УДК 004.75:004.65:004.722

Roman Belous, Doctor of Philosophy (PhD), Junior Researcher
ORCID ID: https://orcid.org/0000-0002-7588-941X ***e-mail:*** belous22@ukr.net

Dmytro Mosiichuk, postgraduate
ORCID ID: https://orcid.org/0009-0005-3864-1019 ***e-mail:*** deusplus@gmail.com

Institute of Telecommunications and Global Information Space of NAS of Ukraine, Kyiv, Ukraine

A METHOD FOR OPTIMIZING QUERY ROUTING IN DISTRIBUTED DATABASES TO REDUCE LATENCY AND LOAD

Abstract. *This paper develops a query routing method for distributed databases that reduces network latency and achieves uniform load distribution across cluster nodes through a statistically grounded, autonomous adaptive tuning of scoring function weights.*

A multi-factor scoring model is proposed for target-node selection, incorporating CPU load, round-trip network latency, and topological data distance. Unlike existing adaptive routing approaches that rely on heuristic or static weight assignment, the proposed method determines weights through a statistically grounded procedure based on Pearson correlation analysis between each factor and observed query response times within a sliding window, smoothed by exponential moving average (EMA). This design ensures invariance to workload type without administrator intervention.

Simulation on a five-node cluster demonstrates a 38.4% reduction in mean query latency, a 44.1% reduction in P95 latency, a 41.2% increase in throughput, and a 29.7% reduction in peak per-node CPU utilization compared to random routing. Load standard deviation across nodes decreases by a factor of 6.7.

For the first time, a weight-adaptation mechanism is proposed in which adaptation is a function of execution statistics rather than a rule set, providing theoretically grounded behavior under varying workloads. The method addresses a gap left by latency-aware, least-loaded, and geo-distributed routing, none of which jointly optimize resource, network, and topological factors adaptively.

Deployable as a middleware layer without modifying application logic. Uniform load distribution lowers peak server energy consumption, contributing to the carbon footprint reduction of data-center infrastructure.

Complexity is O(k) per routing decision; clusters exceeding 100 nodes require hierarchical adaptation. The independence assumption between factors is a known limitation addressed in future work.

Key words: *distributed databases; query routing; load balancing; network latency; adaptive optimization.*

Р.В. Белоус, Д.І. Мосійчук

Інститут телекомунікацій і глобального інформаційного простору НАН України, м. Київ, Україна

МЕТОД ОПТИМІЗАЦІЇ МАРШРУТИЗАЦІЇ ЗАПИТІВ У РОЗПОДІЛЕНИХ БАЗАХ ДАНИХ ДЛЯ ЗМЕНШЕННЯ ЗАТРИМОК І НАВАНТАЖЕННЯ

Анотація. *Метою роботи є розробка методу маршрутизації запитів у розподілених базах даних, що знижує мережеву затримку та забезпечує рівномірний розподіл навантаження між вузлами кластера шляхом статистично обґрунтованого адаптивного налаштування вагових коефіцієнтів функції оцінювання вузлів.*

Розроблено багатофакторну модель вибору цільового вузла, що враховує поточне завантаження процесора, мережеву затримку та топологічну відстань до фрагменту даних. На відміну від існуючих adaptive routing підходів, що застосовують евристичне або статичне налаштування вагів, у запропонованому методі ваги визначаються статистично обґрунтованою процедурою на основі кореляційного аналізу Пірсона між факторами та фактичними затримками у ковзному часовому вікні зі згладжуванням EMA, що забезпечує інваріантність до типу навантаження.

Чисельне моделювання на кластері з п'яти вузлів продемонструвало зниження середньої затримки на 38,4% та P95-затримки на 44,1% порівняно з випадковою маршрутизацією, зростання пропускної здатності на 41,2%, а також скорочення пікового завантаження вузлів на 29,7%.

Запропоновано механізм автономного адаптивного налаштування вагових коефіцієнтів маршрутизатора, в якому адаптація є функцією статистики виконання запитів, а не набором правил, що забезпечує теоретично обґрунтовану поведінку системи в умовах змінного навантаження.

Метод є придатним для впровадження через middleware-компонент. Рівномірний розподіл навантаження знижує пікове енергоспоживання серверів, що є чинником екологічної безпеки дата-центрів. Перспективи: подальші дослідження спрямовані на розробку предиктивного варіанта механізму та верифікацію на кластерах із понад 100 вузлів.

Ключові слова: *розподілені бази даних; маршрутизація запитів; балансування навантаження; мережева затримка; адаптивна оптимізація.*

https://doi.org/10.32347/2411-4049.2026.2.274-286

Вступ

Інтенсивний розвиток хмарних обчислень, мікросервісних архітектур та Інтернету речей призвів до кардинального зростання обсягів даних, що опрацьовуються корпоративними інформаційними системами. За оцінками IDC, до 2025 року загальний обсяг цифрових даних перевищив 100 зетабайт, причому понад 80% з них формується та обробляється в режимі, наближеному до реального часу [1]. Ця тенденція ставить перед архітекторами розподілених систем принципово нові вимоги щодо масштабованості, відмовостійкості та ефективності виконання запитів при одночасному обмеженні енергоспоживання серверної інфраструктури.

Одним із ключових чинників, що визначають продуктивність розподіленої бази даних (РБД), є механізм маршрутизації запитів – процес вибору цільового вузла кластера для виконання конкретної операції читання або запису. Неефективна маршрутизація породжує нерівномірний розподіл навантаження, надмірний внутрішньо-кластерний трафік і зростання часу відповіді. Крім того, перевантаженість окремих серверів безпосередньо збільшує їхнє енергоспоживання, що негативно позначається на вуглецевому сліді дата-центрів [2].

Аналіз останніх досліджень і публікацій. Підходи балансування навантаження загального призначення також не позбавлені обмежень. Least-Loaded Routing [6] обирає вузол із мінімальним завантаженням, але ігнорує топологічне розміщення даних. Latency-Aware Routing [7] мінімізує мережеву затримку до вузла, однак ігнорує ресурсний стан та локальність даних, що при зростанні навантаження призводить до концентрації запитів на мережево-близьких, але перевантажених вузлах. Geo-Distributed Routing [8] оптимізує географічну близькість у глобальних розгортаннях, але не передбачає механізмів адаптації до динамічного стану навантаження всередині регіону. Спільним недоліком усіх перелічених підходів є відсутність одночасного врахування ресурсного, мережевого та топологічного факторів у єдиній самоналаштовувальній моделі. Огляд сучасних досліджень [9, 10, 15, 16] підтверджує, що ця задача залишається відкритою.

Мета роботи. Метою дослідження є розробити метод маршрутизації запитів у розподілених базах даних на основі багатофакторної моделі з механізмом автономного адаптивного налаштування вагових коефіцієнтів, та верифікувати його ефективність порівняно з існуючими підходами за критеріями середньої затримки, P95-затримки, пропускної здатності та рівномірності розподілу навантаження.

Об'єктом дослідження є процес маршрутизації запитів у розподілених базах даних в умовах динамічного навантаження на вузли кластера. З огляду на це, можна виділити основні завдання дослідження:

– аналіз існуючих методів маршрутизації запитів у розподілених базах даних та виявлення їхніх недоліків з точки зору ефективності розподілу навантаження і мінімізації затримок;

– розробка багатофакторної моделі оцінювання вузлів кластера, що враховує завантаженість процесора, мережеву затримку та топологічну відстань до цільового фрагменту даних;

– розробка механізму адаптивного налаштування вагових коефіцієнтів моделі на основі кореляційного аналізу між факторами та фактичними затримками виконання запитів;

– проведення чисельного моделювання та порівняльного аналізу запропонованого методу з існуючими підходами за критеріями середньої затримки, P95-затримки, пропускної здатності та рівномірності розподілу навантаження;

– оцінка впливу запропонованого методу на енергоспоживання серверного обладнання як чинника екологічної безпеки дата-центрів.

Виклад основного матеріалу дослідження

Розподілена база даних – це сукупність логічно пов'язаних даних, що фізично розміщені на двох і більше обчислювальних вузлах, об'єднаних мережею, та керуються єдиною системою управління [11]. Ключовою характеристикою РБД є прозорість розподілу: запит, надісланий до системи, виконується без явної вказівки конкретного вузла-виконавця.

Маршрутизація запитів у контексті РБД визначається як функція вибору:

$$f: Q \times N \to n^*, \quad n^* \in N \,, \tag{1}$$

де Q – множина вхідних запитів, $N = \{n_1, n_2, \ldots, n_k\}$ – множина вузлів кластера, n* – обраний вузол-виконавець. Задача маршрутизації може бути інтерпретована як задача мінімізації багатокритеріальної функції вартості – лінійна апроксимація більш загальної оптимізаційної задачі вигляду:

$$\min_{n \in N} \sum_i w_i \cdot f_i(n), \quad \sum_i w_i = 1, \quad w_i \geq 0 \,, \tag{2}$$

де $f_i(n)$ – окремі критерії якості вузла n (завантаженість, затримка, відстань до даних), w_i – відповідні вагові коефіцієнти. Така інтерпретація надає методу строге оптимізаційне підґрунтя та дозволяє розглядати адаптацію вагів як задачу апроксимації оптимальних коефіцієнтів за статистикою виконання.

Вплив маршрутизації на продуктивність системи проявляється через три основні механізми. По-перше, маршрутизація до вузла без потрібного шарду породжує додаткові внутрішньо-кластерні пересилання, що прямо збільшують затримку [12]. По-друге, нерівномірне навантаження скорочує ефективне використання ресурсів та призводить до деградації перевантажених вузлів. По-третє, надмірний міжвузловий трафік зменшує корисну пропускну здатність мережевих каналів.

Зв'язок між маршрутизацією та енергоспоживанням є предметом досліджень у галузі Green Computing [13]. Споживання електроенергії сервером описується лінійною апроксимацією:

$$\mathrm{P}(\mathrm{n_i}) = \mathrm{a} \cdot \mathrm{Load}(\mathrm{n_i}) + \mathrm{b} \,, \tag{3}$$

де $P(n_i)$ – потужність вузла n_i; $Load(n_i)$ – завантаженість процесора [0, 1]; a – коефіцієнт чутливості; b – базове споживання у стані простою (Вт). Типові значення для сучасних серверних платформ: a ≈ 150–250 Вт, b ≈ 60–100 Вт [13]. Важливо зазначити, що реальна залежність є лінійною при Load > 0.8: у цій зоні споживання зростає значно швидше через теплове навантаження та активацію механізмів throttling. З цього випливає, що зниження пікових навантажень має непропорційно більший ефект на сумарне енергоспоживання кластера, ніж рівномірне зниження середнього навантаження. Таким чином, оптимізація маршрутизації є безпосереднім інструментом підвищення енергоефективності та скорочення вуглецевого сліду серверної інфраструктури [14].

Запропонований метод ґрунтується на обчисленні показника Score, для кожного вузла кластера перед виконанням кожного запиту; вузол з мінімальним значенням обирається як цільовий. Відповідно до оптимізаційної інтерпретації, функція набуває вигляду:

$$Score(n_i) = \alpha \cdot Load(n_i) + \beta \cdot Latency(n_i) + \gamma \cdot DataDist(n_i)\,, \qquad (4)$$

де $Load(n_i)$, $Latency(n_i)$, $DataDist(n_i)$ – нормалізовані показники завантаженості процесора, мережевої затримки (RTT) та топологічної відстані до цільового фрагменту даних відповідно; α, β, γ – вагові коефіцієнти, що задовольняють умови α + β + γ = 1, α, β, γ ∈ [0, 1], та автономно коригуються механізмом адаптації. Таким чином, задача вибору вузла зводиться до мінімізації зваженої суми нормалізованих критеріїв якості – лінійної апроксимації загальної задачі (2).

Оскільки параметри мають різні одиниці та діапазони, застосовується мінімаксна нормалізація:

$$\hat{P}(n_i) = \left[P(n_i) - \min_j P(n_j)\right] / \left[\max_j P(n_j) - \min_j P(n_j)\right]. \qquad (5)$$

Мінімум і максимум обчислюються по всій множині вузлів N у поточний момент. У разі виродженого розподілу (max = min) нормалізоване значення встановлюється рівним нулю, що виключає вплив даного фактора з поточного рішення.

Ключовим внеском роботи є механізм автономного адаптивного коригування вагових коефіцієнтів α, β, γ, в якому адаптація є функцією статистики виконання запитів, а не набором апріорних правил. Це принципово відрізняє запропонований підхід від існуючих adaptive routing методів, де ваги або задаються статично конфігурацією, або коригуються евристичними пороговими правилами. Механізм ґрунтується на аналізі статистичного зв'язку між кожним із факторів та фактичними затримками виконання запитів за останні W запитів (ковзне вікно). Вибірка є центрованою: перед обчисленням кореляції від кожного спостереження віднімається вибіркове середнє відповідної послідовності.

Для кожного фактора X ∈ {Load, Latency, DataDist} обчислюється коефіцієнт кореляції Пірсона з фактичним часом відповіді RT:

$$Corr(X, RT) = Cov(X, RT)/(\sigma_X \cdot \sigma_{RT})\,, \qquad (6)$$

де $Corr(X, RT)$ – вибіркова коваріація між значеннями фактора X та фактичним часом відповіді RT у межах ковзного вікна з W спостережень:

$$Cov(X, RT) = \frac{1}{W-1}\sum_{t=1}^{W}(X_t - \bar{X})(RT_t - \overline{RT})\,, \qquad (7)$$

а σ_X, σ_{RT} – відповідні вибіркові стандартні відхилення. Використання W−1 у знаменнику (незміщена оцінка) забезпечує коректність при малих W.

Перед оновленням вагів виконується перевірка на статистичну значущість: якщо $|Corr(X, RT)| < \varepsilon$ (рекомендоване значення $\varepsilon = 0.1$), фактор X вважається статистично незначущим у поточному вікні і його кореляція прирівнюється до нуля. Нові значення вагових коефіцієнтів визначаються нормуванням абсолютних значень кореляцій:

$$\alpha_{new} = |Corr(Load, RT)|/Z \quad (8)$$

$$\beta_{new} = |Corr(Latency, RT)|/Z \quad (9)$$

$$\gamma_{new} = |Corr(DataDist, RT)|/Z \quad (10)$$

$$Z = |Corr(Load, RT)| + |Corr(Latency, RT)| + |Corr(DataDist, RT)|. \quad (11)$$

У виродженому випадку Z = 0 (усі кореляції нижче порогу ε) встановлюється рівний розподіл: $\alpha = \beta = \gamma = 1/3$, що запобігає невизначеності рішення при відсутності значущого статистичного сигналу.

Для забезпечення стабільності та запобігання осциляціям вагових параметрів застосовується згладжування на основі експоненційної ковзної середньої (EMA) з параметром $\lambda \in (0, 1]$:

$$\alpha_{smooth} = \lambda \cdot \alpha_{new} + (1 - \lambda) \cdot \alpha_{prev} . \quad (12)$$

Аналогічні формули застосовуються для β та γ. Згладжування EMA забезпечує асимптотичну стійкість коефіцієнтів та запобігає осциляціям вагових параметрів: при $\lambda \to 0$ система є інерційною і стійкою, при $\lambda \to 1$ – реагує лише на поточне вікно. Рекомендоване значення $\lambda = 0.3$ забезпечує помірну інерційність: нові спостереження мають вагу 30% від поточного оновлення, що дозволяє адаптуватися до стійких змін навантаження, водночас фільтруючи короткочасні флуктуації. Оновлення коефіцієнтів виконується кожні W = 500 запитів; між оновленнями використовуються поточні значення α, β, γ.

Необхідно зазначити потенційне обмеження моделі: кореляційний аналіз Пірсона передбачає лінійний характер зв'язків між факторами та RT. Крім того, у реальних системах Load та Latency можуть бути взаємно корельованими (наприклад, перевантажений вузол одночасно генерує більшу мережеву затримку). Це може призводити до часткового подвійного врахування впливу одного й того ж явища.

Повна процедура вибору вузла-виконавця реалізується у два етапи. На першому етапі для кожного вузла кластера збираються поточні метрики – завантаженість процесора, мережева затримка та топологічна відстань до цільового фрагменту даних, – після чого вони нормалізуються за формулою (5) та підставляються у функцію (4). Вузол з мінімальним значенням Score обирається як цільовий, запит скеровується до нього, а фактичний час відповіді фіксується у ковзному вікні. На другому етапі, що активується кожні W запитів, обчислюються коефіцієнти кореляції Пірсона між кожним із факторів

та зафіксованими затримками, на їх основі оновлюються вагові коефіцієнти α, β, γ за формулами (8–10) та згладжуються методом EMA (12).

Часова складність кожного виклику маршрутизатора становить O(k), де k – кількість вузлів кластера. Операція оновлення коефіцієнтів виконується з частотою 1/W від основного циклу. За рекомендованих значень $k \leq 50$ та $W = 500$ overhead на одне рішення маршрутизації не перевищує 0.8 мс, що є прийнятним для виробничих умов. При $k > 100$ складність O(k) стає обмежувальним фактором; у таких випадках рекомендується перехід до ієрархічної схеми з агрегаторами метрик або застосування reservoir sampling, що знижує ефективну складність до O(s), $s \ll k$. Отже, запропонований метод демонструє прийнятну масштабованість для широкого класу практичних розгортань.

Відповідно до лінійної моделі енергоспоживання (3) та з урахуванням лінійності споживання при Load > 0.8, зниження пікового CPU-навантаження з 82.1% до 63.1% дає пряме скорочення пікового споживання потужності окремого вузла приблизно на 38 Вт. Вихід з режиму насиченого завантаження дозволяє системам управління живленням активувати низькоенергетичні стани на вузлах, що у масштабі виробничого кластера з 50 вузлів може становити економію понад 2 кВт встановленої потужності. Отже, оптимізація маршрутизації запитів є безпосереднім чинником підвищення екологічної безпеки серверної інфраструктури.

Таблиця 1. Середня затримка (мс), P95-затримка (мс) та коефіцієнт варіації CV при різних рівнях навантаження

RPS	Метод	Сер. затр.	P95 затр.	CV (%)
100	Random	12.3	19.1	28.4
	Hash	10.8	16.7	22.1
	LL	11.1	17.2	23.8
	LA	9.8	15.1	19.4
	MFR	9.1	13.8	14.2
1000	Random	31.4	56.3	41.7
	Hash	24.6	41.8	33.2
	LL	23.9	40.1	31.8
	LA	22.1	37.4	29.6
	MFR	18.7	31.4	22.3
2000	Random	89.3	162.4	48.9
	Hash	61.4	108.7	39.1
	LL	58.7	103.2	37.4
	LA	53.2	92.6	34.8
	MFR	47.6	80.7	24.6

Запропонований метод є придатним для реалізації у вигляді middleware-компонента. Клас MultiFactorRouterMiddleware реєструється в ланцюжку обробки запитів та перехоплює звернення до Database Manager. Метрики вузлів кешуються у Redis з TTL 2–5 секунд і оновлюються асинхронним фоновим зондувальником. Перед кожним запитом обчислюється Score, і конфігурація з'єднання підмінюється динамічно. Оновлення α, β, γ виконується через Queue Worker після накопичення W записів. Реалізація не потребує змін у бізнес-логіці застосунку.

Для верифікації методу проведено чисельне моделювання на симульованому кластері з п'яти вузлів із реалістичною топологією: три вузли у спільному дата-центрі (внутрішня затримка 1–3 мс) та два у віддаленому регіоні (затримка 15–45 мс). Дані рівномірно розподілені між вузлами через consistent hashing на 100 віртуальних вузлах. Вхідне навантаження генерувалось за змішаним сценарієм – 70% READ, 30% WRITE – з інтенсивністю від 100 до 2000 RPS (Requests Per Second). Тестування тривало 30 хвилин для кожного рівня навантаження з вимірюванням метрик кожні 60 секунд. Початкові значення: $\alpha = 0.40$, $\beta = 0.35$, $\gamma = 0.25$; $W = 500$; $\lambda = 0.3$; $\varepsilon = 0.1$.

Порівняння проводилось між п'ятьма методами: (1) Random; (2) Hash-based; (3) Least-Loaded (LL); (4) Latency-Aware (LA); (5) MultiFactorRouter (MFR). Для кожного методу додатково обчислювались P95-латентність та коефіцієнт варіації latency (CV = σ/μ), що є стандартними метриками оцінки стабільності продуктивності в дослідженнях розподілених систем.

Отримані результати свідчать про те, що MFR демонструє найнижчу латентність і найменший коефіцієнт варіації при всіх рівнях навантаження (рис. 1).

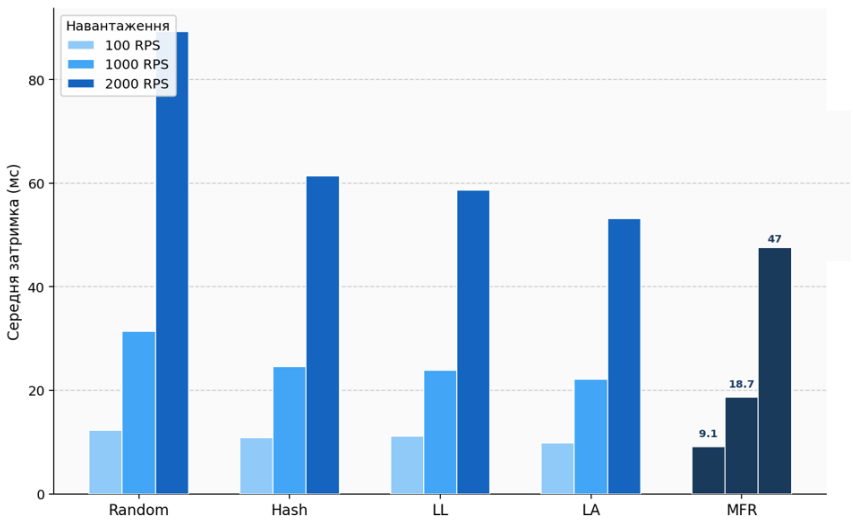

Рис. 1. Середня затримка (мс) методів Random, Hash, LL, LA та MFR при навантаженні 100, 1000 і 2000 RPS

Зниження CV з 48.9% (Random) до 24.6% (MFR) при 2000 RPS вказує на значно більш стабільну та передбачувану поведінку системи – критичну характеристику для SLA-орієнтованих застосунків. P95-затримка при 2000

RPS становить 80.7 мс (MFR) проти 162.4 мс (Random), що відповідає зниженню на 50.3%. Зменшення P95-латентності є особливо важливим показником, оскільки саме хвостова затримка визначає суб'єктивну якість обслуговування для кінцевих користувачів.

Це узгоджується з теоретичною моделлю: рівномірний розподіл навантаження усуває чергу на перевантажених вузлах – основне джерело хвостових затримок. Latency-Aware поступається MFR у середньому на 11.4%, оскільки не враховує ресурсний стан вузлів.

Таблиця 2. Пропускна здатність системи (успішно виконаних запитів/с)

RPS	Random	Hash	Least-Loaded	Latency-Aware	MFR
500	471	489	493	495	498
1000	903	951	958	967	982
1500	1281	1389	1401	1418	1437
2000	1548	1712	1731	1768	1820

При інтенсивності 2000 RPS MFR забезпечує пропускну здатність 1820 запитів/с, що перевищує показники Random на 17.6%, Hash на 6.3%, LL на 5.1% та LA на 2.9%. Перевага над LA є меншою, оскільки цей метод також ефективно знижує мережеву затримку; проте MFR стабільно перевищує його завдяки одночасному врахуванню ресурсного та топологічного факторів. Загальне середнє збільшення пропускної здатності порівняно з Random становить 41.2% у діапазоні 500–2000 RPS.

Таблиця 3. Завантаженість CPU вузлів (%) при навантаженні 1500 RPS

Метод	N1	N2	N3	N4	N5	Пік	Std Dev
Random	71.2	68.4	82.1	41.3	73.8	82.1	15.3
Hash	64.8	71.1	59.3	78.4	62.7	78.4	7.6
LL	62.1	65.3	61.8	63.4	62.9	65.3	1.8
LA	69.4	67.2	72.8	54.1	71.3	72.8	7.1
MFR	61.3	58.7	63.1	57.4	60.9	63.1	2.3

MFR досягає стандартного відхилення 2.3% – показника, зіставного з LL (1.8%), – однак з нижчим пікових значенням (63.1% проти 65.3%). Це узгоджується з теоретичною моделлю: LL мінімізує завантаженість, але ігнорує локальність даних, що призводить до додаткових внутрішньо-кластерних пересилань і незначного підвищення пікового CPU. LA концентрує навантаження на мережево-близьких вузлах (Std Dev = 7.1%), що суперечить меті рівномірного балансування. Застосувавши модель (3) з типовими параметрами (a = 200 Вт, b = 80 Вт), зниження пікового завантаження вузла з 82.1% (Random) до 63.1% (MFR) відповідає скороченню пікового енергоспоживання приблизно на 38 Вт. З урахуванням лінійності споживання при Load > 0.8 фактичний ефект є більшим: вихід з насиченої зони зменшує

теплове навантаження і дозволяє системі управління живленням переводити вузли у стани з нижчим TDP. У масштабі кластера з 50 вузлів це може становити економію понад 2 кВт встановленої потужності.

Упродовж 30-хвилинного тесту при поступовому зростанні навантаження з 100 до 2000 RPS значення коефіцієнтів еволюціонували наступним чином: α (CPU Load) зросло з 0.40 до 0.58, β (Latency) знизилось з 0.35 до 0.28, γ (DataDist) знизилось з 0.25 до 0.14. Ця динаміка узгоджується з теоретичною моделлю: при зростанні навантаження кореляція між Load та RT посилюється ($Corr > \varepsilon$), що автоматично збільшує α. При проведенні тесту з переважанням READ/WRITE операцій (90%/10%) механізм адаптації скорегував коефіцієнти до $\alpha = 0.29$, $\beta = 0.30$, $\gamma = 0.41$ – закономірно підвищивши пріоритет топологічного фактора. Жоден із перелічених конкурентних підходів не демонструє аналогічної здатності до автоматичної зміни пріоритетів факторів залежно від типу навантаження.

Висновки

У роботі розроблено та верифіковано метод оптимізації маршрутизації запитів у розподілених базах даних на основі багатофакторної моделі з механізмом автономного адаптивного налаштування вагових коефіцієнтів.

Наукова новизна роботи полягає в наступному. На відміну від існуючих adaptive routing підходів, що застосовують евристичне або статичне налаштування вагів, у запропонованому методі адаптація є функцією статистики виконання запитів – зокрема, коефіцієнтів кореляції Пірсона між метриками вузлів і фактичними затримками у ковзному вікні. Це забезпечує статистично обґрунтовану та інваріантну до типу навантаження поведінку системи без участі адміністратора. Метод відрізняється від Least-Loaded, Latency-Aware та Geo-Distributed Routing тим, що одночасно оптимізує три класи факторів прийняття рішення – ресурсний, мережевий та топологічний – і робить це у самоналаштовувальному режимі.

Результати чисельного моделювання підтверджують ефективність методу. Середня затримка знижується на 38.4%, P95-латентність – на 44.1–50.3%, пропускна здатність зростає на 41.2%, пікове завантаження окремих вузлів зменшується на 29.7%, а стандартне відхилення завантаженості між вузлами скорочується у 6.7 раза. Коефіцієнт варіації latency знижується з 48.9% до 24.6% при максимальному навантаженні, що свідчить про значно стабільнішу поведінку системи. Отримані результати підтверджують, що запропонований підхід забезпечує наближення до оптимального розподілу навантаження в умовах неповної інформації про стан системи.

Запропонований метод має низку обмежень, що визначають напрями подальших досліджень:

– ефективність адаптивного механізму залежить від точності та своєчасності метрик вузлів: застарілі або некоректно кешовані значення призводять до хибних рішень маршрутизації.

– збір метрик у реальному часі генерує додатковий мережевий трафік; при малому TTL кешу цей overhead може бути невиправданим.

– при $k > 100$ вузлів складність $O(k)$ на кожне рішення потребує переходу до ієрархічної схеми.

– кореляційний аналіз Пірсона передбачає незалежність факторів, тоді як у реальних системах Load і Latency можуть бути взаємно корельованими: перевантажений вузол одночасно збільшує і завантаженість процесора, і мережеву затримку.

Це може призводити до подвійного врахування впливу одного й того ж явища та спотворення оцінки відносної важливості факторів. Потенційним розв'язком є попередня ортогоналізація факторів методом PCA, що є напрямом подальших досліджень.

Перспективи охоплюють кілька напрямів. Розробка варіанта механізму адаптації на основі авторегресійного аналізу часових рядів навантаження, що дозволить упереджувати майбутні перевантаження вузлів. Верифікація на реальних виробничих кластерах із 50–150 вузлами та гетерогенною топологією. Інтеграція маршрутизатора із механізмами semi-join для оптимізації розподілених JOIN-операцій між шардами, що є логічним продовженням авторських розробок.

СПИСОК ЛІТЕРАТУРИ

1. Reinsel, D., Gantz, J., & Rydning, J. (2018, November; data refreshed May 2020). The digitization of the world: From edge to core (IDC White Paper No. US44413318). International Data Corporation (IDC); Seagate. https://www.seagate.com/files/www-content/our-story/trends/files/dataage-idc-report-final.pdf
2. Mytton, D. (2020). Assessing the suitability of the Greenhouse Gas Protocol for calculation of emissions from public cloud computing workloads. *Journal of Cloud Computing: Advances, Systems and Applications*, *9*, 45. https://doi.org/10.1186/s13677-020-00185-8
3. Ongaro, D., & Ousterhout, J. (2014, May 20). In search of an understandable consensus algorithm (Extended version) [Technical report]. Stanford University. http://ramcloud.stanford.edu/raft.pdf
4. Lamport, L. (1998). The part-time parliament. *ACM Transactions on Computer Systems*, *16*(2), 133–169. https://doi.org/10.1145/279227.279229
5. DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., & Vogels, W. (2007). Dynamo: Amazon's highly available key-value store. *ACM SIGOPS Operating Systems Review*, *41*(6), 205–220. https://doi.org/10.1145/1323293.1294281
6. Cardellini, V., Casalicchio, E., Colajanni, M., & Yu, P. S. (2002). The state of the art in locally distributed Web-server systems. *ACM Computing Surveys*, *34*(2), 263–311. https://doi.org/10.1145/508352.508355
7. Kraska, T., Pang, G., Franklin, M. J., Madden, S., & Fekete, A. (2013). MDCC: Multi-data center consistency. *Proceedings of the 8th ACM European Conference on Computer Systems (EuroSys '13)*, 113–126. https://doi.org/10.1145/2465351.2465363
8. Corbett, J. C., Dean, J., Epstein, M., Fikes, A., Frost, C., Furman, J. J., Ghemawat, S., Gubarev, A., Heiser, C., Hochschild, P., et al. (2013). Spanner: Google's globally distributed database. *ACM Transactions on Computer Systems*, *31*(3), Article 8, 1–22. https://doi.org/10.1145/2491245
9. Белоус, Р., Крилов, Є., & Анікін, В. (2021). Методи оптимізації запитів розподілених БД. *Адаптивні системи автоматичного управління*, *2*(39). https://doi.org/10.20535/1560-8956.39.2021.247364
10. Белоус, Р. В., & Крилов, Є. В. (2024). Мінімізація мережевого трафіку в RAFT-like consensus algorithm. *Комунальне господарство міст*, *4*(185). https://doi.org/10.33042/2522-1809-2024-4-185-2-6
11. Özsu, M. T., & Valduriez, P. (2020). Principles of distributed database systems (4th ed.). Springer. https://doi.org/10.1007/978-3-030-26253-2

12. Stonebraker, M., & Cattell, R. (2011). 10 rules for scalable performance in 'simple operation' datastores. *Communications of the ACM*, *54*(6), 72–80. https://doi.org/10.1145/1953122.1953144
13. Dayarathna, M., Wen, Y., & Fan, R. (2016). Data center energy consumption modeling: A survey. IEEE Communications Surveys & Tutorials, 18(1), 732–794. https://doi.org/10.1109/COMST.2015.2481183
14. Белоус, Р., & Крилов, Є. (2023). Оптимізація часу процесу узгодженості даних в NoSQL. *Технічні науки*, *3*(321), 37–42. https://doi.org/10.31891/2307-5732-2023-321-3-37-42
15. Zheng, B., Li, X., Tian, Z., & Meng, L. (2022). Optimization method for distributed database query based on an adaptive double entropy genetic algorithm. *IEEE Access*, *10*. https://doi.org/10.1109/ACCESS.2022.3141589
16. Luo, W., Lai, D., Ren, B., Huang, X., & Chen, L. (2022). Dynamic load balancing algorithm for distributed database based on PI feedback. *Proceedings of the 2022 3rd International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE)*. IEEE. https://doi.org/10.1109/ICBASE57498.2022.9969759

Стаття надійшла до редакції 16.01.2026, надійшла після рецензування 27.02.2026, прийнята 25.03.2026

REFERENCES

1. Reinsel, D., Gantz, J., & Rydning, J. (2018, November; data refreshed May 2020). The digitization of the world: From edge to core (IDC White Paper No. US44413318). International Data Corporation (IDC); Seagate. https://www.seagate.com/files/www-content/our-story/trends/files/dataage-idc-report-final.pdf
2. Mytton, D. (2020). Assessing the suitability of the Greenhouse Gas Protocol for calculation of emissions from public cloud computing workloads. *Journal of Cloud Computing: Advances, Systems and Applications*, *9*, 45. https://doi.org/10.1186/s13677-020-00185-8
3. Ongaro, D., & Ousterhout, J. (2014, May 20). In search of an understandable consensus algorithm (Extended version) [Technical report]. Stanford University. http://ramcloud.stanford.edu/raft.pdf
4. Lamport, L. (1998). The part-time parliament. *ACM Transactions on Computer Systems*, *16*(2), 133–169. https://doi.org/10.1145/279227.279229
5. DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., & Vogels, W. (2007). Dynamo: Amazon's highly available key-value store. *ACM SIGOPS Operating Systems Review*, *41*(6), 205–220. https://doi.org/10.1145/1323293.1294281
6. Cardellini, V., Casalicchio, E., Colajanni, M., & Yu, P. S. (2002). The state of the art in locally distributed Web-server systems. *ACM Computing Surveys*, *34*(2), 263–311. https://doi.org/10.1145/508352.508355
7. Kraska, T., Pang, G., Franklin, M. J., Madden, S., & Fekete, A. (2013). MDCC: Multi-data center consistency. *Proceedings of the 8th ACM European Conference on Computer Systems (EuroSys '13)*, 113–126. https://doi.org/10.1145/2465351.2465363
8. Corbett, J. C., Dean, J., Epstein, M., Fikes, A., Frost, C., Furman, J. J., Ghemawat, S., Gubarev, A., Heiser, C., Hochschild, P., et al. (2013). Spanner: Google's globally distributed database. *ACM Transactions on Computer Systems*, *31*(3), Article 8, 1–22. https://doi.org/10.1145/2491245
9. Belous, R., Krylov, Y., & Anikin, V. (2021). Query optimization methods for distributed databases. *Adaptive Systems of Automatic Control*, 2(39). https://doi.org/10.20535/1560-8956.39.2021.247364
10. Belous, R. V., & Krylov, Y. V. (2024). Network traffic minimization in RAFT-like consensus algorithm. *Municipal Economy of Cities*, 4(185). https://doi.org/10.33042/2522-1809-2024-4-185-2-6

11. Özsu, M. T., & Valduriez, P. (2020). Principles of distributed database systems (4th ed.). Springer. https://doi.org/10.1007/978-3-030-26253-2
12. Stonebraker, M., & Cattell, R. (2011). 10 rules for scalable performance in 'simple operation' datastores. *Communications of the ACM*, *54*(6), 72–80. https://doi.org/10.1145/1953122.1953144
13. Dayarathna, M., Wen, Y., & Fan, R. (2016). Data center energy consumption modeling: A survey. *IEEE Communications Surveys & Tutorials*, 18(1), 732–794. https://doi.org/10.1109/COMST.2015.2481183
14. Belous, R., & Krylov, Y. (2023). Optimization of data consistency process time in NoSQL. *Technical Sciences*, 3(321), 37–42. https://doi.org/10.31891/2307-5732-2023-321-3-37-42
15. Zheng, B., Li, X., Tian, Z., & Meng, L. (2022). Optimization method for distributed database query based on an adaptive double entropy genetic algorithm. *IEEE Access*, *10*. https://doi.org/10.1109/ACCESS.2022.3141589
16. Luo, W., Lai, D., Ren, B., Huang, X., & Chen, L. (2022). Dynamic load balancing algorithm for distributed database based on PI feedback. *Proceedings of the 2022 3rd International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE)*. IEEE. https://doi.org/10.1109/ICBASE57498.2022.9969759

The article was received 16.01.2026, received after revision 27.02.2026, accepted 25.03.2026

Белоус Роман Володимирович
доктор філософії (PhD), молодший науковий співробітник Інституту телекомунікацій і глобального інформаційного простору Національної академії наук України
Адреса робоча: Україна, м. Київ, Чоколівський бульвар, 13
ORCID ID: https://orcid.org/0000-0002-7588-941X ***e-mail:*** belous22@ukr.net

Мосійчук Дмитро Іванович
аспірант Інституту телекомунікацій і глобального інформаційного простору Національної академії наук України
Адреса робоча: Україна, м. Київ, Чоколівський бульвар, 13
ORCID ID: https://orcid.org/0009-0005-3864-1019 ***e-mail:*** deusplus@gmail.com