемної області дослідження систем розподілу запитів та балансування навантаження, а також актуальних питань, пов'язаних із забезпеченням стійкої роботи серверних комплексів в умовах високої динаміки трафіку та різнорідності характеристик вузлів. Здійснено систематичний огляд існуючих підходів і класичних алгоритмів балансування, зокрема Round Robin, Least Connections, Weighted Round Robin та адаптивних ресурсно-орієнтованих алгоритмів.

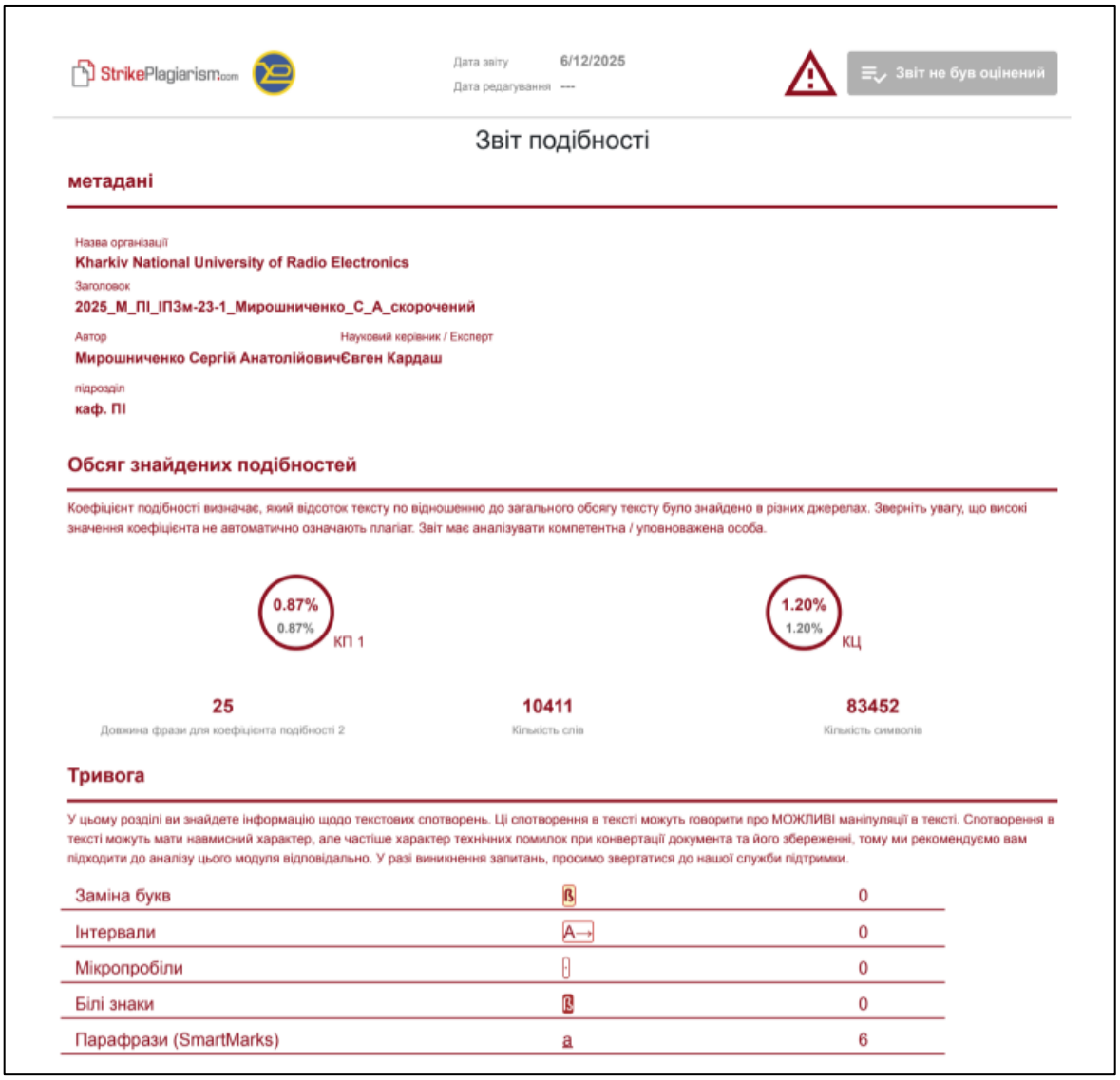
Отримані результати підтверджують доцільність використання комплексних багатофакторних підходів до розподілу запитів, а запропонована методика дозволяє наочно і системно порівнювати ефективність різних алгоритмів у контрольованих експериментальних умовах, що може стати основою для впровадження більш адаптивних і стійких до навантажень балансувальників у практичних системах.

16

Рисунок А.16 – 16 слайд (рисунок виконано самостійно).

ДОДАТОК Б

Результат проходження на академічний плагіат



StrikePlagiarism.com

Дата звіту 6/12/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kharkiv National University of Radio Electronics
Заголовок
2025_М_ПІ_ІПЗм-23-1_Мирошниченко_С_А_скорочений
Автор
Мирошниченко Сергій Анатолійович
Науковий керівник / Експерт
Євген Кардаш
підрозділ
каф. ПІ

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

0.87% 0.87% КП 1

1.20% 1.20% КЦ

25	10411	83452
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

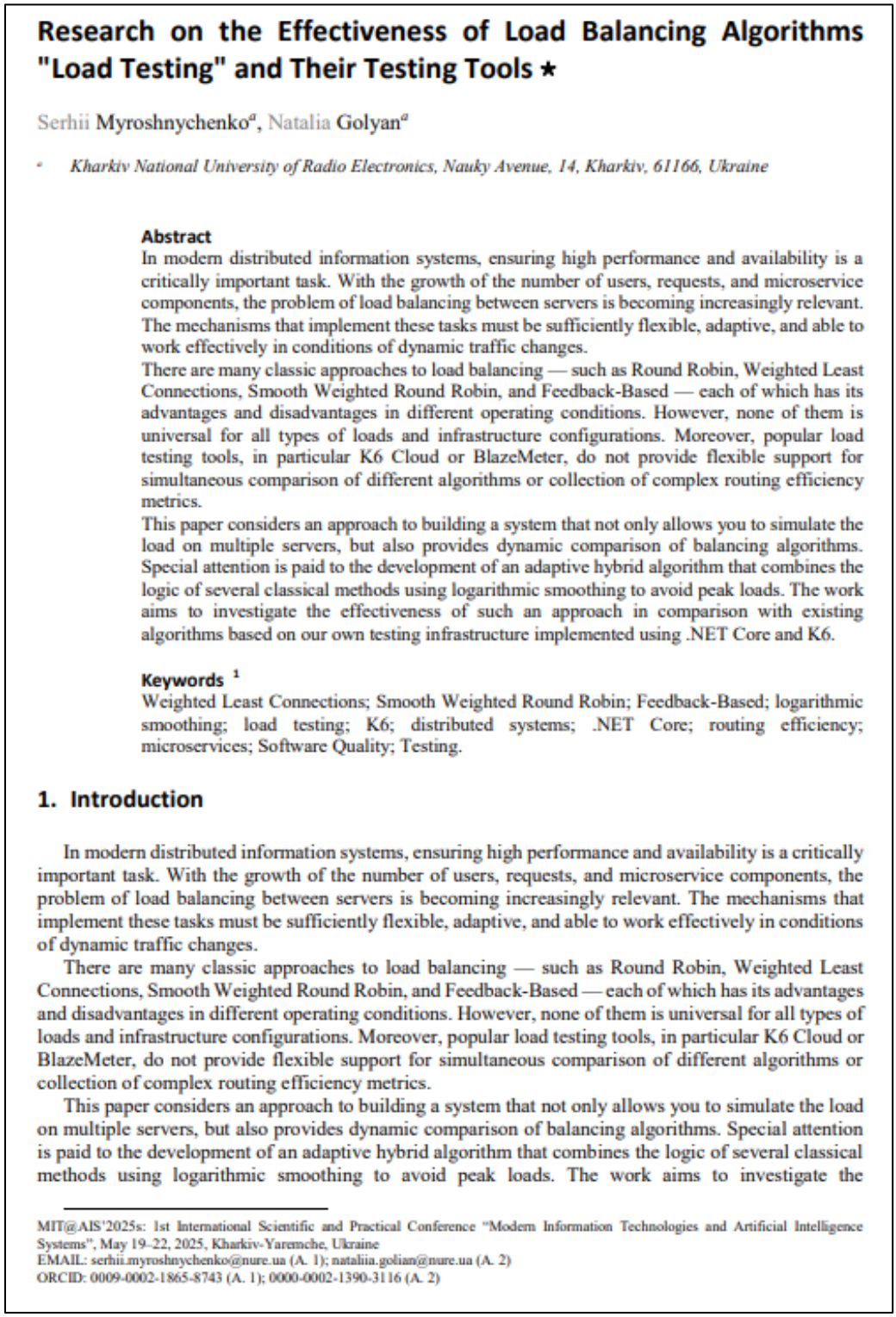
У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	6

Рисунок Б.1 – Результати перевірки (рисунок виконано самостійно).

ДОДАТОК В

Апробація результатів роботи на конференції «1 Міжнародна науково-практична конференція «Modern Information Technologies and Artificial Intelligent Systems MIT@AIS-2025»»



Research on the Effectiveness of Load Balancing Algorithms "Load Testing" and Their Testing Tools ★

Serhii Myroshnychenko[a], Natalia Golyan[a]

[a] *Kharkiv National University of Radio Electronics, Nauky Avenue, 14, Kharkiv, 61166, Ukraine*

Abstract

In modern distributed information systems, ensuring high performance and availability is a critically important task. With the growth of the number of users, requests, and microservice components, the problem of load balancing between servers is becoming increasingly relevant. The mechanisms that implement these tasks must be sufficiently flexible, adaptive, and able to work effectively in conditions of dynamic traffic changes.

There are many classic approaches to load balancing — such as Round Robin, Weighted Least Connections, Smooth Weighted Round Robin, and Feedback-Based — each of which has its advantages and disadvantages in different operating conditions. However, none of them is universal for all types of loads and infrastructure configurations. Moreover, popular load testing tools, in particular K6 Cloud or BlazeMeter, do not provide flexible support for simultaneous comparison of different algorithms or collection of complex routing efficiency metrics.

This paper considers an approach to building a system that not only allows you to simulate the load on multiple servers, but also provides dynamic comparison of balancing algorithms. Special attention is paid to the development of an adaptive hybrid algorithm that combines the logic of several classical methods using logarithmic smoothing to avoid peak loads. The work aims to investigate the effectiveness of such an approach in comparison with existing algorithms based on our own testing infrastructure implemented using .NET Core and K6.

Keywords [1]

Weighted Least Connections; Smooth Weighted Round Robin; Feedback-Based; logarithmic smoothing; load testing; K6; distributed systems; .NET Core; routing efficiency; microservices; Software Quality; Testing.

1. Introduction

In modern distributed information systems, ensuring high performance and availability is a critically important task. With the growth of the number of users, requests, and microservice components, the problem of load balancing between servers is becoming increasingly relevant. The mechanisms that implement these tasks must be sufficiently flexible, adaptive, and able to work effectively in conditions of dynamic traffic changes.

There are many classic approaches to load balancing — such as Round Robin, Weighted Least Connections, Smooth Weighted Round Robin, and Feedback-Based — each of which has its advantages and disadvantages in different operating conditions. However, none of them is universal for all types of loads and infrastructure configurations. Moreover, popular load testing tools, in particular K6 Cloud or BlazeMeter, do not provide flexible support for simultaneous comparison of different algorithms or collection of complex routing efficiency metrics.

This paper considers an approach to building a system that not only allows you to simulate the load on multiple servers, but also provides dynamic comparison of balancing algorithms. Special attention is paid to the development of an adaptive hybrid algorithm that combines the logic of several classical methods using logarithmic smoothing to avoid peak loads. The work aims to investigate the

MIT@AIS'2025s: 1st International Scientific and Practical Conference "Modern Information Technologies and Artificial Intelligence Systems", May 19–22, 2025, Kharkiv-Yaremche, Ukraine
EMAIL: serhii.myroshnychenko@nure.ua (A. 1); nataliia.golian@nure.ua (A. 2)
ORCID: 0009-0002-1865-8743 (A. 1); 0000-0002-1390-3116 (A. 2)

Рисунок В.1 – Перша сторінка статті (рисунок виконано самостійно)

effectiveness of such an approach in comparison with existing algorithms based on our own testing infrastructure implemented using .NET Core and K6[1].

2. Justification of the choice of means of testing the effectiveness of algorithms

To implement the load balancing algorithm comparison system, the .NET Core platform was chosen, which provides high performance, scalability, and flexibility in web application development. Thanks to its dependency and extension support, .NET Core allows easy integration of routing modules, metrics processing, and API controllers.

To simulate client requests and create real load, the K6 tool is used - a modern platform for scripted load testing with the ability to automate, collect metrics, and JavaScript scripts.

Monitoring of the internal state of servers and metrics is implemented through the Prometheus.NET library, which allows for real-time data collection and easy integration with Grafana for visualization.

The selected algorithms (Round Robin, Weighted Least Connections, Smooth Weighted Round Robin, Feedback-Based) are implemented as separate services, which allows for flexible modification and comparison of their behavior in a single environment.

3. Analysis of existing balancing algorithms

In modern distributed software systems, load balancing plays a key role in ensuring reliability, scalability, and high availability of services. The most popular ones are listed below:

Round Robin (cyclical balancing), this approach involves sequentially redirecting each new request to the next server in the list. Its advantage is simplicity of implementation, but it does not take into account the real load or performance of each server, which can lead to imbalance.

Weighted Round Robin, is an improved version of the cyclic algorithm, in which each server is assigned a weight that determines the frequency of its selection.

Thus, more powerful servers receive more requests. The problem is that the weight is static and does not take into account dynamic changes in the load[2].

Smooth Weighted Round Robin, this algorithm eliminates the jumpiness in the distribution of requests inherent in the classic Weighted Round Robin by smoothly distributing the weight over time. It provides a more uniform distribution even in complex configurations.

Least Connections, this algorithm selects the server with the fewest active connections. It is effective in scenarios where the request load varies greatly. However, it does not take into account the power or performance of the servers[3].

Weighted Least Connections, this is a combination of the Least Connections and Weighted Round Robin approaches.

Servers with fewer active connections and higher weights are given priority. This scheme provides a more dynamic distribution, but requires regular updates of the weights[4].

Feedback-Based algorithms, these methods take into account feedback from the servers (CPU, latency, number of requests, etc.) to make decisions. These are some of the most flexible and intelligent algorithms, but they are complex to implement and require real-time metrics collection.

Analysis shows that none of the above approaches is universal. Most systems use one of these algorithms without the ability to combine their advantages or dynamically adjust depending on the actual load. This creates a need for new adaptive approaches that can combine the strengths of multiple algorithms.

4. Analysis of existing approaches to testing balancing algorithms

Evaluating the effectiveness of load balancing algorithms is critical for developing stable and scalable distributed systems. However, in practice, existing testing tools have a number of limitations that significantly complicate a full comparison between algorithms. The closest candidates for such a system are considered below:

Рисунок В.2 – Друга сторінка статті (рисунок виконано самостійно)

- K6 Cloud. K6 is a modern load testing tool that allows you to simulate a large number of users and supports writing scripts in JavaScript. However, the cloud version of K6 Cloud is not focused on testing load balancing systems. It does not allow you to record metrics that reflect the effectiveness of specific algorithms (for example, server selection time, distribution of requests across instances, performance, etc.). Thus, it is more suitable for general API testing, rather than for analytics of balancing systems;
- BlazeMeter. BlazeMeter is another popular service for load generation and automated testing. However, when trying to compare several balancing algorithms, it becomes necessary to manually change the system configuration each time or create separate environments for each algorithm. This makes it impossible to test multiple strategies in parallel under the same conditions.

Thus, existing platforms are more focused on general performance testing than on specialized analysis of the effectiveness of balancing algorithms.

They do not provide transparent metrics on the selected request routing path, the response time of each server, changes in load, or adaptation to traffic changes.

The lack of a full-fledged toolkit for in-depth analysis and comparison of algorithms creates the need for a proprietary testing environment that provides control over the balancing logic, transparency in the collection of metrics, and the ability to compare results in real time[5].

5. Software solutions and adaptive algorithm

As part of the development of a system for evaluating the effectiveness of load balancing algorithms, a flexible software platform was implemented. It includes an adaptive request routing algorithm and tools for collecting, analyzing, and displaying performance metrics.

The system is modularly built on the .NET platform and uses middleware to dynamically redirect incoming requests. It supports multiple load balancing algorithms (Round Robin, Least Connections, etc.), with the ability to switch between them at runtime.

The key contribution of this project is the development of an adaptive algorithm that combines principles from classical strategies such as Smooth Weighted Round Robin and Feedback-Based, while introducing logarithmic smoothing to reduce the impact of load spikes. This enables more balanced and accurate request distribution[6],[7],[8].

The efficiency score of each server is calculated using the formula:

$$Score(S) = \frac{Log(ActiveReq + 1) + Log(CpuUsa + 1) + Log(TotalReq + 1) + Log(RespType + 1)}{\text{Weight} + \text{e}} \quad (1)$$

where ε - ε is a small positive constant to prevent division by zero. This formula takes into account not only real-time server activity and CPU load but also historical performance metrics such as the total number of handled requests and response time.

A key innovation is the dynamic computation of each server's weight:

$$Weight = \frac{Ci * Mi}{\text{Weight} + \text{eLog}(\text{AVGResponseTimei} + 1) + \text{Log}(\text{FailureRatei} + 1)} \quad (2)$$

where Ci represents the number of logical CPU cores, Mi is the amount of available RAM, and AvgResponseTimei andFailureRatei are the average response time and the error rate, respectively. This ensures that servers with higher capacity, stability, and lower failure rates are prioritized when distributing the load.

This approach enables the creation of a scalable and adaptive load balancing system that responds to real-time changes in demand, while also considering the technical characteristics, reliability, and performance of each server.

6. Conclusion

In conclusion, each of the algorithms has its own advantages, but also limitations that manifest themselves in cases of sudden load changes, heterogeneous server computing power, or high requirements for real-time adaptability.

Рисунок В.3 – Третя сторінка статті (рисунок виконано самостійно)

In addition, an analysis of existing testing tools, such as K6 Cloud and BlazeMeter, showed that none of them provides full support for comparing the effectiveness of load balancing algorithms. They do not allow for automated recording of key metrics (such as request distribution, response time of individual servers, load change in dynamics) and do not support parallel testing of several strategies in the same environment.

According to the results, we can say about the need to create our own adaptive balancing system that would combine the strengths of classical algorithms with logarithmic peak smoothing mechanisms.

This approach will allow us to take into account the current state of servers, the number of requests, CPU load, and other dynamic parameters, providing more accurate and effective balancing in conditions of variable traffic.

In addition, an integral part of the research was the development of our own algorithm testing and comparison system, which would allow us to record the balancer's behavior in real time, visualize the collected metrics, and provide for changing the algorithm without restarting the system. Such a tool could be effective in large-scale distributed systems.

7. References

[1] J. Smith et al., "Adaptive Load Balancing System Using .NET Core and K6: A Case Study," in Proc. of the 2022 Int. Conf. on Distributed Systems, ACM, 2022, pp. 143–150. doi:10.1145/3543513.3543591.

[2] A. Tanenbaum and M. van Steen, Distributed Systems: Principles and Paradigms, 2nd ed., Pearson, 2007.

[3] M. Harchol-Balter, Performance Modeling and Design of Computer Systems: Queueing Theory in Action, Cambridge University Press, 2013.

[4] J. Liu, et al., "An Optimized Weighted Least Connections Algorithm for Load Balancing in Cloud Environments," in IEEE Access, vol. 8, pp. 11245–11255, 2020. doi:10.1109/ACCESS.2020.2965933.

[5] C. Risi, "The Pros and Cons of Different Tools – BlazeMeter," Craig Risi Blog, Jul. 2024. [Online]. Available: https://www.craigrisi.com/post/the-pros-and-cons-of-different-tools-blazemeter

[6] K. Chawla, "Reinforcement Learning-Based Adaptive Load Balancing for Dynamic Cloud Environments," arXiv, Sep. 2024. [Online]. Available: https://arxiv.org/abs/2409.04896

[7] A. Adewojo and J. M. Bass, "A Novel Weight-Assignment Load Balancing Algorithm for Cloud Applications," in Proc. of the 2022 Int. Conf. on Cloud Computing Technologies, SCITEPRESS, 2022, pp. 341–348. [Online]. Available: https://www.scitepress.org/Papers/2022/110916/110916.pdf

[8] GeeksforGeeks, "Adaptive Load Balancing – System Design," GeeksforGeeks, Jan. 2024. [Online]. Available: https://www.geeksforgeeks.org/adaptive-load-balancing-system-design/

Рисунок В.4 – Четверта сторінка статті (рисунок виконано самостійно)

ДОДАТОК Г

Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимогам ДСТУ 3008:2015

Експертний висновок результатів перевірки кваліфікаційної роботи

студент (посада) | програмної інженерії (кафедра) | ІПЗм-23-1 (група)

Сергій МИРОШНИЧЕНКО
(прізвище, ім'я, по батькові)

Зауваження

Пункт ДСТУ 3008-2015	Зміст пункту	Сторінка кваліфікаційної роботи
1	2	3
	7.1 Загальні положення	
	7.3 Нумерація сторінок звіту	
	7.5 Рисунки	
	7.6 Таблиці	
	7.7 Переліки	
	7.8 Примітки	
	7.9 Виноски	
	7.10 Формули та рівняння	
	7.11 Посилання	
	7.13 Список авторів	
	7.14 Скорочення та умовні познаки	
	7.15 Додатки	

Експерт _____ (підпис) Вадим НЕЧВОЛОД (прізвище, ініціали)

Зауважень немає.
14.06.2025

Рисунок Г.1 – Експертний висновок (рисунок виконано самостійно)