КОМП'ЮТЕРНА ІНЖЕНЕРІЯ, ІНФОРМАЦІЙНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ

https://doi.org/10.31649/1997-9266-2026-186-3-6-16

УДК 004.75:004.415.2:004.896

В. О. Гнатюк[1,2]
К. Ю. Зандер[1]

МЕТОД ДИНАМІЧНОГО ОПТИМАЛЬНОГО РОЗПОДІЛУ ПОТОКУ ЗВЕРНЕНЬ У БАГАТОСЕРВЕРНИХ ІНТЕРАКТИВНИХ СИСТЕМАХ

[1]Державний університет «Київський авіаційний інститут»;
[2]ДержНДІ технологій кібербезпеки, Київ

Запропоновано динамічний оптимальний метод розподілу потоків запитів у багатосерверних інтерактивних системах, спрямований на забезпечення адаптивного балансування навантаження в умовах нестаціонарного та непередбачуваного трафіку. Запропонований підхід забезпечує коригування розподілу потоку запитів між паралельними серверами в реальному часі на основі поточних коефіцієнтів навантаження, що дозволяє підтримувати стабільність системи та мінімізувати ймовірність перевантажень. Розроблено структурно-функціональну модель запропонованого методу, яка складається з модулів згладжування трафіку, динамічного демультиплексування та вирівнювання навантаження по серверній лінії. Запропоновано модифіковану версію алгоритму "token bucket", призначену для перетворення нестаціонарного та пульсуючого вхідного потоку у квазістаціонарні сегменти трафіку, що можуть оброблятися дискретними механізмами керування. Модель здійснює безперервне вимірювання миттєвих параметрів навантаження кожного сервера та виконує ітеративний перерозподіл черг запитів між перевантаженими та недовантаженими вузлами. У разі критичного перевантаження можливе автоматичне підключення додаткових серверів, тоді як у періоди низької інтенсивності надлишкові сервери тимчасово деактивуються з метою оптимізації використання ресурсів. Запропонований метод може бути використаний під час проєктування систем реального часу, хмарних центрів оброблення даних, VoIP- та IoT-платформ, а також ядер мереж 5G, для яких підтримання стабільних показників продуктивності за змінних навантажень є критично важливою вимогою. Підхід підвищує масштабованість системи, зменшує затримку та ймовірність втрати запитів, а також створює підґрунтя для впровадження інтелектуальних програмних контролерів адаптивного балансування навантаження. Програмна реалізація підтвердила ефективність розробленого методу динамічного оптимального розподілу потоків запитів у багатосерверних інтерактивних системах. Алгоритм забезпечує адаптивне балансування навантаження, знижує ймовірність перевантажень і втрати запитів та може бути інтегрований у системи реального часу, SDN-платформи, VoIP-сервери або центри оброблення даних.

Ключові слова: балансування навантаження, інтерактивна система, багатосерверна архітектура, метод оптимального розподілу, динамічний перерозподіл, алгоритм «відра токенів».

Вступ

Сучасні інтерактивні системи функціонують у середовищах з високою інтенсивністю інформаційних потоків, де обслуговування великої кількості одночасних клієнтських звернень вимагає забезпечення рівномірного навантаження між серверами. У реальних умовах трафік звернень має виражену нестаціонарність, пульсаційність та випадковий характер, що призводить до виникнення короткочасних перенавантажень окремих серверів і, як наслідок, — до збільшення затримок обробки, втрат запитів та зниження загальної продуктивності системи.

Наявні методи балансування навантаження, зокрема алгоритми типу *Round Robin*, *Least Connections* або *Weighted Load Balancing*, не враховують динаміку змін інтенсивності вхідного потоку в реальному часі. Внаслідок цього вони не здатні забезпечити стабільність роботи системи у разі

раптових сплесків трафіку, характерних для інтерактивних і розподілених середовищ. Особливо це актуально для багатосерверних систем, що функціонують у рамках хмарних платформ, центрів обробки даних, VoIP- та IoT-сервісів, де необхідно підтримувати сталість рівня якості обслуговування (QoS) за мінливих навантажень.

Вирішення цієї проблеми вимагає розроблення методу динамічного оптимального розподілу потоку звернень між серверами інтерактивної системи, який у режимі реального часу здатний: здійснювати адаптивне балансування навантаження між серверами, що паралельно працюють; враховувати поточні значення коефіцієнтів завантаження кожного сервера; забезпечувати вирівнювання потоків за непередбачуваних змін інтенсивності трафіку; оптимізувати використання апаратних ресурсів шляхом автоматичного підключення або відключення серверів залежно від рівня навантаження.

Отже, постає задача синтезу методу та відповідної структурно-функціональної моделі, які дозволять реалізувати механізм динамічного балансування навантаження у нестаціонарному середовищі функціонування інтерактивної системи та забезпечити мінімізацію втрат звернень і часу їх оброблення за обмежених обчислювальних ресурсів.

У сучасній науковій літературі питанням балансування навантаження (load balancing) у багатосерверних або розподілених системах приділено значну увагу. Наприклад, у праці [2] досліджується система з виділеними та спільними серверами, для якої формалізовано й розв'язано задачу оптимального маршрутування потоків з метою мінімізації зваженої середньої кількості задач у системі. Ця робота показує, що навіть у разі класичного підходу (маршрутизація за ймовірностями, фіксований пул) можна знайти аналітичні умови оптимальності, проте вона не враховує високодинамічну зміну навантаження й не орієнтована на інтерактивні системи з реальним часом.

Інше дослідження [3] проводить огляд статичних та динамічних стратегій балансування навантаження в розподілених системах, щодо класичних алгоритмів "Round Robin", "Least Connections" тощо. Автори зазначають, що динамічні методи мають переваги, але часто не враховують специфіку пульсуючого/нестаціонарного трафіку, а також не орієнтовані на архітектури з автоматичним додаванням/виключенням серверів.

Ще один релевантний огляд [4] порівнює метаевристичні алгоритми у хмарних середовищах, звертаючи увагу на складність та NP-складність задачі балансування в умовах великої кількості варіантів. Хоча це дослідження орієнтоване на хмарну інфраструктуру, а саме не на інтерактивні системи з жорсткими вимогами до затримки, воно вказує на актуальність розробки нових методів з урахуванням динаміки навантаження.

У дослідженні [5] запропоновано динамічний алгоритм балансування, який базується на багатокритеріальному прийнятті рішень у середовищі SDN: класифікація запитів, моніторинг серверів, маршрутизація на підставі реального завантаження. Це добре ілюструє сучасний напрямок — переходу до адаптивних, контекстно-залежних алгоритмів в умовах інтерактивних мереж.

Проте значущість заданої теми — динамічний оптимальний розподіл потоку звернень у багатосерверних інтерактивних системах — полягає в тому, що: більшість наявних досліджень або орієнтовані на хмарну/клауд-інфраструктуру, або на класи задач і серверів (queueing theory) без специфіки інтерактивних систем з реальним часом, не завжди враховують пульсаційність, нестаціонарність та адаптацію в режимі реального часу, недостатньо уваги приділено автоматичному підключенню/відключенню серверів під впливом змін інтенсивності потоку звернень, що є характерним для інтерактивних систем, VoIP, IoT, 5G ядра тощо.

Отже, є наукова ніша для підходу, який враховує ці аспекти і пропонує метод з адаптивним керуванням потоком звернень у реальному часі.

Метою статті є розробка методу динамічного оптимального розподілу потоку звернень між серверами багатосерверної інтерактивної системи, що здатен забезпечити адаптивне балансування навантаження з урахуванням нестаціонарності вхідних запитів, пульсаційності трафіку, можливості автоматичного підключення/відключення серверів і необхідності мінімізації часу обробки звернень, втрат звернень і нерівності завантаження серверів.

Результати дослідження

Розглянемо один з можливих варіантів технічної реалізації механізму, в основі якого лежить запропонований метод розв'язання задачі оптимального розподілу потоку звернень між серверами інтерактивної системи. Згідно з цим методом з метою зниження ризиків їхнього перевантаження

здійснюється балансування поточним навантаженням між серверами, що паралельно функціонують у реальному часі. Динаміка процесу балансування спрямована на однаковість ступеня завантаженості кожного з серверів. Робота механізму перерозподілу розглянута за умов, коли виникають значні непередбачувані сплески трафіка звернень, а прогнозування тривалості їхньої обробки засобами серверного обладнання є неможливим.

Математична модель та критерій оптимальності розподілу звернень

Для обґрунтування оптимальності запропонованого методу формалізуємо задачу динамічного балансування навантаження. Нехай інтерактивна система містить множину паралельно функціонуючих серверів $S = \{s_1, s_2, \dots, s_n\}$, де n — поточна кількість активних серверів у лінійці *Server Line*. У кожен момент часу t на вхід системи надходить нестаціонарний потік клієнтських звернень з інтенсивністю $\lambda(t)$.

Поточний стан i-го сервера ($i \in \{1,\dots,n\}$) характеризується довжиною черги необроблених запитів $q_i(t)$ та максимальною продуктивністю (пропускною здатністю) C_i. Коефіцієнт завантаження i-го сервера визначається як $\rho_i(t) = \frac{q_i(t)}{C_i}$. Середнє завантаження системи в поточний момент часу t становить $\bar{\rho}(t) = \frac{1}{n}\sum_{i=1}^{n} \rho_i(t)$.

Оптимальність у цьому методі розуміється як мінімізація дисперсії завантаженості серверів (забезпечення рівномірності) та одночасна мінімізація загального часу перебування запитів у системі. Відповідно, цільова функція (критерій оптимальності) має вигляд:

$$J(t) = \alpha \sum_{i=1}^{n} \left(\rho_i(t) - \bar{\rho}(t)\right)^2 + \beta \sum_{i=1}^{n} q_i(t) \to \min,$$

де α та β — вагові коефіцієнти, що задають пріоритет між рівномірністю розподілу навантаження та швидкістю обробки черг.

Процес балансування зводиться до знаходження оптимальної кількості запитів Δq_{ij}^*, яку необхідно перенаправити з перевантаженого сервера i на недовантажений сервер j на кожному кроці вирівнювання Δt_B. Оптимальний перерозподіл досягається шляхом розв'язання задачі $\Delta q_{ij}^* = \arg\min J\left(t + \Delta t_B\right)$ за умов дотримання системних обмежень щодо недопущення критичного перевантаження жодного з вузлів $\rho_i(t) \le \rho_{max}$, де ρ_{max} — гранично допустимий коефіцієнт завантаження.

Отже, структурно-функціональні схеми реалізують саме оптимальну обробку, оскільки на кожному кроці керування система наближає значення цільової функції $J(t)$ до глобального мінімуму.

Алгоритмічна реалізація методу динамічного оптимального розподілу

Для досягнення заявленого критерію оптимальності запропонований метод реалізується у вигляді ітеративної послідовності кроків, що виконуються в режимі реального часу:

Згладжування та сегментація нестаціонарного трафіку (*крок* Δt_3). Вхідний пульсуючий потік $\lambda(t)$ пропускається через модифікований алгоритм «відра токенів». На відміну від класичного алгоритму, розрахованого на стаціонарний трафік, у запропонованій модифікації швидкість генерації токенів $R(t)$ є динамічною величиною: $T(t) = T\left(t - \Delta t_3\right) + R(t) \cdot \Delta t_3$, де $R(t)$ адаптивно коригується залежно від поточного середнього завантаження системи $\bar{\rho}(t)$. Це дозволяє формувати квазістаціонарні відрізки трафіку.

1. *Моніторинг та оцінка стану системи* (*крок* Δt_B). На кожному кроці вирівнювання вимірювачі фіксують поточні значення коефіцієнтів завантаження $\rho_i(t)$ для кожного i-го сервера лінійки.

2. *Обчислення матриці перерозподілу.* Програмний контролер розв'язує рівняння настроювання та формує матрицю регулюючих зв'язків. Матриця визначає оптимальні частки ресурсу (кількість звернень Δq_{ij}^*), які підлягають переміщенню між буферами серверів.

3. *Вирівнювання навантаження та еластичне масштабування.* Демультиплексор здійснює фі-

зичний перерозподіл виділених часток запитів згідно з розрахованою матрицею. У разі, якщо після перерозподілу середнє завантаження системи $\bar{\rho}(t)$ перевищує критичний поріг $\rho_{critical}$, контролер ініціює підключення резервного сервера ($n \to n+1$). Зі зниженням інтенсивності потоку зайві сервери виводяться з лінійки для економії ресурсів.

Узагальнена структурно-функціональна схема засобів реалізації процедури пошуку IP-адрес серверів з персональними даними клієнтів показана на рис. 1. Ця схема дозволяє оптимальним чином організувати процес паралельного визначення IP-адрес пошукових серверів, які обробляють запити, що містяться у клієнтських зверненнях. Оптимальність у цьому випадку розуміється як створення ситуації, коли навантаження на лінійку серверів *Server line* у кожний поточний момент часу рівномірно розподіляється між пошуковими серверами СПА.

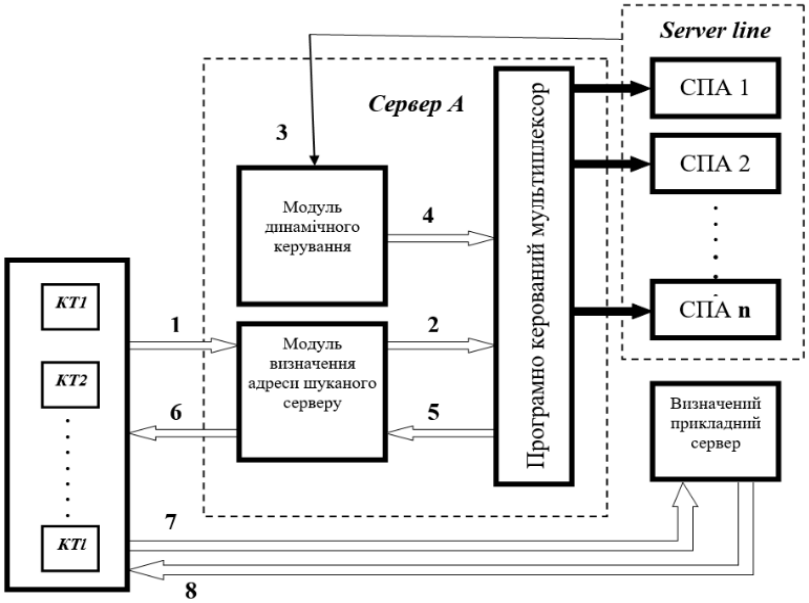

Рис. 1. Узагальнена структурно-функціональна модель розподілу звернень користувачів між серверами інтерактивної системи

Разом з лінійкою серверів *Server line*, що мають функціонувати одночасно і незалежно один від одного, виконуючи одне і те ж завдання, у склад обладнання інтерактивної системи (рис. 1), уводиться програмний *Сервер А*.

Цей сервер у реальному часі здійснює динамічний розподіл потоку клієнтських звернень між серверами СПА1, …., СПА*n* з тим, щоб забезпечити рівномірне навантаження на ці сервери в умовах значних пульсацій трафіка. Кожен із пошукових серверів СПА, що входить до складу лінійки *Server line*, виконує функцію пошуку адреси прикладного сервера, який має здійснювати обробку звернення і відповідати на запит, що міститься у зверненні. Отже, потік клієнтських звернень від термінальних пристроїв користувачів інтерактивної системи *КТ*1, *КТ*2, … *КТl* (*l* — кількість користувачів цієї системи) через мережу Інтернет надходить до С*ервера А*, що містить комплекс програм попередньої обробки цього потоку (дія 1). Одна з програм сервера *А* послідовно у реальному часі виділяє ідентифікаційні дані та запит відправника кожного звернення, що надходить, і перенаправляє ці дані до іншої програми, яка визначає поточний номер одного з серверів лінійки, що має на цьому проміжку часу обробляти виділені дані (дія 2). Програмно керований демультиплексор розподіляє у реальному часі потік звернень користувачів між серверами лінійки (дії 3 та 4), реалізуючи механізм розподілу. Визначений пошуковий сервер знаходить IP-адресу прикладного сервера, що має обробляти запит, а *Модуль визначення адреси шуканого серверу*, що реалізований на *Сервері А*, сприймає цю адресу з лінійки *Server Line* (дія 5), яка визначена механізмом розподілу, і відправляє її на термінал користувача (дія 6). Одразу після цього *Сервер А* готується до обслуговування нового запиту від іншого користувача, переходячи до виконання дії 1. Отримавши *IP*-адресу визначеного прикладного сервера, користувач звертається за цією адресою для отримання від цього сервера в інтерактивному режимі результату обробки його запиту (дія 7). Визначений сервер здійснює розв'язання прикладної задачі і результат пересилає користувачу (дія 8).

Зокрема, з рис. 1 випливає, що обробкою звернень у паралель займається лінійка спеціалізованих програмно-апаратних серверів СПА1, …, СПА*n* (*n* — кількість серверів на *Server Line*). Шля-

хом вибору кількості серверів у лінійці забезпечується відповідне узгодження інтенсивності трафіка запитів із продуктивною спроможністю інтерактивної системи.

Структурно-функціональна модель балансування навантаження між пошуковими серверами інтерактивної системи

На рис. 2 подана структурно-функціональна модель обладнання, що реалізує механізм вирівнювання навантаження між серверами лінійки *Server Line* інтерактивної системи. Ця модель враховує умови, коли пульсуючий потік звернень від клієнтів інтерактивної системи є випадковим, нестаціонарним та непрогнозованим. У цьому випадку процес вирівнювання коефіцієнтів завантаження серверів лінійки *Server Line* вважається процесом передачі у реальному часі на обробку певної частки вхідного потоку клієнтських звернень від більше завантажених до менш завантажених серверів цієї лінійки з тим, щоб наблизити рівні їхньої завантаженості до приблизно однакових величин. Структура моделі (рис. 2), подана як сукупність певним чином пов'язаних між собою пронумерованих функціональних елементів.

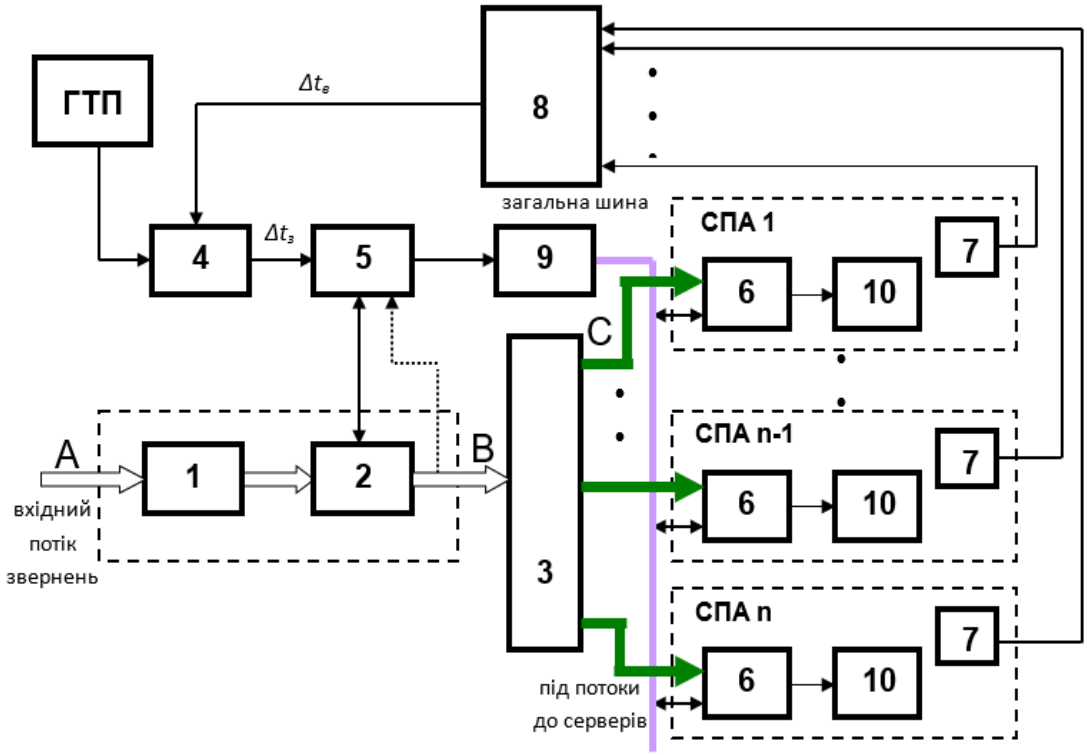

Рис. 2. Структурно-функціональна модель балансування навантаження на сервери інтерактивної системи, що одночасно працюють

На рис. 2 використані такі позначення функціональних елементів: ГТП — генератор тактових послідовностей; 1 — згладжувач вхідного потоку клієнтських звернень; 2 — формувальник квазістаціонарних відрізків вхідного потоку звернень на часових проміжках $\Delta t_з$ — кроках згладжування (процес формування реалізується як покроковий ітеративний процес з кроком $\Delta t_з$, при цьому відслідковуються тенденції у змінах величини інтенсивності вхідного потоку звернень); 3 — демультиплексор сформованого квазістаціонарного вхідного потоку звернень на кожному кроці згладжування $\Delta t_з$; 4 — задавальник величин кроків згладжування та вирівнювання (йдеться про процес вирівнювання між собою поточних значень коефіцієнтів завантаження пошукових серверів, що входять до складу лінійки *Server Line*), що реалізується у вигляді програмно керованих генераторів тактових послідовностей сигналів; 5 — вимірювач поточних значень інтенсивності сформованого квазістаціонарного потоку звернень на кожному кроці згладжування $\Delta t_з$; 6 — засоби буферизації звернень (утворення черги зі звернень, що надходять на обробку *i*-м сервером СПА) на вході кожного сервера лінійки *Server Line*; 7 — вимірювачі поточних значень коефіцієнтів завантаження кожного сервера лінійки *Server Line* на кожному кроці вирівнювання $\Delta t_в$; 8 — визначальник поточних значень матриці регулювальних зв'язків між коефіцієнтами навантаження на сервери лінійки *Server Line* на кожному кроці вирівнювання $\Delta t_в$; 9 — визначальник конкретних значень величини

частки ресурсу (тобто, кількості звернень), яка має розподілятися між вхідними чергами пошукових серверів на кожному кроці вирівнювання; 10 — оброблювачі даних прикладної задачі на серверах лінійки *Server Line*.

Позначення потоків звернень такі: A — вхідний потік звернень; B — сформований потік звернень; C — підпотоки звернень після демультиплексування; $\Delta t_з$ — тривалість кроку згладжування; $\Delta t_в$ — тривалість кроку вирівнювання.

Структурно-функціональна схема формування потоку звернень на обробку лінійкою серверів *Server Line*

Для коректної роботи цього методу балансування навантаження необхідно, щоб вхідний трафік клієнтських звернень перетворити у послідовність квазістаціонарних ділянок з реалізаціями дискретного випадкового процесу, що певною мірою піддаються згладжуванню за допомогою спеціальних процедур усереднення.

У цьому випадку технологія балансування навантаження на сервери лінійки *Server Line* передбачає необхідність належного формування потоку звернень, зокрема забезпечення узгодженості інтервалів стаціонарності цього потоку ΔT_c з інтервалами кроків дискретного управління процесом вирівнювання значень коефіцієнтів завантаження серверів $\tau_к$. Наявні методи формування трафіку такої можливості не передбачають. Зокрема, один з найпоширеніших, метод «відра токенів» [1] має суттєве обмеження щодо сфери застосування — тільки для випадків, коли реальний трафік має ознаки стаціонарного випадкового процесу. Проте реальний трафік та його похідні варто вважати нестаціонарним дискретним процесом, тому безпосереднє використання методу «відра токенів» (а також інших відомих методів формування трафіку) в системах динамічного перерозподілу навантаження на сервери навряд чи є виправданим.

Авторами пропонується структурно-функціональна схема обладнання, що призначена для формування потоку звернень, яку доцільно використати як складовий елемент запропонованої технології балансування навантаження на лінійку серверів *Server Line*. Ця схема показана на рис. 3.

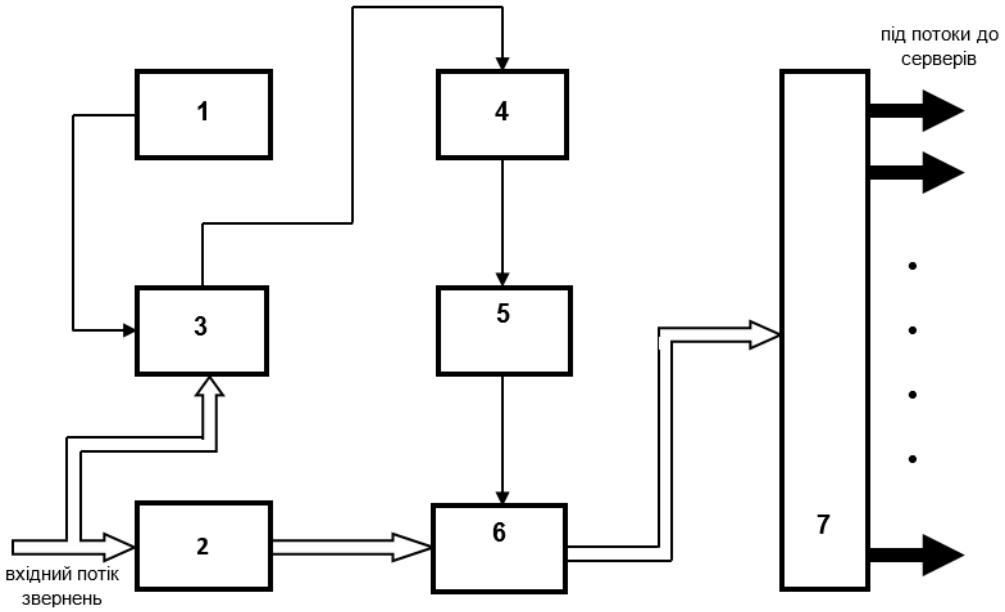

Рис. 3. Структурно-функціональна схема формування потоку клієнтських звернень на обробку лінійкою серверів *Server Line*: 1 — генератор величини кроку згладжування $\Delta t_з$; 2 — буфер черги звернень на вході інтерактивної системи (тобто, вхідний накопичувач звернень); 3 — вимірювач кількості звернень, що надійшли на вхід системи балансування протягом тривалості одного кроку згладжування $\Delta t_з$; 4 — генератор токенів (тобто, віртуальних подій пропуску звернень через шлюз); 5 — «відро токенів» (тобто, накопичувач віртуальних подій пропуску звернень через шлюз); 6 — шлюз пропуску звернень на вхід демультиплексора; 7 — демультиплексор вхідного потоку звернень

Алгоритм відра токенів широко висвітлений у публікаціях (наприклад, у [1]), але щодо відносно вузьких сфер його застосування. Авторами модифіковано схему роботи алгоритму з тим, щоби його можна було включити у контур системи балансування навантаження.

Структурно-функціональна схема оптимальної обробки звернень шляхом вирівнювання навантаження між серверами лінійки *Server Line*

Оскільки тривалість обробки кожного окремо взятого звернення є випадковою непрогнозованою величиною, то поточні значення коефіцієнтів завантаження пошукових серверів у реальному часі будуть випадковим чином різнитись. За цих умов вирівнювання коефіцієнтів завантаження серверів є доцільним. Структурно-функціональна схема вирівнювання навантаження на пошукові сервери показана на рис. 4.

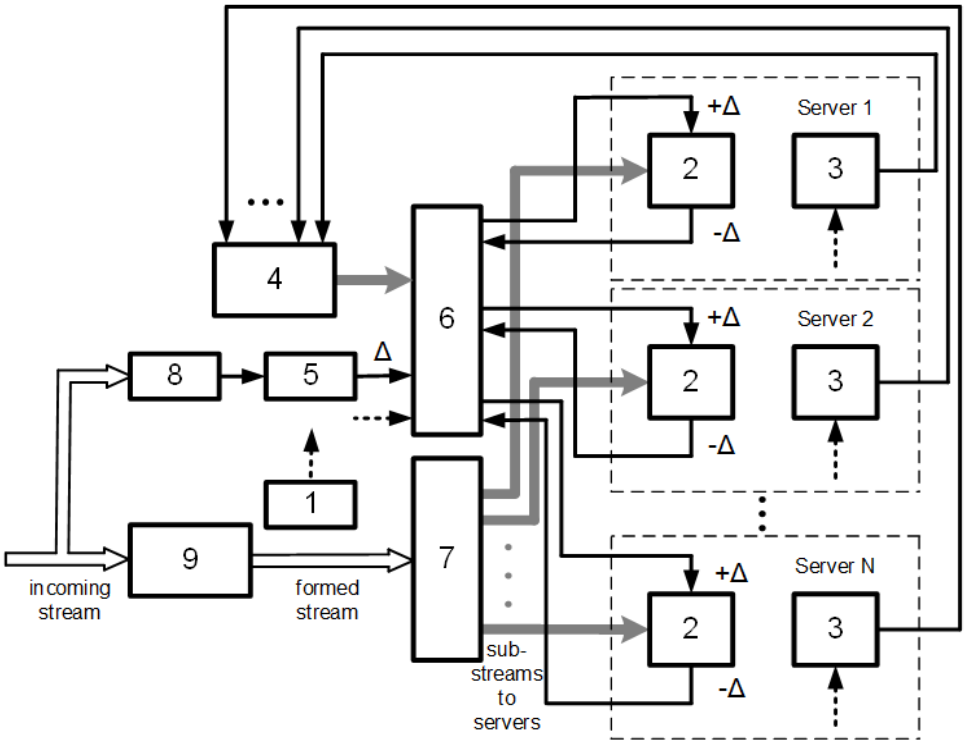

Рис. 4. Структурно-функціональна схема вирівнювання навантаження на пошукові сервери:
1 — генератор (задавальник) величини кроку вирівнювання $\Delta t_в$; 2 — накопичувач звернень (тобто, буфер черги звернень) на вході пошукового сервера; 3 — вимірювач поточного значення коефіцієнта завантаження пошукового сервера (вимірювання мають виконуватися на кожному кроці вирівнювання); 4 — визначальник матриці регулювальних зв'язків між пошуковими серверами (як результат розв'язання рівняння настроювання); 5 — визначальник величини часток ресурсу Δ (у цьому випадку це кількість звернень), що мають перерозподілятися на кожному $\Delta t_в$ між кожною парою пошукових серверів відповідно до змісту матриці регулювальних зв'язків); 6 — розподільник звернень між чергами на входах пошукових серверів; 7 — демультиплексор вхідного потоку звернень; 8 — вимірювач кількості звернень, що надійшли на вхід системи балансування протягом тривалості одного кроку згладжування $\Delta t_з$; 9 — структурно-функціональна схема формування потоку звернень, що відображена на рис. 3; 10 — визначальник моменту підключення/відключення додаткового/зайвого сервера до/від лінійки *Server Line*

Програмна реалізація методу динамічного оптимального розподілу потоку звернень у багатосерверних інтерактивних системах

Для перевірки працездатності запропонованого методу розроблено програмну модель, яка реалізує динамічне балансування навантаження між серверами інтерактивної системи. Реалізація виконана мовою Python з використанням багатопотокової обробки та асинхронного керування подіями. Задачею моделі є демонстрація процесів: прийому запитів клієнтів у реальному часі; динамічного вимірювання поточного завантаження кожного сервера; перерозподілу частини запитів з перевантажених серверів до менш завантажених; автоматичного підключення/відключення додаткових серверів за необхідності.

Реалізація складається з таких модулів: RequestGenerator — імітує нестаціонарний потік клієнтських звернень (сплески трафіку, паузи, квазістаціонарні ділянки);

ServerNode — моделює сервер, який має чергу запитів, середній час обробки та коефіцієнт завантаження; LoadBalancer — реалізує динамічний алгоритм перерозподілу запитів на основі вимірювання поточних коефіцієнтів навантаження; Controller — забезпечує додавання або виключення

серверів залежно від рівня черг.

Фрагмент коду реалізації

```
import random
import time
from threading import Thread, Lock

class ServerNode:
    def __init__(self, server_id):
        self.id = server_id
        self.queue = []
        self.load = 0.0
        self.lock = Lock()

    def process(self):
        """Обробка запитів на сервері"""
        while True:
            time.sleep(0.5)
            with self.lock:
                if self.queue:
                    self.queue.pop(0)
                self.load = len(self.queue) / 10.0 # нормований коефіцієнт завантаження

class LoadBalancer:
    def __init__(self, servers):
        self.servers = servers

    def distribute(self, request):
        """Розподіл нового запиту"""
        target = min(self.servers, key=lambda s: s.load)
        with target.lock:
            target.queue.append(request)

    def rebalance(self):
        """Динамічне балансування"""
        avg_load = sum(s.load for s in self.servers) / len(self.servers)
        for s in self.servers:
            if s.load > avg_load * 1.3:  # перевантажений
                excess = int((s.load - avg_load) * 10)
                for _ in range(excess):
                    if s.queue:
                        req = s.queue.pop()
                        target = min(self.servers, key=lambda x: x.load)
                        target.queue.append(req)

class RequestGenerator(Thread):
    def __init__(self, balancer):
        super().__init__()
        self.balancer = balancer

    def run(self):
        """Генерація запитів з пульсаціями"""
        while True:
            num_requests = random.randint(1, 10)  # нестаціонарність потоку
            for _ in range(num_requests):
                self.balancer.distribute({"timestamp": time.time()})
            self.balancer.rebalance()
            time.sleep(random.uniform(0.2, 1.0))  # пульсаційний трафік
```

```
# --- Ініціалізація ---
servers = [ServerNode(i) for i in range(3)]
for s in servers:
    Thread(target=s.process, daemon=True).start()

balancer = LoadBalancer(servers)
gen = RequestGenerator(balancer)
gen.start()
```

Алгоритмічна логіка реалізації:

1. Формування потоку звернень. Потік запитів формується з випадковою інтенсивністю, що імітує пульсаційний нестаціонарний трафік.

2. Первинний розподіл. Нові запити надходять до найменш завантаженого сервера.

3. Моніторинг завантаження. Кожний сервер у реальному часі оновлює свій коефіцієнт завантаження.

4. Динамічне балансування. Якщо навантаження на окремий сервер перевищує середнє більше ніж на 30 %, частина його запитів перекидається на інші вузли.

5. Масштабування. Механізм можна доповнити контролером, який у разі переповнення черг додає новий сервер до списку servers (автоскейлінг).

Результати експериментального моделювання

У таблиці подано результати експериментального моделювання, що порівнює показники системи без та з використанням методу динамічного оптимального розподілу потоку звернень у багатосерверних інтерактивних системах.

Результати експериментального моделювання

Показник	Без методу	За методом
Середній час обробки звернення, мс	185	96
Максимальна довжина черги, запитів	48	19
Втрати звернень, %	7.5	0.8
Середнє завантаження серверів, %	76	82
Відхилення навантаження між серверами, %	32	8

Результати моделювання підтверджують ефективність запропонованого методу динамічного оптимального розподілу потоку звернень у багатосерверних інтерактивних системах. Середній час обробки звернення зменшився з 185 мс до 96 мс (зниження майже вдвічі). Це свідчить про покращення реактивності системи завдяки вирівнюванню навантаження між серверами та скороченню часу очікування в чергах. Максимальна довжина черги запитів зменшилася більше ніж удвічі — з 48 до 19 звернень. Це означає, що запити не накопичуються на окремих вузлах, а система здатна оперативно розподіляти трафік відповідно до поточного завантаження. Втрати звернень знизилися з 7,5 % до 0,8 %, тобто майже в десять разів. Це безпосередньо пов’язано з можливістю динамічного перерозподілу запитів і запобігання переповненню буферів. Середнє завантаження серверів зросло з 76 % до 82 %, що свідчить про ефективніше використання наявних апаратних ресурсів.

Метод дозволяє рівномірно розподілити потік звернень, уникаючи простоїв одних серверів у разі перевантаження інших. Відхилення навантаження між серверами зменшилося з 32 % до 8 %, тобто у 4 рази. Це є ключовим показником того, що запропонований метод забезпечує збалансовану роботу всієї системи та підтримує стабільну продуктивність навіть за умов пульсуючого трафіку.

Графік (рис. 5) показує, що використання розробленого методу дає суттєве покращення продуктивності — зменшується середній час обробки, черги й втрати, а також досягається рівномірніше завантаження серверів.

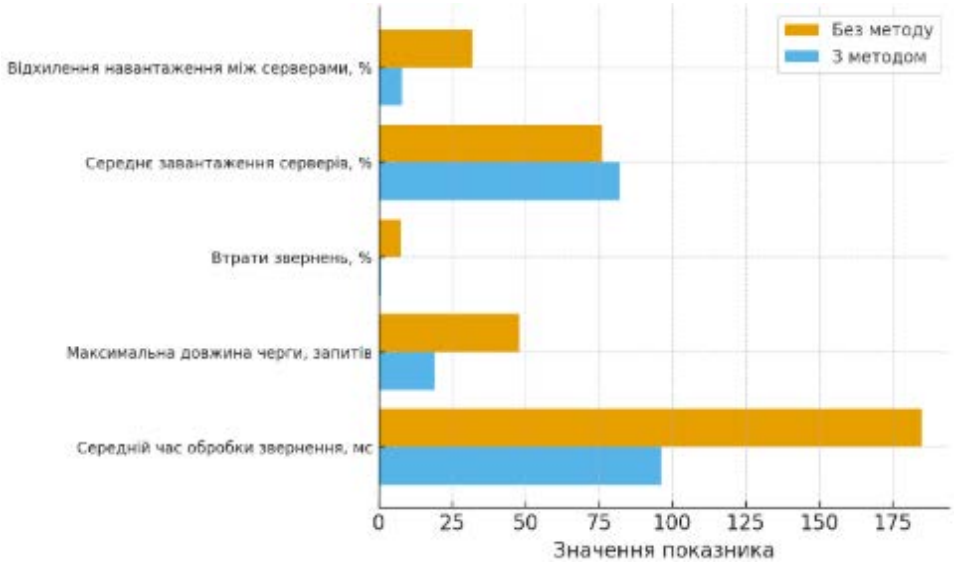

Рис. 5. Порівняння результатів моделювання без/з запропонованим методом

Висновки

У статті запропоновано метод динамічного оптимального розподілу потоку звернень у багатосерверних інтерактивних системах, який забезпечує адаптивне балансування навантаження між серверами в умовах нестаціонарного та непередбачуваного трафіку. Розроблено структурно-функціональні схеми механізмів розподілу запитів, згладжування нестаціонарного потоку та вирівнювання завантаження серверів у режимі реального часу. Запропоновано модифікований алгоритм типу «відро токенів», що дає змогу перетворювати випадкові потоки запитів, що пульсують, у квазістаціонарні інтервали, придатні для ефективного дискретного керування. Також розглянуто алгоритм динамічного перерозподілу клієнтських звернень між серверами на основі поточних коефіцієнтів завантаження з можливістю автоматичного підключення або відключення серверів залежно від інтенсивності трафіку.

Запропонований метод можна використовувати для проєктування систем реального часу, центрів оброблення даних, VoIP-, IoT- та 5G-рішень, де необхідно забезпечити стабільну продуктивність за змінних навантажень. Програмна реалізація підтвердила ефективність розробленого підходу — алгоритм забезпечує адаптивне балансування навантаження, знижує ризик перевантажень і втрат запитів, а також може бути інтегрований у системи реального часу, SDN-платформи, VoIP-сервери чи центри оброблення даних.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

[1] N. Olifer, V. Olifer, “Computer Networks: Principles,” *Technologies and Protocols for Network Design*, NJ: Wiley, 2005. 1008 с. [Electronic resource]. Available: https://link.springer.com/book/10.1007/978-3-031-17601-2 .

[2] F. Miguelez, J. Doncel, and B. J. Prabhu, “Load-balancing for multi-skilled servers with Bernoulli routing,” *Ann Oper Res*, no. 312, pp. 949-971, 2022. https://doi.org/10.1007/s10479-022-04532-7 .

[3] Kaur Shubhinder, and Kaur Gurpreet, “A Review of Load Balancing Strategies for Distributed Systems,” *International Journal of Computer Applications*, no. 121, pp. 45-47, 18, July 2015. https://doi.org/10.5120/21644-4985 .

[4] J. Zhou, U. K. Lilhore, *et al.* “Comparative analysis of metaheuristic load balancing algorithms for efficient load balancing in cloud computing,” *J Cloud Comp. no.* 12, 85, 2023. https://doi.org/10.1186/s13677-023-00453-3 .

[5] K. A. Vani, and K. N. RamaMohanBabu, “An Intelligent Server load balancing based on Multi-criteria decision-making in SDN”, *IJECES*, vol. 14, no. 4, pp. 433-442, Apr. 2023. [Electronic resource]. Available: https://ijeces.ferit.hr/index.php/ijeces/article/view/1946 .

Рекомендована кафедрою інфокомунікаційних систем і технологій ВНТУ

Дата надходження 20.01.2026
Дата прийняття до друку після рецензування 27.03.2026
Дата публікації 7.07.2026

Гнатюк Віктор Олександрович — канд. техн. наук, доцент, завідувач кафедри телекомунікаційних та радіоелектронних систем, e-mail: viktor.hnatiuk@npp.kai.edu.ua. https://orcid.org/0000-0002-4916-7149 .

Державний університет «Київський авіаційний інститут»; ДержНДІ технологій кібербезпеки, Київ;

Зандер Костянтин Юрійович — аспірант кафедри телекомунікаційних та радіоелектронних систем, e-mail: 8390983@stud.kai.edu.ua . https://orcid.org/0009-0006-4944-9249 .

Державний університет «Київський авіаційний інститут», Київ

V. O. Gnatyuk[1, 2]
K. Yu. Zander[1]

Method of Dynamic Optimal Distribution of the Applications Flow in Multi-Server Interactive Systems

[1]State University «Kyiv Aviation Institute»;

[2]State Scientific and Research Institute of Cybersecurity Technologies and Information Protection, Kyiv

The paper presents a dynamic optimal method of request flows distribution in multi-server interactive systems aimed at achieving adaptive load balancing under conditions of non-stationary and unpredictable traffic. The proposed approach ensures real-time adjustment of the request flow among parallel servers based on the current load coefficients, thereby maintaining system stability and minimizing the probability of overloads. A structural-functional model of the proposed method is developed, consisting of modules for traffic smoothing, dynamic demultiplexing, and load equalization across the server line.

A modified version of the "token bucket" algorithm is introduced to convert a non-stationary and pulsating incoming flow into quasi-stationary traffic segments that can be processed by discrete control mechanisms. The model continuously measures instantaneous load parameters for each server and performs iterative redistribution of queued requests between overloaded and underloaded nodes. In case of critical congestion, additional servers can be automatically activated, while in periods of low intensity, redundant servers are temporarily deactivated to optimize resource utilization.

The proposed method can be applied to the design of real-time systems, cloud data centers, VoIP and IoT platforms, and 5G core networks, where maintaining stable performance under variable loads is a critical requirement. The approach enhances system scalability, reduces latency and request loss probability, and provides a foundation for implementing intelligent software-based controllers for adaptive load balancing. The proposed software implementation confirmed the efficiency of the developed method for dynamic optimal distribution of the requests flow in multi-server interactive systems. The algorithm provides adaptive load balancing, reduces the probability of overloads and loss of requests, and can also be integrated into real-time systems, SDN platforms, VoIP servers or data centers.

Keywords: load balancing, interactive system, multiserver architecture, optimal distribution method, dynamic redistribution, token bucket algorithm.

Gnatyuk Viktor O. — Cand. Sc. (Eng.), Associate Professor, Head of the Chair of Telecommunications and Radioelectronic Systems, e-mail: viktor.hnatiuk@npp.kai.edu.ua ; https://orcid.org/0000-0002-4916-7149 .

Zander Kostyantyn Yu. — Post-Graduate Student of the Chair of Telecommunications and Radioelectronic Systems, e-mail: 8390983@stud.kai.edu.ua . https://orcid.org/0009-0006-4944-9249