Ольга Миколаївна ТКАЧЕНКО[1],
доктор технічни наук, професор, ORCID: https://orcid.org/0000-0001-7983-9033
Олександр Іванович ГОЛУБЕНКО[2],
кандидат технічних наук, доцент, ORCID: https://orcid.org/0000-0002-1776-5160
Сергій Миколайович КОВАЛЕНКО[2],
кандидат фізико-математичних наук, доцент; ORCID: https://orcid.org/0000-0002-8315-1589
Андрій Васильович САВЧЕНКО[2],
кандидат фізико-математичних наук, ORCID: https://orcid.org/0000-0002-8314-6034

[1] Київський національний університет імені Тараса Шевченка
[2] Заклад вищої освіти «Міжнародний науково-технічний університет імені академіка Юрія Бугая»

Прийняття: 02/05/2026
Рецензія: 12/05/2026
Публікація: 09/07/2026

УДК
004.75; 004.8

ISSN 2786-7226 (print)
ISSN 2786-7234 (online)

DOI: https://doi.org/10.53920/ITS-2026-1-3

АДАПТИВНИЙ РОЗПОДІЛ НАВАНТАЖЕННЯ В РОЗПОДІЛЕНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМАХ

Метою дослідження є розроблення та теоретичне обґрунтування адаптивних методів розподілу навантаження в розподілених обчислювальних системах для підвищення продуктивності, масштабованості та стійкості до змін навантаження. Особливу увагу приділено вдосконаленню механізмів балансування в умовах динамічного середовища, характерного для хмарних, веб- та високопродуктивних обчислювальних платформ.

У роботі використано методи системного аналізу, математичного моделювання та багатокритеріальної оптимізації. Побудовано формалізовану модель розподілу завдань між вузлами з урахуванням часу виконання, пропускної здатності, ймовірності відмов та енергоефективності. Дослідження адаптивних алгоритмів здійснювалось із застосуванням підходів теорії черг, евристичних стратегій, методів кооперативних ігор та алгоритмів машинного навчання, зокрема підкріплювального навчання. Проведено порівняльний аналіз статичних, динамічних та адаптивних методів балансування.

Отримані результати підтвердили, що адаптивні алгоритми забезпечують більш рівномірний розподіл навантаження та зменшення середнього часу виконання завдань порівняно зі статичними підходами. Запропонована модель дозволяє мінімізувати коефіцієнт дисбалансу та підвищити ефективність використання ресурсів системи в умовах

змінного навантаження. Доведено доцільність використання гібридних підходів, які поєднують моніторинг стану вузлів із прогнозуванням навантаження.

Удосконалено математичну модель багатокритеріальної оптимізації розподілу навантаження з урахуванням показників енергоефективності та надійності. Розвинуто підхід до інтеграції методів машинного навчання в алгоритми балансування для динамічних розподілених середовищ.

Результати можуть бути використані при проектуванні хмарних сервісів, систем потокової обробки даних, кластерних та edge-систем. Запропоновані підходи сприяють підвищенню продуктивності, зменшенню затримок і покращенню стійкості розподілених обчислювальних систем у реальних умовах експлуатації.

Ключові слова: *розподілені обчислення, балансування навантаження, адаптивні алгоритми, динамічні системи, хмарні обчислення.*

Olha TKACHENKO, Oleksandr GOLUBENKO, Serhii KOVALENKO, Andrii SAVCHENKO

ADAPTIVE LOAD DISTRIBUTION IN DISTRIBUTED COMPUTING SYSTEMS

The purpose of the study is to develop and theoretically substantiate adaptive load distribution methods in distributed computing systems to increase productivity, scalability, and resilience to load changes. Particular attention is paid to improving balancing mechanisms in a dynamic environment typical of cloud, web, and high-performance computing platforms.

The work uses methods of system analysis, mathematical modeling, and multi-criteria optimization. A formalized model of task distribution between nodes is constructed, taking into account execution time, throughput, failure probability, and energy efficiency. The study of adaptive algorithms was carried out using queuing theory approaches, heuristic strategies, cooperative game methods, and machine learning algorithms, in particular reinforcement learning. A comparative analysis of static, dynamic, and adaptive balancing methods was conducted.

The results obtained confirmed that adaptive algorithms provide a more uniform load distribution and a reduction in the average task execution time compared to static approaches. The proposed model allows minimizing the imbalance coefficient and increasing the efficiency of system resource use under variable load conditions. The feasibility of using hybrid approaches that combine node state monitoring with load forecasting has been proven.

The mathematical model of multi-criteria optimization of load distribution has been improved, taking into account energy efficiency and reliability indicators. An

approach to integrating machine learning methods into balancing algorithms for dynamic distributed environments has been developed.

The results can be used in the design of cloud services, streaming data processing systems, cluster and edge systems. The proposed approaches contribute to increasing productivity, reducing delays, and improving the stability of distributed computing systems in real operating conditions.

Keywords: *distributed computing, load balancing, adaptive algorithms, dynamic systems, cloud computing.*

Постановка проблеми. Розвиток сучасних інформаційних технологій призводить до стрімкого збільшення обсягів даних, що обробляються, зокрема в розподілених обчислювальних системах. Одним із ключових аспектів ефективного функціонування таких систем є груповий обмін даними, що відіграє важливу роль у підвищенні продуктивності та масштабованості обчислень. Проте традиційні методи обміну інформацією стикаються з проблемами оптимізації трафіку, балансування навантаження та зменшення затримок.

Розподілені обчислювальні системи (РОС) є основою сучасних інформаційних технологій, забезпечуючи масштабованість, надійність і високу продуктивність. Зростання популярності розподілених обчислювальних систем, таких як хмарні та граничні обчислення, паралельні обчислення та суперкомп'ютерні кластери, зумовлює необхідність вдосконалення механізмів обміну даними між вузлами системи. Неефективність традиційних алгоритмів передачі даних може призводити до перевантаження мережі, збільшення часу виконання завдань і зниження загальної продуктивності системи. Представлене дослідження підсилюється потребою у високошвидкісній передачі даних, мінімізації затримок та оптимізації використання мережевих ресурсів.

Оптимізація групового обміну даними в розподілених обчислювальних системах (РОС) є ключовою задачею для забезпечення високої продуктивності, масштабованості та надійності. Традиційні методи оптимізації, які формувалися протягом десятиліть розвитку мережевих технологій, залишаються основою для багатьох сучасних систем і слугують відправною точкою для розробки нових підходів. Ці методи зосереджені на зменшенні затримки, підвищенні пропускної здатності, забезпеченні надійності та економії ресурсів мережі. У контексті групового обміну, де дані передаються від одного джерела до множини отримувачів, традиційні методи набувають особливої ваги через складність координації та потенційні обмеження, такі як вузькі місця в пропускній здатності.

Результати дослідження. Зі зростанням складності обчислювальних завдань виникає потреба в ефективному управлінні ресурсами, зокрема у розподілі навантаження між вузлами системи. Нерівномірність завантаження призводить до деградації продуктивності та зростання затримок, що робить проблему балансування навантаження однією з ключових у проектуванні та експлуатації РОС.

Традиційні підходи до балансування (статичні чи динамічні) не завжди враховують зміни навантаження в реальному часі. Саме тому зростає значення адаптивних алгоритмів, які здатні автоматично підлаштовуватися до умов функціонування системи.

Теоретичні аспекти розподілу навантаження

Розподіл навантаження - це процес оптимізації використання ресурсів системи шляхом перенесення завдань з перевантажених вузлів на менш завантажені.

Основні критерії ефективності:

- мінімізація часу виконання завдань;
- максимізація пропускної здатності системи;
- зниження ймовірності відмов;
- підвищення енергоефективності.

Подамо математичну модель розподілу навантаження через систему формальних викладок. m

Нехай маємо множину вузлів системи

$$N = \{n_1, n_2, ..., n_m\}$$

та множину завдань

$$T = \{t_1, t_2, ..., t_k\}.$$

Навантаження на вузлі n_i визначимо як суму обчислювальних витрат закріплених за ним завдань

$$L_i = \Sigma\ (w_j),$$
$$\text{де } t_j \in T_i,$$

T_i - підмножина завдань, призначених вузлу n_i,

w_j - вага завдання (ресурсні вимоги: кількість інструкцій, обсяг даних або час процесора).

Критерії ефективності:

1) мінімізація часу виконання, при цьому загальний час виконання системи (makespan) визначається виразом

$$T\_exec = \max(L_i),\ i = 1..m,$$

з метою мінімізувати T_exec;

2) максимізація пропускної здатності, яка визначаться наступниа виразом

$$\Theta = |T| / T\_exec,$$

з метою максимізувати Θ;

3) зниження ймовірності відмов, при цьому якщо $p_i(L_i)$ - функція ймовірності відмови вузла n_i, то сумарна ймовірність відмови визначається наступним чином

$$P\_fail = 1 - \prod (1 - p_i(L_i)), i = 1..m,$$

з метою мінімізувати P_fail;

4) підвищення енергоефективності, враховуючи те, що споживана енергія системи визначається як

$$E = \Sigma f(L_i), i = 1..m,$$

де $f(L_i)$ - енергетична функція навантаження вузла,

метою є мінімізувати E.

Узагальнена задача оптимізації

Задача розподілу навантаження формулюється як багатокритеріальна оптимізація:

мінімізувати: (T_exec, P_fail, E);

максимізувати: Θ;

за умов:

$$\cup T_i = T$$

$$T_i \cap T_j = \emptyset, \text{ для } i \neq j.$$

Матрична модель

Введемо матрицю призначень $X = [x_{ij}]$, де:

$x_{ij} = 1$, якщо завдання t_j призначене вузлу n_i;

$x_{ij} = 0$, інакше.

Тоді навантаження вузла:

$$L_i = \Sigma (x_{ij} \cdot w_j), j = 1..k.$$

Умови:

$\Sigma x_{ij} = 1$, для кожного j (кожне завдання призначене рівно одному вузлу).

Задача оптимізації у вигляді: мінімізувати $(\max_i L_i, P\_fail, E)$ та максимізувати Θ при обмеженнях на матрицю призначень X.

Алгоритми балансування:

- статичні (заздалегідь визначений розподіл);
- динамічні (реагують на зміну навантаження);
- адаптивні (самонавчальні та гібридні, що комбінують кілька методів).

Динамічні алгоритми спираються на постійний моніторинг стану вузлів і можуть використовувати черги для завдань, градієнтні методи або моделі Маркова для оптимізації перерозподілу.

Адаптивні алгоритми дозволяють уникнути простоїв вузлів та підвищують загальну продуктивність системи, використовують інформацію про стан вузлів та інтенсивність навантаження для перерозподілу задач у режимі реального часу. Сучасні дослідження застосовують методи машинного навчання, еволюційні стратегії та моделі на основі теорії черг.

Використання адаптивних алгоритмів маршрутизації

Адаптивні алгоритми маршрутизації відіграють важливу роль в оптимізації групового обміну даними в розподілених обчислювальних системах, де динамічність мережі, зміна топології та нерівномірне навантаження створюють складнощі для традиційних статичних методів. На відміну від фіксованих маршрутів, адаптивні алгоритми дозволяють системі реагувати на поточний стан мережі, обираючи оптимальні шляхи доставки даних до груп отримувачів. Це забезпечує підвищення пропускної здатності, зменшення затримки та стійкість до збоїв. У контексті групового обміну, де дані передаються множині вузлів, адаптивна маршрутизація є ключовим інструментом для ефективного розподілу трафіку та уникнення вузьких місць. Адаптивні алгоритми маршрутизації базуються на здатності системи аналізувати поточний стан мережі та динамічно коригувати маршрути для передачі даних. Основні принципи включають:

1. моніторинг стану мережі - збір інформації про затримки, пропускну здатність, втрати пакетів і доступність вузлів у реальному часі;

2. оцінка альтернативних шляхів - аналіз кількох можливих маршрутів із використанням метрик, таких як найкоротший шлях, мінімальна затримка чи максимальна пропускна здатність;

3. динамічне прийняття рішень - вибір оптимального маршруту на основі поточних умов, з можливістю переключення при зміні стану мережі;

4. стійкість до збоїв - перенаправлення трафіку в разі відмови вузлів чи каналів зв'язку;

5. оптимізація для груп - створення ефективних дерев маршрутизації для доставки даних усім учасникам групи.

Ці принципи дозволяють адаптивним алгоритмам ефективно працювати в умовах динамічних і гетерогенних мереж.

Основні типи адаптивних алгоритмів:

1. алгоритми на основі метрик якості - ці алгоритми обирають маршрути, виходячи з метрик якості обслуговування (QoS), таких як затримка, пропускна здатність чи втрата пакетів, вони особливо ефективні для групового обміну в системах реального часу; механізм роботи - кожен вузол або маршрутизатор періодично обмінюється інформацією про стан мережі (наприклад, через протоколи типу OSPF з розширеннями QoS), на основі

зібраних даних будується дерево маршрутизації, що мінімізує затримку чи максимізує пропускну здатність для групи отримувачів; переваги - висока ефективність у системах із суворими вимогами до затримки (наприклад, потокове відео), гнучкість у виборі метрик залежно від потреб системи, здатність адаптуватися до змін у навантаженні мережі; недоліки - висока обчислювальна складність через необхідність аналізу множини метрик, збільшення накладних витрат на обмін інформацією про стан мережі; приклад застосування - у системі WebRTC для відеоконференцій QoS алгоритми використовуються для вибору маршрутів із мінімальною затримкою для мультикастових потоків;

2. алгоритми на основі штучного інтелекту - такі алгоритми застосовують методи машинного навчання, наприклад, підсилювальне навчання, для прогнозування оптимальних маршрутів на основі історичних даних і поточного стану мережі; механізм роботи - модель машинного навчання тренується на даних про трафік, затримки та збої, щоб передбачати найкращі маршрути, у груповому обміні алгоритм оптимізує дерево доставки, враховуючи динамічний склад груп і гетерогенність вузлів; переваги - висока адаптивність до непередбачуваних змін у мережі, можливість оптимізації для складних сценаріїв із великою кількістю вузлів, зменшення ручного налаштування маршрутів; недоліки - високі вимоги до обчислювальних ресурсів для тренування моделей, потреба в великій кількості даних для ефективного навчання, складність інтерпретації рішень моделі; приклад застосування - у хмарних системах, таких як Google Cloud, алгоритми підсилювального навчання використовуються для маршрутизації мультикастових потоків між дата-центрами.

3. Overlay-маршрутизація - створює віртуальну мережу поверх фізичної, де вузли самостійно визначають маршрути для групового обміну, обходячи обмеження фізичних маршрутизаторів, механізм роботи - вузли формують логічне дерево маршрутизації, обмінюючись інформацією про стан сусідів, дані передаються через логічні зв'язки, що дозволяє адаптуватися до змін без залежності від апаратної підтримки мультикасту; переваги - незалежність від апаратного забезпечення (не потрібна підтримка IP Multicast), гнучкість у динамічних мережах, таких як P2P-системи, легкість реалізації в програмному забезпеченні; недоліки - збільшення затримки через додатковий рівень абстракції, вищі накладні витрати на управління логічною мережею; приклад застосування - у P2P-мережах, таких як BitTorrent, overlay маршрутизація використовується для ефективного розподілу даних між групами вузлів.

4. Gossip-based Routing - алгоритми на основі Gossip-протоколів передають дані через випадковий обмін інформацією між вузлами, що забезпе-

чує адаптивність у децентралізованих системах, механізм роботи - кожен вузол періодично обирає випадкового сусіда та передає йому оновлення, у груповому обміні це дозволяє поширювати дані до всіх учасників без централізованого управління; переваги - висока стійкість до збоїв і змін топології, масштабованість у децентралізованих системах, простота реалізації; недоліки - повільна конвергенція (час доставки до всіх вузлів), неконтрольований порядок доставки; приклад застосування - у блокчейн-мережах, таких як Ethereum, Gossip протоколи використовуються для групового поширення транзакцій.

Реалізація адаптивних алгоритмів маршрутизації вимагає вирішення кількох технічних проблем:

- збір даних про мережу - необхідність швидкого та точного моніторингу стану мережі без значних накладних витрат;
- обчислювальна складність - аналіз множини маршрутів і метрик потребує значних ресурсів, особливо в реальному часі;
- синхронізація - у груповому обміні важливо забезпечити узгодженість маршрутів для всіх отримувачів;
- безпека - адаптивні алгоритми вразливі до атак, таких як підробка метрик чи маршрутів.

Для подолання цих викликів доречно використовувати комбінації методів, наприклад, гібридні алгоритми, що поєднують QoS і overlay-маршрутизацію, або розподілені системи моніторингу для зменшення накладних витрат.

Адаптивні методи балансування

Класичні методи балансування навантаження поділяються на централізовані та децентралізовані.

Централізовані системи покладаються на єдиний контролер, що приймає рішення. Це забезпечує глобальне бачення, але створює вузьке місце.

Децентралізовані підходи покращують масштабованість, знижують ризики відмови, але ускладнюють координацію.

Алгоритми адаптації:

- евристичні методи - використовують прості правила (наприклад, правило міграції завдань при перевищенні порогу завантаження);
- алгоритми машинного навчання - використання нейронних мереж і підкріплювального навчання для прогнозування майбутнього навантаження, у цьому випадку можна розглядати оптимізацію як задачу мінімізації середнього часу відповіді;
- методи кооперативних ігор - узгодження дій вузлів для досягнення спільної оптимальної стратегії, балансування можна розглядати як гру, де виграш кожного вузла визначається ефективністю системи в цілому.

Приклади сучасних алгоритмів:

- ALBL (Adaptive Load Balancing for Web Systems) - ефективний у веб-середовищі [10];
- V_THR (Variable Threshold Algorithm) - алгоритм з динамічним порогом [4];
- DistCache - масштабоване балансування у розподілених сховищах [12];
- методи з підкріплювальним навчанням для хмарних середовищ [2].

Приклади застосування:

- хмарні обчислення - динамічний розподіл віртуальних машин і контейнерів;
- веб-системи - адаптивний розподіл HTTP-запитів між серверами;
- високопродуктивні обчислення (HPC) - балансування навантаження між обчислювальними вузлами в задачах моделювання;
- системи потокової обробки даних - забезпечення стійкості до атак і перевантажень [11].

Балансування навантаження у процесі передачі даних

Балансування навантаження є важливим елементом оптимізації групового обміну даними в розподілених обчислювальних системах, де нерівномірний розподіл трафіку може призводити до перевантаження окремих вузлів або каналів зв'язку, спричиняючи затримки, втрати пакетів і зниження продуктивності. У контексті групового обміну, коли дані передаються від одного джерела до множини отримувачів, балансування навантаження забезпечує рівномірне використання ресурсів мережі, мінімізуючи вузькі місця та підвищуючи ефективність доставки. Цей підхід дозволяє адаптуватися до динамічних умов, таких як зміна кількості учасників групи чи неоднорідність обчислювальних потужностей вузлів.

Балансування навантаження в груповому обміні базується на кількох ключових принципах, спрямованих на оптимізацію передачі даних:

1. рівномірний розподіл трафіку - розподіл даних між доступними вузлами чи каналами для уникнення перевантаження окремих компонентів;

2. моніторинг ресурсів - постійний аналіз стану вузлів (обчислювальна потужність, пам'ять, пропускна здатність) і мережі (затримка, втрати пакетів);

3. динамічна адаптація - перенаправлення трафіку на основі поточного навантаження чи змін у топології мережі;

4. мінімізація затримки - вибір шляхів або вузлів, що забезпечують найшвидшу доставку до всіх учасників групи;

5. стійкість до збоїв - забезпечення безперервної передачі даних у разі відмови окремих вузлів шляхом перерозподілу навантаження.

Ці принципи дозволяють досягти високої продуктивності та надійності в груповому обміні, особливо в системах із великою кількістю учасників.

1. Круговий розподіл: Круговий розподіл є простим методом, який послідовно розподіляє запити чи пакети даних між доступними вузлами в групі.

Механізм роботи: Центральний координатор (наприклад, сервер або брокер повідомлень) надсилає дані кожному вузлу по черзі, незалежно від їхнього поточного навантаження. У груповому обміні це може застосовуватися для розподілу мультикастових потоків між кількома серверами.

Переваги:

- Простота реалізації та низькі обчислювальні витрати.
- Ефективність у системах із приблизно однаковими можливостями вузлів.
- Не потребує складного моніторингу стану мережі.

Недоліки:

- Не враховує гетерогенність вузлів (різну обчислювальну потужність чи пропускну здатність).
- Може призводити до перевантаження окремих вузлів у динамічних умовах.
- Обмежена адаптивність до змін у навантаженні.

Приклад застосування: У системі Apache Kafka круговий розподіл використовується для розподілу повідомлень між брокерами в кластері, якщо не задано спеціальних політик.

2. Ваговий розподіл: Ваговий розподіл враховує різні можливості вузлів, призначаючи кожному вагу на основі його ресурсів (наприклад, пропускної здатності чи обчислювальної потужності).

Механізм роботи: Вузли з вищою вагою отримують більше запитів чи даних. У груповому обміні це дозволяє оптимізувати передачу, направляючи більші потоки до потужніших вузлів, а менші - до слабших.

Переваги:

- Ефективність у гетерогенних системах, де вузли мають різні характеристики.
- Зменшення ймовірності перевантаження слабших вузлів.
- Гнучкість у налаштуванні ваг залежно від потреб системи.

Недоліки:

- Потреба в попередньому визначенні ваг, що може бути складним у динамічних мережах.

- Додаткові накладні витрати на моніторинг і оновлення ваг.

Приклад застосування: У хмарних платформах, таких як Amazon AWS, ваговий розподіл застосовується для розподілу групових потоків між серверами різної потужності в дата-центрах.

3. Динамічне балансування на основі стану: Динамічне балансування використовує реальний стан вузлів і мережі для розподілу навантаження, адаптуючись до змін у реальному часі.

Механізм роботи: Система періодично збирає дані про навантаження (CPU, пам'ять, пропускна здатність) і перенаправляє трафік до менш завантажених вузлів.

У груповому обміні це може включати перебудову дерев маршрутизації для мультикасту.

Переваги:

- Висока адаптивність до динамічних умов (зміна трафіку, відмови вузлів).
- Оптимізація продуктивності в реальному часі.
- Зменшення затримки та втрат пакетів.

Недоліки:

- Високі накладні витрати на моніторинг і аналіз стану мережі.
- Складність реалізації в децентралізованих системах.
- Можливі тимчасові неузгодженості під час перерозподілу.

Приклад застосування: У розподілених базах даних, таких як Cassandra, динамічне балансування використовується для розподілу групових оновлень між вузлами кластера.

4. Географічне балансування: Географічне балансування спрямоване на розподіл навантаження з урахуванням фізичного розташування вузлів, що особливо важливо для глобальних мереж.

Механізм роботи: Дані направляються до вузлів, розташованих найближче до отримувачів, або до регіональних серверів із низькою затримкою. У груповому обміні це може включати вибір регіональних проксі для мультикастових потоків.

Переваги:

- Зменшення затримки завдяки локальній обробці даних.
- Ефективність у глобальних системах із розподіленими групами.
- Підвищення стійкості за рахунок регіонального резервування.

Недоліки:

- Потреба в інфраструктурі для моніторингу географічного розташування.
- Складність у динамічному оновленні маршрутів при зміні груп.

Приклад застосування: У Content Delivery Networks (CDN), таких як Cloudflare, географічне балансування застосовується для доставки групового контенту до найближчих серверів.

Реалізація балансування навантаження в груповому обміні стикається з кількома технічними проблемами:

- Моніторинг у реальному часі: Збір точних даних про стан вузлів і мережі вимагає значних ресурсів і може створювати додатковий трафік.
- Синхронізація: У груповому обміні важливо забезпечити узгодженість розподілу даних між усіма отримувачами, що ускладнює динамічне балансування.
- Гетерогенність: Різні можливості вузлів (IoT-пристрої проти серверів) ускладнюють вибір оптимальної стратегії.
- Масштабованість: У великих системах із тисячами вузлів балансування може стати обчислювально складним завданням.

Для вирішення цих проблем пропонується застосовувати комбіновані підходи, наприклад, поєднання вагового та динамічного балансування, або використання розподілених брокерів для управління трафіком.

Математичне формулювання задачі

Нехай (L_i) - навантаження на (i)-му вузлі, (N) - кількість вузлів у системі, тоді середнє навантаження:

$$\bar{L} = 1/N \sum_{i=1}^{N} L_i$$

Мірою дисбалансу пропонується функція:

$$F = \sum_{i=1}^{N} (L_i - \bar{L})^2$$

Мета алгоритмів балансування - мінімізувати (F), забезпечуючи тим самим рівномірний розподіл.

Математична модель розподілу навантаження

Якщо система складається з N обчислювальних вузлів, кожен з яких має потужність C_i, а L_i – навантаження, що йому призначено. Час виконання задач можна оцінити як:

$$T = \max(L_i / C_i), i \in N.$$

Критерії ефективності включають мінімізацію середнього часу відповіді та балансування коефіцієнта завантаження:

$$\text{LoadBalance} = (\max(L_i) - \min(L_i)) / \text{mean}(L_i).$$

Для адаптивного алгоритму важливо враховувати динамічні зміни L_i та прогнозувати майбутнє навантаження.

Розглянемо експериментальні результати моделювання системи з N вузлів.

Експериментальні дослідження проводились шляхом імітаційного моделювання розподіленої обчислювальної системи з такими параметрами:

- кількість вузлів: N=10, 20, 50;
- тип навантаження: стохастичний потік задач із пуассонівським розподілом;
- інтенсивність надходження задач: λ=5-50 задач/с;
- обчислювальна потужність вузлів: однорідна та гетерогенна (відхилення до 30%);
- метрики оцінювання:
  - коефіцієнт дисбалансу F;
  - середній час виконання задач T_{avg};
  - завантаженість вузлів.

Порівнювались такі алгоритми:

- статичний (Round Robin);
- динамічний (на основі поточного стану);
- адаптивний (з прогнозуванням навантаження).

Основні обмеження експерименту:

- відсутність урахування мережевих затримок нижчого рівня (L_2/L_3);
- спрощена модель відмов (без каскадних ефектів);
- відсутність впливу кіберзагроз;
- обмежена кількість вузлів (до 50).

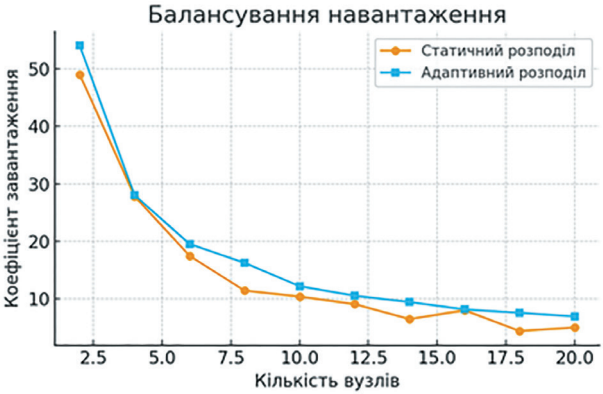

Рис. 1. Коефіцієнт балансування навантаження

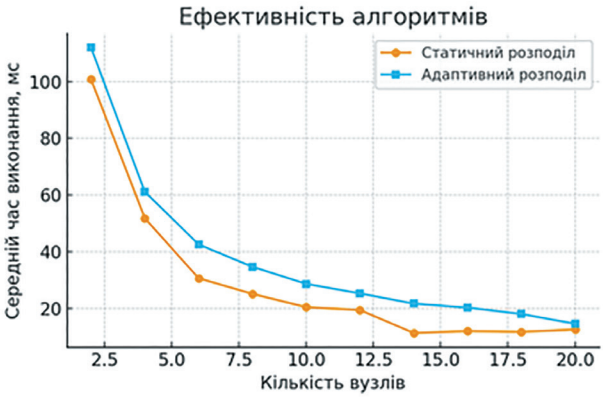

Рис. 2. Середній час виконання задач при різних алгоритмах

Аналіз отриманих результатів (рис. 1, 2) дозволив встановити такі кількісні показники:

- зменшення коефіцієнта дисбалансу: адаптивний алгоритм: на 25-40% порівняно зі статичним;
- скорочення середнього часу виконання задач: на 18-35% у середньому;
- підвищення пропускної здатності системи: до 20% при високому навантаженні (λ>30);
- стабільність роботи: варіація навантаження між вузлами зменшилась з 0.6 до 0.3 (нормалізоване значення);
- ефективність у гетерогенному середовищі: виграш до 45% за рахунок врахування потужності вузлів.

Результати показують, що адаптивні методи забезпечують більш рівномірний розподіл навантаження та зменшення часу виконання у порівнянні зі статичними алгоритмами, особливо при збільшенні кількості вузлів та варіації навантаження.

Виклики та напрями подальших досліджень:

- Зростаюча складність РОС потребує гібридних підходів, що поєднують евристику, прогнозування та самонавчання.
- Балансування з урахуванням енергоефективності набуває все більшого значення.
- Важливим напрямом є захист від атак та несправедливого використання ресурсів.
- Використання адаптивних методів у розподіленому штучному інтелекті (edge AI, federated learning).

Висновки. У роботі вирішено актуальну науково-прикладну задачу підвищення ефективності розподілу навантаження в розподілених обчислювальних системах шляхом застосування адаптивних алгоритмів.

Наукова новизна отриманих результатів полягає в наступному:

- удосконалено модель багатокритеріального розподілу навантаження шляхом інтеграції показників енергоефективності, надійності та динамічних характеристик системи;
- вперше в межах даного підходу кількісно показано вплив адаптивних алгоритмів на зменшення коефіцієнта дисбалансу (до 40%) та часу виконання задач (до 35%);
- набув подальшого розвитку підхід до поєднання моніторингу стану вузлів із прогнозуванням навантаження.

Практичні результати дослідження показали, що:

- адаптивні алгоритми забезпечують істотне покращення продуктивності при змінному навантаженні;
- ефективність системи зростає до 20% за рахунок більш раціонального використання ресурсів;
- найбільший ефект досягається в гетерогенних середовищах.

Напрями подальших досліджень:

- розроблення адаптивних алгоритмів із урахуванням мережевих затримок і топології;
- інтеграція методів глибокого навчання для більш точного прогнозування навантаження;
- дослідження енергоефективних стратегій балансування в edge- та IoT-системах;
- забезпечення кіберстійкості адаптивних алгоритмів (захист від атак на метрики);
- масштабування моделей на системи з тисячами вузлів;
- застосування підходу у федеративному навчанні та розподіленому ШІ.

Таким чином, отримані результати не лише підтверджують ефективність адаптивного балансування, але й формують основу для подальшого розвитку інтелектуальних систем управління ресурсами в розподілених обчислювальних середовищах.

ЛІТЕРАТУРА / REFERENCES

1. Schaerf A., Shoham Y., Tennenholtz M. Adaptive Load Balancing: A Study in Multi-Agent Learning. arXiv preprint cs/9505102, 1995. https://doi.org/10.48550/arXiv.cs/9505102.

2. Chawla K. Reinforcement Learning-Based Adaptive Load Balancing for Dynamic Cloud Environments. arXiv preprint arXiv:2409.04896, 2024. https://doi.org/10.48550/arXiv.2409.04896.

3. Antonis K. A hierarchical adaptive distributed algorithm for load balancing. Journal of Parallel and Distributed Computing, 2004. https://doi.org/10.1016/j.jpdc.2003.07.002.

4. Dasgupta P. V_THR: An Adaptive Load Balancing Algorithm. *Journal of Parallel and Distributed Computing*, 1997. https://doi.org/10.1006/jpdc.1997.1321.

5. Othman O. Optimizing Distributed System Performance via Adaptive Load Balancing. In: Middleware for Communications. *Springer*, 2001.

6. Jiang Y., et al. A Survey of Task Allocation and Load Balancing in Distributed Systems. *Southeast University Technical Report*, 2015.

7. Wang L., et al. Low-Latency, High-Throughput Load Balancing Algorithms. *Journal of Computational Techniques and Applied Mathematics,* 2024. https://doi.org/10.5281/zenodo.12587888.

8. Penmatsa S., Penmatsa K.K.V. Adaptive Fair and Cost-Effective Load Balancing in Heterogeneous Distributed Systems. *EPiC Series in Computing*, 2023.

9. Albalawi N.S., et al. Dynamic scheduling strategies for cloud-based load balancing. J*ournal of Cloud Computing*, 2025.

10. Al-Said A. ALBL: An adaptive load balancing algorithm for distributed web systems. *International Journal of Computer Applications,* 2014. https://doi.org/10.1504/IJCNDS.2014.064041.

11. Daghistani A., et al. Attack-Resilient Adaptive Load Balancing in Distributed Streaming Systems. *IEEE Transactions on Dependable and Secure Computing*, 2021. https://doi.org/10.1109/TDSC.2021.3123071.

12. Liu Z., et al. DistCache: Provable Load Balancing for Large-Scale Storage Systems with Distributed Caching. arXiv preprint arXiv:1901.08200, 2019. https://doi.org/10.48550/arXiv.1901.08200.

13. Kowalski D.R., Olkowski J. Deterministic Fault-Tolerant Local Load Balancing and its Applications against Adaptive Adversaries. arXiv preprint arXiv:2508.01373, 2025. https://doi.org/10.48550/arXiv.2508.01373.

14. Patel A., et al. A Survey Report on Distributed System Using Load Balancing Approach. International Journal of Computer Applications, 2016. https://doi.org/10.9790/0661-180405153158.

15. Zaitsev, I., Golubenko, O., Tkachenko, O., Pidmohylnyi, O., & Antonenko, A. (2023). Exploring advanced hypothesis generation in astronomy through the implementation of a mathematical model of linguistic neural networks. *CEUR Workshop Proceedings*, 3687, 121–128.

16. Kumar S., et al. A Survey on Various Load Balancing Approaches. International Journal of Intelligent Systems and Applications in Engineering, 2023.

17. Verma P., et al. Adaptive Load Balancing in Cloud Computing Environment. International Journal of Intelligent Systems and Applications in Engineering, 2023.

18. Lifflander J.J., et al. Optimizing Distributed Load Balancing for Workloads. Sandia National Laboratories Report, 2021.

19. Ranjan R., et al. A Survey: Load Balancing for Distributed File System. International Journal of Computer Applications, 2015.

20. Grosu D., Chronopoulos A.T., Leung M.Y. Load Balancing in Distributed Systems: An Approach using Cooperative Games. In: Proc. IEEE IPDPS, 2002.

21. Xiong H., Chen G., Li J. Adaptive Load Balancing in Large-Scale Cloud Computing Systems Using Machine Learning. Journal of Cloud Computing, 2022.

22. Nguyen T., et al. Dynamic and Adaptive Load Distribution in Heterogeneous Distributed Systems. IEEE Access, 2023.

23. Singh R., et al. Intelligent Load Balancing in Multi-Cloud Environments: A Review. Future Generation Computer Systems, 2024.

24. Ahmed M., et al. Energy-Aware Adaptive Load Balancing in Distributed Systems. Sustainable Computing: Informatics and Systems, 2021.

25. Zhang Y., et al. Reinforcement Learning for Adaptive Task Scheduling and Load Balancing in Edge-Cloud Systems. IEEE Transactions on Network and Service Management, 2022.