УДК 004.75:004.421.2:004.6 DOI: 10.18372/2073-4751.86.21269

Гавриленко Д. Ю.,
orcid.org/0009-0002-9873-8494,
e-mail: havrylenko.dima@gmail.com

АЛГОРИТМ БАЛАНСУВАННЯ НАВАНТАЖЕННЯ У РОЗПОДІЛЕНИХ СИСТЕМАХ УПРАВЛІННЯ ВЕЛИКИМИ ПОТОКАМИ ДАНИХ У РЕЖИМІ РЕАЛЬНОГО ЧАСУ

Національний університет «Київський авіаційний інститут»

Вступ

У сучасних інформаційних системах, які обробляють великомасштабні потоки подій і виконують функції контролю в реальному часі, передбачувана затримка, стабільність обробки та постійна якість обслуговування є критичними вимогами в умовах сильної варіації швидкості введення. Однак у середовищах з розподіленими обчислювальними ресурсами розподіл навантаження між вузлами за своєю суттю є нерівномірним: окремі вузли можуть відчувати локальне перевантаження, черги завдань зростають, час обробки повідомлень збільшується, а переповнення буфера призводить до втрати даних і погіршення якості обслуговування. Ці ефекти посилюються піковими навантаженнями, швидкими змінами інтенсивності потоку, групуванням подій навколо обмеженого набору «гарячих» кнопок маршрутизації та неоднорідністю апаратних і мережевих характеристик, що призводить до нерівномірної продуктивності між вузлами. За таких обставин статичні стратегії балансування навантаження не можуть забезпечити достатню адаптивність, оскільки вони не відображають поточний стан системи та не компенсують динамічні зміни в базовому розподілі та доступності ресурсів.У той же час чисті динамічні методи, засновані на міграції розділів або реконфігурації маршруту, часто ігнорують накладні витрати, пов'язані з передачею стану, затримкою мережі та затримкою реконфігурації, таким чином порушуючи суворі обмеження реального часу та знижуючи точність операцій керування. Тому існує особлива потреба в адаптивних алгоритмах балансування навантаження, які приймають рішення на основі онлайн-індикаторів продуктивності, зменшують локальні перевантаження, стабілізують затримку та забезпечують рівномірне використання ресурсів, а також контролюють витрати на балансування. Вирішення цієї проблеми є важливим для надійної роботи розподілених систем управління, які обробляють потоки даних великого обсягу в режимі реального часу.

Постановка проблеми та аналіз досліджень

Останні роботи зосереджені на балансуванні навантаження в умовах розподіленої обробки даних і сервісів реального часу. У дослідженні Д. Вовченка та Л. Олещенка [1] узагальнено підходи до балансування вузлів у програмних системах розподіленої обробки великих даних. Автори показують, що вибір алгоритму впливає на продуктивність і стійкість системи за нерівномірних навантажень. Складанний та ін. [2] розглядають паралельну обробку в розширювальних хеш-структурах і оцінюють продуктивність таких рішень. Це важливо для потокових сценаріїв, де розподіл ключів визначає нерівномірність навантаження між партиціями. X. Sun [3] пропонує підхід до динамічного розподіленого планування у потокових обчисленнях. У роботі акцентовано баланс між затримкою виконання та ефективністю завантаження ресурсів. N. Singh et al. [4] досліджують балансування і

сервіс-дискавері в середовищі Docker Swarm для мікросервісних застосунків. Показано, що оркестрація впливає на розподіл запитів і стійкість сервісу під час масштабування. S. Pasham [5] аналізує графові алгоритми оптимізації потоків даних у розподілених хмарних архітектурах. Такий підхід є корисним для задач маршрутизації потоків та зменшення «вузьких місць». C. Wei et al. [6] розглядають алгоритм балансування TCP-довгих з'єднань на основі механізму негативного зворотного зв'язку. Робота демонструє ефективність керування навантаженням за рахунок оперативної реакції на стан вузлів. M. Kashani і E. Mahdipour [7] узагальнюють алгоритми балансування у fog computing. Автори підкреслюють роль гетерогенності та обмежених ресурсів на периферії, що збігається з вимогами потокових систем реального часу. G. Enjam [8] пропонує енергоефективне вирівняння з використанням покращення штучного інтелекту. Це підтримує схильність до адаптивних алгоритмів, які враховують різноманітні критерії. H. Zhang et al. [9] вивчають глибокі знання для оцінки якості даних і виявлення аномалій у великих розділених потоках і такі методи є корисними для захисту балансу від спотворених вимірів та «шумних» даних моніторингу. Такі методи корисні для захисту балансування від викривлених метрик і «шумних» даних моніторингу. D. Sah et al. [10] розглядають схематизацію врівноваження для використання розумних датчиків в IIoT і їхня робота підкреслює важливість оптимального розподілу навантаження для надійної роботи поділеної інфраструктури. У результаті огляду виявлено, що література підтвердила важливість динамічних та адитивних підходів. В той же час залишилась потреба в такому алгоритмі, який би одночасно врахував затримку часу, нерівномірність потоків через різні вузли, а також перерозподіл навантаження в системах управління великими потоками даними.

Мета дослідження

Мета роботи полягає в розробці та обґрунтуванні алгоритму балансування навантаження в розподілених системах які управляють великими потоком даних у режимі реального часу.

Виклад основного матеріалу

Розподілені системи управління великими потоками даних працюють в умовах постійної зміни навантаження. Інтенсивність вхідних потоків є нерівномірною. Частина повідомлень концентрується навколо обмеженого набору ключів. Це призводить до формування «гарячих» партицій. У таких умовах окремі вузли перевантажуються. Черги зростають. Затримка обробки перевищує допустимі межі. Якість сервісу погіршується, що є критичним для режиму реального часу. Для дослідження алгоритму балансування розглянуто модель розподіленої потокової системи з N обчислювальних вузлів. Потоки даних складаються з подій, кожна з яких має ключ маршрутизації. Ключ визначає належність події до певної партиції. Партиції розподіляються між вузлами. На кожному вузлі в реальному часі збираються експлуатаційні метрики. До них входить навантаження процесора, яке визначає величину вхідних повідомлень середню швидкість обробки та часові характеристики затримки.

Запропонований алгоритм засновано на адаптивному переналаштуванні навантаження з використанням відновлювального зв'язку. Для кожного окремого вузла обчислюється цілісний показник навантаження. Це сформульовано на основі нормованих значень завантаження процесора черги та перерви в обробці. Вузли з підвищеним значенням показника класифікуються як обтяжені. Вузли з низьким значенням вважають запасними. Рішення щодо перерозподілу приймається періодично з певним часовим інтервалом. Це дозволяє реагувати на зміни в навантаженні без зайвих витрат.

Алгоритм адаптивного балансування навантаження у розподіленій потоковій системі. Вхідними даними є множина вузлів $\{1..N\}$, потік подій з ключами маршрутизації, набір онлайн-метрик вузлів (CPU utilisation, довжина черги, затримка/час виконання), пороги гістерезису T_{high}, T_{low}, інтервал контролю Δ, максимальна кількість міграцій за цикл M_{max}, а також параметр K – кількість "hot keys", дозволених до міграції за один цикл. На кожному інтервалі Δ для кожного вузла iобчислюються нормовані значення метрик $\hat{cpu}_i$, $\hat{q}_i$, $\hat{lat}_i$ (наприклад, min–max нормалізація у межах кластера за поточний інтервал). Далі формується інтегральний індикатор навантаження L_i як зважена сума нормованих метрик: $L_i = w_1\hat{cpu}_i + w_2\hat{q}_i + w_3\hat{lat}_i$, де $w_1 + w_2 + w_3 = 1$ та ваги задаються політикою сервісу (за вимоги real-time можна підвищити вагу затримки). Вузол класифікується як перевантажений, якщо $L_i > T_{high}$ протягом щонайменше H послідовних інтервалів (гістерезис), і як резервний, якщо $L_i < T_{low}$ протягом щонайменше H інтервалів. Після класифікації виконується перший етап – оперативна маршрутизація: для нових подій обирається вузол призначення з мінімальним L_i серед резервних (або серед усіх, якщо резервних немає), причому для key-based processing зберігається узгодженість партицій (через оновлену мапу "partition→node"). Другий етап – кероване ребалансування – активується лише тоді, коли дисбаланс між вузлами перевищує заданий поріг, наприклад $max\,(L_i) - min\,(L_i) > \delta$ або коли p99 latency на перевантажених вузлах виходить за SLA. Тоді на кожному перевантаженому вузлі визначаються "hot keys" як ключі/партиції з найбільшим внеском у чергу або частоту подій; формується кандидатний список не більше ніж K ключів. Для кожного кандидата оцінюється очікуваний виграш від міграції (зниження L_i, зменшення tail latency) та очікувана вартість (state transfer, network overhead). До виконання допускаються лише ті міграції, для яких виграш перевищує вартість і які не порушують ліміт M_{max}. Обрані ключі/партиції переносяться на резервні вузли з мінімальним L_i із оновленням маршрутизаційної таблиці; після міграцій запускається "cooldown"-період, упродовж якого повторні переміщення цих самих ключів заборонені, що зменшує осциляції та зберігає локальність даних. Алгоритм працює безперервно, підтримуючи баланс між швидкою реакцією на пікові зміни та контролем накладних витрат на ребалансування; ефективність оцінюється показниками throughput, p95/p99 latency, дисперсією завантаження та часткою втрат повідомлень. Блок-схема запропонованого алгоритму подана на рисунку 1.

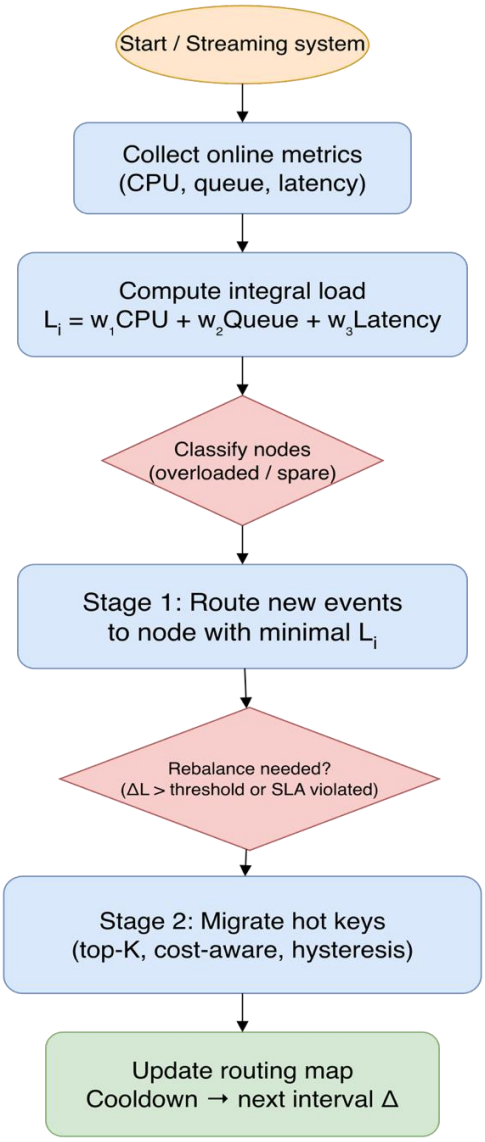

Рис. 1. Блок-схема запропонованого адаптивного алгоритму балансування навантаження для обробки розподіленого потоку даних у реальному часі

Алгоритм працює в двох етапах. На першій стадії виконується оперативна маршрутна організація нових подій. Події

спрямовуються до вузлів з малим індикатором навантаження. На допоміжному етапі здійснюється повторне вирівнювання сегментації. Воно викликає енергію лише в ситуації коли перевищується граничний рівень зміни навантаження між вузлами. Для зменшення витрат лише обмежений перелік «гарячих» ключів, що формують найбільший вклад у перевантаження.

Для запобігання вразливості алгоритму використовують механізм гістерезису. Переміщення його дозволяється тільки у випадку стійкого перевищення порогу протягом різних часових дірок. Крім того зменшується максимальна кількість міграцій за цикл, вони є обмеженими. Це знижує його коливальний ефект і зберігає обробку локальності даних. Оцінку ефективності алгоритму провели шляхом порівняння з основними методами балансування. Використовувалися такі метрики: пропускна здатність системи, квантилі p95, p99, дисперсія завантаження вузлів та частка втрати повідомлень. Результати порівняльного аналізу надані в таблиці 1.

Таблиця 1. Порівняння алгоритмів балансування навантаження

Алгоритм балансування	Throughput, повідомл./с	p95 latency, мс	p99 latency, мс	Дисперсія завантаження	Втрати повідомлень, %
Рівномірний (Round Robin)	12 500	185	260	0,42	2,8
Least Connections	13 200	160	230	0,35	2,1
Зворотний зв’язок	14 100	135	190	0,24	1,4
Запропонований алгоритм	15 300	110	155	0,15	0,7

Отримані результати свідчать що запропонована алгоритмічна програма забезпечує високий пропуск і значне зменшення затримок. Увагу привертає зменшення p99 latency, яке є важливим показником для систем реального часу. Спостерігається зменшення в розподіленні навантаження вузлів, що підтверджує більш рівномірне використання ресурсів. На рисунку 2 наведено динаміку: залежність p95 затримки від часу роботи системи для різних алгоритмів балансування.

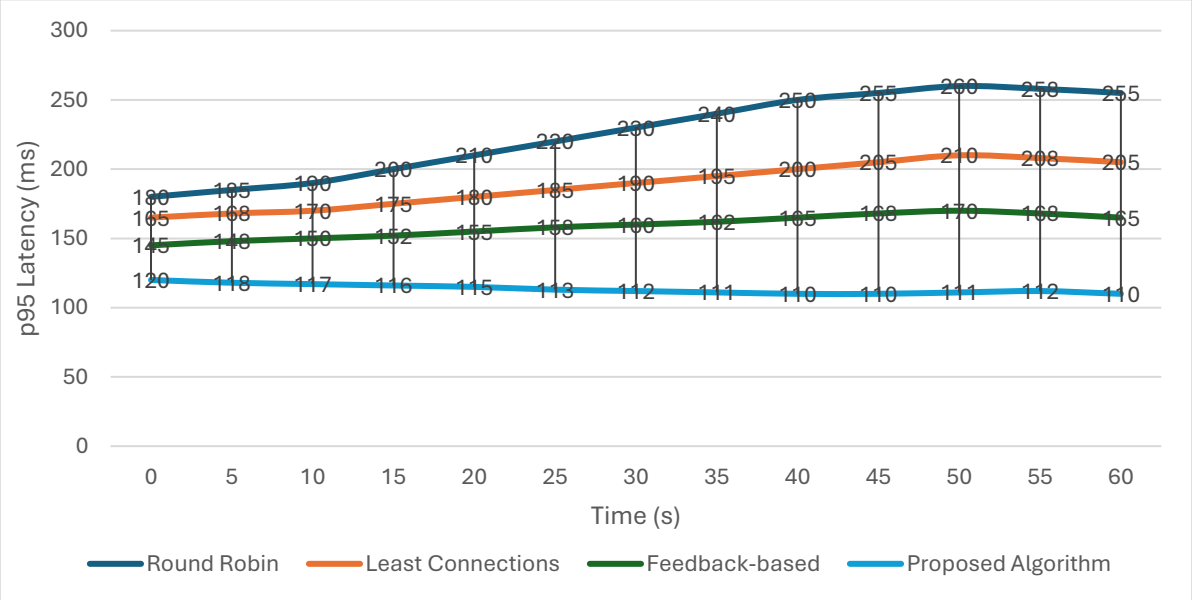

Рис. 2. Динаміка p95 затримки для різних алгоритмів балансування

Часова динаміка затримки p95 показує значні відмінності між алгоритмами балансування. Початкові стратегії характеризуються поступовим нарощуванням затримки в умовах накопичення навантаження та коливаннях витрат. Алгоритм з зворотнім зв'язком показує високу стабільність, однак все ще реагує повільно на різкі зміни в інтенсивності. Запропонований алгоритм забезпечує найнижчі оцінки p95 протягом усього періоду спостереження та швидке відновлення після піків навантаження, що підтверджує його придатність для систем в реальному часі. На рисунку 3 показано зміну середнього завантаження вузлів при зростанні частки «гарячих» ключів у потоці даних.

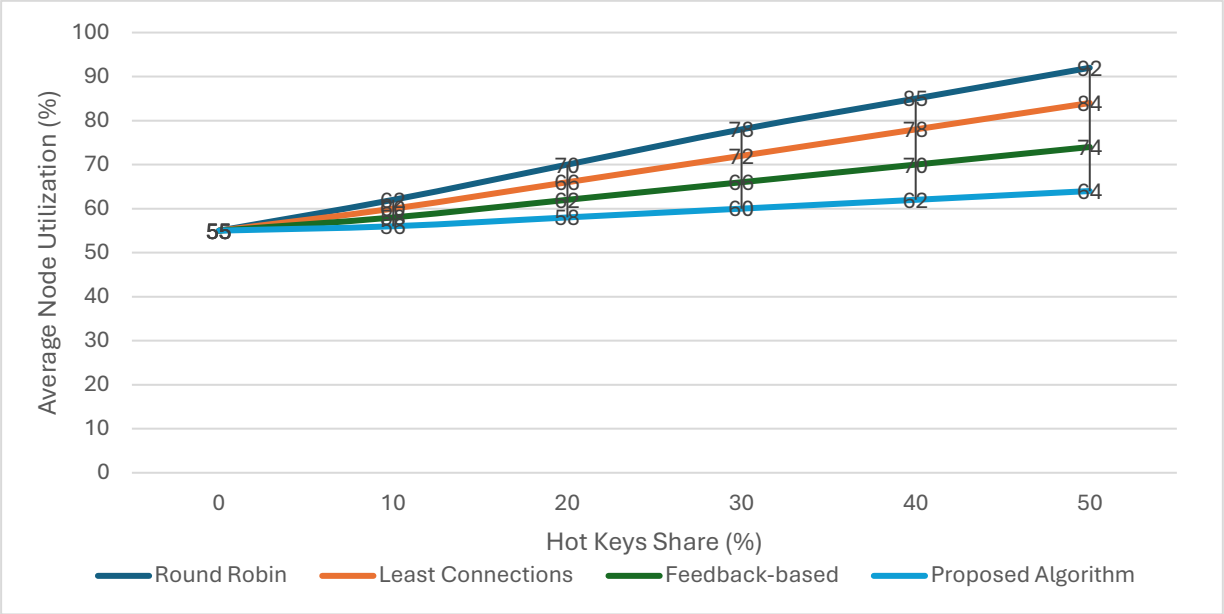

Рис. 3. Залежність середнього завантаження вузлів від частки «гарячих» ключів

Залежність середнього вузла від розподілу «гарячих» ключів свідчить про зростання дисбалансу в налаштуванні при ущільненні потоку на обмеженій кількості ключів. Для простих алгоритмів з підвищенням кількості «гарячих» ключів спостерігається різке зростання навантаження на окремі вузли, що вказує на формування локальних заторів. Алгоритм з реакційним зв'язком зменшує цей наслідок, однак не стирає його повністю. Запропонований алгоритм підтримує наближення до рівноважного навантаження вузлів навіть під час роботи з дуже великим числом «гарячих» ключів, що підтверджує ефективність адаптивного перерозподілу навантаження для реального часу. Таким чином запропонований метод для балансування навантаження заснований на цілісному показнику та регульованому перерозподілу частин забезпечує більш стабільну роботу розподіленої системи в умовах нерівномірних навантажень та «гарячих» ключів. Порівняльні результати підтверджують збільшення пропускної спроможності та зниження критичних квантилів затримки переважно p99, що є важливим для реального часу. Одночасно розподіл навантаження на вузли знижується і частка втрат повідомлень зменшується, що показує ефективніше використання ресурсів та підвищення надійності послуг. Отже, алгоритм підходить для застосувань у системах управління великими потоками даних де потрібні низькі затримки стійкість до пікових навантажень та контроль над витратами на перерозподіл.

Висновки

У дослідженні розглянуто проблему балансування навантаження в

розподілених командних системах з великими обсягами даних у режимі реального часу. Було показано що нерівномірність потоків та скупчення подій на "гарячих ключах" призводять до локалізованого зростання перевантажень, збільшення черг і погіршення затримок, що є складним для часового обмеження. Запропоновано алгоритм адаптивного балансування, який базується на інтегральному показнику навантаження, сформованому з метрик CPU, часу черги та затримки з керованим ребалансуванням частин. Передбачено механізми гистерезису та обмеження міграцій, які зменшують коливання та витрати. Порівняння з простими стратегіями підтвердило збільшення пропускної спроможності і зменшення p95 та p99 затримок, а також зниження коливань навантаження вузлів і частки втрати повідомлень. Отримані висновки свідчать про доцільність використання запропонованого методу в потокових реальних системах для підвищення стабільності сервісу та ефективності використання ресурсів. Перспективами дослідження є зосередження на інтеграції прогнозування навантаження, адаптації порогів ребалансування до контексту завдання та оцінки алгоритму на реальних потокових платформах з різними моделями стану та міграції.

Література

1. *Вовченко Д. С., Олещенко Л. М.* Аналіз алгоритмів балансування вузлів в програмних системах розподіленої обробки великих даних. Вчені записки ТНУ імені В. І. Вернадського. Серія: Технічні науки. 2025. Т. 36 (75), № 2. DOI: https://doi.org/10.32782/2663-5941/2025.2.2/06

2. *Складанний П. М., Костюк Ю. В., Рзаєва С. Л., Мазур Н. П.* Паралельна обробка даних у розширювальних хеш-структурах та оцінка їх продуктивності. Кібербезпека: освіта, наука, техніка. 2025. № 3 (31). С. 243–269. DOI: https://doi.org/10.28925/2663-4023.2025.31.1015

3. *Sun X.* Dynamic distributed scheduling for data stream computing: balancing task delay and load efficiency. Journal of Computer Technology and Software. 2025. Vol. 4, № 1. DOI: https://doi.org/10.5281/zenodo.14785261

4. *Singh N., Hamid Y., Juneja S.* Load balancing and service discovery using Docker Swarm for microservice based big data applications. Journal of Cloud Computing. 2023. Vol. 12. Art. 4. DOI: https://doi.org/10.1186/s13677-022-00358-7

5. *Pasham S. D.* Graph-Based Algorithms for Optimizing Data Flow in Distributed Cloud Architectures. International Journal of Acta Informatica. 2022. Vol. 1, № 1. P. 67–95. URL: https://www.yuktabpublisher.com/index.php/IJAI/article/view/173/129

6. *Wei C., Hou J., Ma D., Zhao J., Sun Y.* Design and implementation of a TCP long connection load balancing algorithm based on negative feedback mechanism. Journal of Physics: Conference Series. 2020. Vol. 1659. Art. 012001. DOI: https://doi.org/10.1088/1742-6596/1659/1/012001

7. *Kashani M. H., Mahdipour E.* Load balancing algorithms in fog computing. IEEE Transactions on Services Computing. 2022. Vol. 16, № 2. P. 1505–1521. DOI: https://doi.org/10.1109/TSC.2022.3174475

8. *Enjam G. R.* Energy-Efficient Load Balancing in Distributed Insurance Systems Using AI-Optimized Switching Techniques. International Journal of Artificial Intelligence, Data Science, and Machine Learning. 2022. Vol. 3, № 4. P. 68–76. DOI: https://doi.org/10.63282/3050-9262.IJAIDSML-V3I4P108

9. *Zhang H., Jia X., Chen C. et al.* Deep learning-based real-time data quality

assessment and anomaly detection for large-scale distributed data streams. International Journal of Medical and All Body Health Research. 2025. Vol. 6, № 1. P. 1–11. DOI: https://doi.org/10.54660/IJMBHR.2025.6.1.01-11

10. *Sah D. K., Nguyen T. N., Cengiz K. et al.* Load-balance scheduling for intelligent sensors deployment in industrial internet of things. Cluster Computing. 2022. Vol. 25. P. 1715–1727. DOI: https://doi.org/10.1007/s10586-021-03316-1

Гавриленко Д. Ю.

АЛГОРИТМ БАЛАНСУВАННЯ НАВАНТАЖЕННЯ У РОЗПОДІЛЕНИХ СИСТЕМАХ УПРАВЛІННЯ ВЕЛИКИМИ ПОТОКАМИ ДАНИХ У РЕЖИМІ РЕАЛЬНОГО ЧАСУ

У статті аналізуються алгоритмічні методи балансування навантаження в розподілених системах управління, які обробляють великі обсяги потоків даних в умовах обмежень реального часу. Зі збільшенням швидкості потоку та складності архітектури операційною метою є досягнення пропускної здатності та обмеженої затримки, одночасно забезпечуючи рівномірне використання різнорідних обчислювальних ресурсів. Попередні дослідження показують, що в середовищах реального часу неправильний розподіл роботи призводить до черг, тривалого часу очікування та затримок у обробці повідомлень. Основними причинами дисбалансу є пікові навантаження, швидкі зміни інтенсивності потоку, концентрація подій навколо «гарячих» кнопок і апаратна неоднорідність у вузлах. Систематизовано найсучасніші стратегії балансування навантаження, включаючи статичні, динамічні та адаптивні алгоритми, а також механізми на основі зворотного зв’язку. Підкреслено обмеження статичних політик у налаштуваннях потокового передавання та встановлено основи алгоритмів, здатних швидко реагувати на зміни стану системи. Запропоновано концептуальний адаптивний підхід до балансування навантаження на основі індикаторів продуктивності онлайн-вузла, включаючи зайнятість черги завдань, навантаження на ЦП і час виконання завдання. Продуктивність стратегій балансування оцінюється за допомогою пропускної здатності, квантилів затримки p95 і p99, рівномірності навантаження на рівні вузла та частки втрачених повідомлень. Результати показують, що адаптивне балансування навантаження покращує надійність розподілених систем шляхом зниження ризику перевантаження та стабілізації затримки черги під час інтенсивної обробки потоку. Подальші дослідження мають бути зосереджені на прогнозуванні робочого навантаження, контекстно-залежній пороговій адаптації та розробці політики перерозподілу, що залежить від середовища, для потокових платформ виробництва.

Ключові слова: *балансування навантаження, розподілені системи, потоки даних, реальний час, адаптивний алгоритм, затримка, пропускна здатність, ребалансування.*

Havrylenko D.

ALGORITHM FOR LOAD BALANCING IN DISTRIBUTED CONTROL SYSTEMS FOR LARGE-SCALE DATA STREAMS IN REAL TIME

The article analyses algorithmic methods of load balancing in distributed control systems that process large volumes of data streams under real-time constraints. As flow rates and architectural complexity increase, the operational goal is to achieve throughput and limited latency while ensuring even utilization of heterogeneous computing resources. Previous studies show that in real-time environments, misallocation of work leads to queues, long wait times, and delays in message processing. The main causes of imbalance are peak loads, rapid changes in flow intensity, concentration of events around "hot" buttons, and hardware heterogeneity in nodes. State-of-the-art load balancing strategies are systematized, including static, dynamic, and adaptive algorithms, as well as feedback-based mechanisms. The limitations of static policies in streaming settings are emphasized, and the foundations of algorithms capable of quickly responding to system state changes are established. A conceptual adaptive approach to load balancing based on online node performance indicators, including task queue occupancy, CPU load, and task execution time, is proposed. The performance of the balancing strategies is evaluated using throughput, p95 and p99 delay quantiles, node-level load uniformity, and the proportion of lost messages. The results show that adaptive load balancing improves the reliability of distributed systems by reducing the risk of congestion and stabilizing queuing delay during heavy flow processing. Further research should focus on workload prediction, context-sensitive threshold adaptation, and environment-sensitive reallocation policy design for streaming manufacturing platforms.

Keywords: *load balancing, distributed systems, data flows, real time, adaptive algorithm, delay, throughput, rebalancing.*

Стаття подана до редакції: 13/02/2026
Стаття прийнята до опублікування: 05/03/2026
Стаття опублікована: 30/05/2026