Специфікація програмного забезпечення

СПЕЦИФІКАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«Дослідження алгоритмів балансування навантаження програмних систем на .Net Core»

Автор:

ст. гр. ІПЗм-23-1

Мирошниченко Сергій Анатолійович

Харків, 2025

**ВСТУП**

* 1. Огляд програмного забезпечення

Більшість сучасних вебсервісів функціонують у середовищах з нерівномірним розподілом навантаження, де критично важливим є швидке та ефективне перенаправлення запитів до вільних ресурсів. Для цього використовуються спеціалізовані системи балансування, які впливають на продуктивність, відмовостійкість і масштабованість застосунків.

Існує чимало реалізацій, що надають базову маршрутизацію — однак вони часто не враховують зміну стану інфраструктури в реальному часі. Унаслідок цього виникають затримки в обробці запитів або перевантаження окремих компонентів системи. Тому виникає потреба у гнучкому рішенні, здатному:

* здійснювати вибір сервера не лише за фіксованими правилами, а й з урахуванням поточної ситуації в системі;
* оновлювати інформацію про ресурси з мінімальними затримками;
* бути сумісним з архітектурами на основі .NET та легко вбудовуватись у вже існуючі сервіси.

Система, що розробляється в межах цього дослідження, покликана усунути вказані проблеми шляхом впровадження адаптивного підходу до балансування. Вона дозволяє не лише підвищити ефективність розподілу запитів, а й забезпечити стабільну роботу сервісів при змінних навантаженнях без шкоди для продуктивності.

* 1. Мета

Метою цієї кваліфікаційної роботи є дослідження ефективності існуючих алгоритмів балансування навантаження та розробка універсального підходу, здатного адаптуватися до змін у системі в режимі реального часу. Основна задача — створити механізм розподілу запитів, який забезпечує стабільну продуктивність та рівномірне навантаження на сервери за будь-яких умов.

Для реалізації цієї мети передбачено виконання таких завдань:

* проаналізувати особливості роботи класичних і адаптивних алгоритмів (Round Robin, Weighted Round Robin, Least Connections, Adaptive);
* сформулювати вимоги до гнучкої системи балансування з урахуванням статичних і динамічних показників (використання CPU, кількість активних з’єднань, частота помилок);
* розробити модуль збору метрик у реальному часі з подальшим використанням цих даних для прийняття рішень щодо маршрутизації;
* реалізувати алгоритм, який комбінує різні підходи балансування та автоматично адаптується до змін у навантаженні;
* протестувати систему в умовах пікових навантажень із використанням інструментів генерації запитів та оцінити її ефективність за такими показниками: середній час відповіді, рівень завантаження серверів, частота відмов, стабільність обробки запитів.

Успішне виконання поставлених завдань дозволить створити практично застосовне рішення для адаптивного розподілу навантаження, що легко інтегрується в .NET-інфраструктуру та підвищує загальну надійність і масштабованість системи.

* 1. Межі

У межах даного дослідження розглядаються такі аспекти:

* реалізація системи балансування навантаження на платформі .NET із використанням кастомного middleware;
* дослідження ефективності базових алгоритмів розподілу HTTP-запитів (Round Robin, Weighted Round Robin, Least Connections, Adaptive);
* симуляція серверного навантаження з використанням емуляторів на базі ASP.NET Core;
* збір і обробка метрик стану серверів (CPU, активні запити, коефіцієнт помилок тощо) через внутрішні REST-ендпоїнти;
* використання інструменту K6 для генерації навантаження та аналізу поведінки системи в різних сценаріях.

При цьому дослідження не охоплює:

* розгортання балансувальника у продакшн-середовищах із реальним мережевим проксіюванням запитів;
* використання апаратних або сторонніх програмних рішень балансування (наприклад, NGINX, HAProxy, Kubernetes);
* питання безпеки, авторизації та автентифікації HTTP-запитів.

Окреслені межі дозволяють зосередитися виключно на архітектурі та поведінці алгоритмів балансування в умовах контрольованого середовища, а також на розробці та валідації універсального адаптивного підходу в рамках .NET-екосистеми.

# **2 ЗАГАЛЬНИЙ ОПИС**

2.1. Перспективи програмного забезпечення

Із розвитком високонавантажених веб-систем зростає потреба в інструментах, здатних забезпечити стабільний розподіл трафіку та ефективне використання серверних ресурсів. Розробка компактної та адаптивної системи балансування HTTP-запитів для .NET-платформи відкриває нові можливості для гнучкого управління навантаженням навіть у проєктах із обмеженою інфраструктурою.

Використання адаптивних алгоритмів, які враховують реальні метрики (CPU, активні з'єднання, частота помилок), дозволяє підвищити ефективність обробки запитів у динамічних умовах. Таке програмне забезпечення сприяє зниженню затримок, кращій стійкості до пікових навантажень та оптимізації використання серверних потужностей без необхідності залучення складних сторонніх рішень.

Завдяки модульній структурі, система легко адаптується до нових вимог — наприклад, підтримка нових алгоритмів або розширення моніторингових можливостей не потребують суттєвих змін архітектури. Інтеграція з такими інструментами, як Prometheus, забезпечує прозорий збір метрик, що відкриває шлях до побудови систем автоматичного реагування або аналітики.

2.2. Функції програмного забезпечення

На поточному етапі проєкту передбачається реалізація таких основних можливостей системи:

* сервіс балансування оброблятиме вхідні HTTP-запити без блокувань, забезпечуючи масштабованість і підтримку великої кількості одночасних клієнтських звернень;
* модуль збору метрик здійснюватиме регулярний моніторинг стану серверів (використання CPU, кількість активних запитів, середній час відповіді, частота помилок), зберігаючи історичні дані для подальшого аналізу;
* кожен алгоритм балансування буде підтримувати динамічне врахування метрик у режимі реального часу, дозволяючи адаптивно обирати сервер з найкращими поточними характеристиками;
* реалізується механізм розрахунку рейтингу вузлів (на основі вагових коефіцієнтів, навантаження та порогових значень), що забезпечить гнучкий контроль за розподілом трафіку;
* система дозволятиме налаштовувати параметри алгоритмів (ваги, пороги, максимальні значення активних з’єднань), адаптуючи їх під конкретні сценарії навантаження;
* фоновий процес відповідатиме за оновлення даних про вузли кожні 5 секунд, що забезпечить актуальність рішень у виборі сервера навіть при раптових змінах у системі.

Передбачений функціонал забезпечить основу для створення адаптивного та надійного механізму балансування, який не потребує постійного втручання користувача й здатен ефективно функціонувати в умовах змінного навантаження.

2.3. Характеристики користувачів

Характеристики зацікавлених користувачів наведені у таблиці 2.1.

Таблиця 2.1 – Характеристика зацікавлених користувачів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Зацікавлена сторона** | **Основна цінність** | **Обов’язки** | **Головна ціль** | **Обмеження** |
| **Розробник .NET-систем** | **Можливість швидко інтегрувати адаптивне балансування без складних налаштувань** | **Інтеграція системи в існуючий застосунок, налаштування параметрів балансувальника** | **Підвищення стабільності та продуктивності застосунку** | **Обмежений час на впровадження та підтримку, нестача досвіду з системами балансування** |

Кінець таблиці 2.1

Продовження таблиці 2.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Системний адміністратор** | Отримання наочних метрик навантаження для моніторингу стану серверів | Контроль за працездатністю інфраструктури, аналіз метрик, втручання у разі збоїв | Стабільна робота серверів без перевантаження | Неможливість постійного ручного моніторингу, залежність від точності метрик |
| **Бізнес-аналітик** | Збір статистики для оцінки навантаження й ефективності розподілу запитів | Аналіз даних, формування звітів і рекомендацій щодо покращення системної продуктивності | Прийняття обґрунтованих рішень щодо масштабування або оптимізації | Обмежений доступ до технічних деталей, потреба у візуалізації та зручному інтерфейсі |
| **Клієнт кінцевого продукту** | Швидка і стабільна відповідь системи незалежно від навантаження | Надсилання запитів до системи | Мінімальні затримки та відсутність збоїв у роботі сервісу | Відсутність контролю над внутрішніми процесами, залежність від обраної архітектури системи |
| **DevOps-інженер** | Гнучкість у налаштуванні, можливість автоматизації розгортання та оновлень | Встановлення, CI/CD-інтеграція, оновлення системи балансування | Автоматизація процесів, забезпечення безперервної роботи застосунку | Необхідність сумісності з існуючими інструментами та пайплайнами |

2.4. Загальні обмеження

У межах даного проєкту визначено низку загальних обмежень, що дозволяють зосередити увагу на ключових аспектах реалізації та тестування системи балансування запитів.

* рішення орієнтоване виключно на платформу .NET 8 або новішу. Підтримка попередніх версій фреймворку (наприклад, .NET 6 або .NET Framework) не передбачається;
* система розроблена для роботи в односерверному середовищі з симульованими вузлами. Реальна кластеризація, розподіл навантаження між фізично розміщеними серверами або інтеграція з хмарними балансувальниками (AWS ELB, Azure LB тощо) не реалізуються;
* проєкт не включає повноцінну функціональність HTTP-проксіювання: балансувальник не пересилає запити до реальних бекендів, а лише визначає маршрут, фокусуючись на логіці алгоритму;
* підтримка тільки HTTP-протоколу. Інші протоколи (gRPC, WebSocket тощо) в рамках дослідження не використовуються;
* для тестування навантаження передбачено інтеграцію лише з інструментом K6. Інші генератори трафіку (Locust, JMeter) не включені до складу системи;
* аспекти безпеки (TLS, автентифікація, контроль доступу) не входять до сфери реалізації. Очікується, що такі функції забезпечуються зовнішньою інфраструктурою.

Ці обмеження дозволяють зосередитися на розробці та дослідженні адаптивної логіки балансування, не ускладнюючи реалізацію додатковими компонентами, що не є критичними для основної мети проєкту.

2.5. Припущення і залежності

У процесі розробки, тестування та розгортання системи балансування HTTP-запитів було зроблено такі основні припущення:

* мережева інфраструктура між клієнтами, балансувальником і симульованими серверами стабільна, з мінімальними затримками, що не спотворюють результати вимірювань;
* серверні вузли (імітаційні) працюють без збоїв протягом усього періоду тестування, і їхні відповіді на запити /api/status завжди повертаються вчасно;
* апаратне середовище має достатню кількість ресурсів (CPU, RAM) для запуску кількох ASP.NET Core застосунків одночасно з генерацією навантаження;
* користувачі системи мають базовий досвід роботи з .NET-проєктами та не потребують складних інструментів налаштування або деплойменту;
* всі компоненти системи працюють у межах локального середовища або однієї мережі без потреби в міжмережевих проксі чи шифруванні з’єднань.

Залежності проєкту включають:

* .NET SDK версії 8.0 або новішої, необхідний для збирання та запуску основного балансувальника та серверних емуляторів;
* інструмент генерації навантаження K6 для відтворення реалістичних сценаріїв звернень клієнтів;
* сховище конфігурацій (appsettings.json або зовнішній JSON/YAML-файл) для зберігання параметрів роботи алгоритмів — зокрема, вагових коефіцієнтів та порогів;
* бібліотека Prometheus.Net (або аналогічна) для експорту метрик у Prometheus-сумісному форматі;
* браузер або Postman (опціонально) — для ручної перевірки відповіді балансувальника на HTTP-запити під час налагодження.

Ці припущення та залежності формують основу для стабільного функціонування системи в контрольованому середовищі та забезпечують правильну роботу алгоритмів розподілу навантаження.

# **3 КОНКРЕТНІ ВИМОГИ**

3.1. Вимоги до зовнішніх інтерфейсів

3.1.1. Апаратний інтерфейс

Для стабільної роботи системи балансування запитів та симульованих серверних компонентів необхідна серверна платформа з такими мінімальними характеристиками:

* процесор: x64-сумісний CPU з принаймні 4 логічними ядрами (2.5 ГГц або вище) для одночасного обслуговування декількох .NET-процесів;
* оперативна пам’ять: щонайменше 6 ГБ для запуску балансувальника, симульованих серверів та генератора навантаження; рекомендовано 8–16 ГБ для тестування під піковим навантаженням;
* мережевий інтерфейс: пропускна здатність не менше 100 Мбіт/с та локальна затримка між балансувальником і серверними вузлами не більше 5 мс;
* відкриті TCP-порти: для взаємодії балансувальника з вузлами серверів, зокрема порти 5001–5003 для HTTP-запитів до /api/status;
* накопичувач: SSD-диск для швидкої роботи логування та зберігання журналів тестувань (опціонально, якщо записуються великі обсяги логів).

Дотримання цих вимог забезпечує коректне виконання логіки вибору серверів у режимі реального часу, точний збір метрик та відсутність затримок, що могли б вплинути на достовірність результатів дослідження.

3.1.2. Програмний інтерфейс

Програмний інтерфейс реалізовано у вигляді .NET-базованого застосунку, який включає модуль балансувальника запитів та серверні компоненти, що імітують навантаження. Рішення орієнтоване на інтеграцію в системи, що потребують гнучкого розподілу HTTP-трафіку між кількома обробниками з урахуванням динамічного стану кожного з них.

Система розроблена з використанням технології .NET 8, а як середовище розробки використовується Visual Studio 2022. Основна частина логіки побудована у вигляді модульної структури сервісів та middleware-компонентів, що забезпечує просту інтеграцію з ASP.NET Core-проєктами.

Комунікація між модулями системи здійснюється по HTTP-протоколу через TCP/IP. Імітовані сервери працюють локально (наприклад, на портах 5001–5003), але можуть бути розгорнуті й віддалено при збереженні відповідної адресації та мережевого доступу.

Такий підхід забезпечує максимальну гнучкість у розробці, тестуванні та адаптації системи до різних умов навантаження та архітектурних обмежень.

3.1.3. Комунікаційний протокол

Комунікація між основними компонентами системи балансування (балансувальником, серверними вузлами та модулем збору метрик) здійснюється за допомогою HTTP-протоколу поверх TCP/IP. Балансувальник надсилає запити на визначений шлях /api/status кожному вузлу для отримання актуальних метрик (завантаження CPU, кількість активних запитів, статус вузла).

Зовнішні мережеві протоколи типу WebSocket, SignalR або gRPC у межах проєкту не використовуються. Комунікація побудована на простому, прозорому та легко масштабованому HTTP-взаємодії, що забезпечує зручність тестування та налагодження.

3.1.4. Обмеження пам’яті

Рекомендований обсяг пам’яті — від 6 ГБ і вище, особливо у випадках одночасного запуску кількох серверів, сервісу балансування та інструментів тестування (наприклад, K6), щоб уникнути нестачі ресурсів та зниження продуктивності під час пікового навантаження.