Issue 18, part 1, 2025

https://doi.org/10.23939/sisn2025.18.035

УДК 004.75

МЕТОД БАЛАНСУВАННЯ НАВАНТАЖЕННЯ СЕРВЕРНИХ ВУЗЛІВ НА ОСНОВІ ДИНАМІЧНОЇ ОЦІНКИ РЕЙТИНГУ

Тарас Мілянець[1], Андрій Пукач[2]

Національний університет "Львівська політехніка",
кафедра автоматизованих систем управління, Львів, Україна
[1] E-mail: Taras.M.Milianets@lpnu.ua, ORCID: 0009-0004-6693-6757
[2] E-mail: Andriy.I.Pukach@lpnu.ua, ORCID: 0009-0001-8563-3311

У статті досліджено проблему ефективного розподілу навантаження в інформаційних системах, що працюють в умовах змінного та непередбачуваного трафіку. Проведено аналіз традиційних статичних алгоритмів балансування, таких як Round Robin і Least Connections, та виявлено їхні обмеження в динамічному середовищі. Запропоновано підхід на основі автоматизованої оцінки рейтингу вузлів, який враховує поточний стан серверів, включаючи навантаження на процесор, оперативну пам'ять та середній час відповіді. Проведено порівняльне моделювання роботи рейтингового методу та методу найменшого завантаження на основі згенерованого набору запитів. Результати демонструють переваги рейтингового підходу в зниженні пікових навантажень і забезпеченні стабільності системи. Метод є масштабованим та придатним для впровадження в сучасних хмарних середовищах.

Ключові слова: балансування навантаження, рейтинг вузлів, адаптивні алгоритми.

Постановка проблеми

У сучасну епоху цифрової трансформації інформаційні системи стикаються з дедалі більшими навантаженнями, викликаними швидким зростанням кількості користувачів, пристроїв і обсягів даних. Ці умови вимагають не лише потужного апаратного забезпечення, але й інтелектуальних механізмів керування ресурсами. Одним із ключових аспектів забезпечення стабільності та продуктивності таких систем є ефективне балансування навантаження між серверними вузлами.

Сучасні сервіси працюють у динамічних умовах, де характер запитів постійно змінюється, а рівень використання ресурсів може суттєво коливатися навіть протягом кількох хвилин. У умовах технологічного розвитку обсяг даних та кількість запитів до серверів і комп'ютерних систем зростає в геометричній прогресії. Для того щоб відповідати високим стандартам надійності та доступності, важливо ефективно розподіляти навантаження між серверами або іншими компонентами. Навантаження на сервери є непередбачуваним, оскільки користувацькі запити можуть значно відрізнятися за кількістю та складністю. Традиційні методи статичного балансування навантаження не забезпечують ефективного розподілу запитів, що може призводити до утворення «вузьких місць» і неефективного використання ресурсів. У цьому контексті виникає потреба в адаптивних підходах, які здатні в режимі реального часу приймати рішення про оптимальний розподіл запитів. Використання лише статичних або спрощених алгоритмів не дозволяє ефективно використовувати обчислювальні потужності, що в свою чергу впливає на швидкодію та надійність системи.

Найбільш поширені підходи до балансування навантаження використовують алгоритми, такі як Round Robin, Least Connections та Weighted Round Robin (Syed, 2024, Sharma, 2023). Ці алгоритми розподіляють запити між серверами, враховуючи обмежену кількість факторів, наприклад, кількість активних з'єднань або простий послідовний розподіл запитів. Хоча такі методи є дієвими для невеликих систем із відносно стабільним навантаженням, вони стикаються з серйозними труднощами в умовах високонавантажених і динамічних систем (Sharma, 2023, Rao, 2023). Найчастіше подібні підходи не здатні швидко адаптуватися до змін у характері запитів, через що система може затримувати відповіді або навіть зазнавати збоїв у разі перевантаження окремих вузлів.

Щоб вирішити ці проблеми, розглядаються динамічні підходи до балансування навантаження, які дозволяють системі адаптуватися до змін у режимі реального часу, забезпечуючи оптимальний розподіл ресурсів. Основна ідея динамічного балансування полягає у безперервному моніторингу показників продуктивності системи, таких як завантаження процесора, використання оперативної пам'яті та пропускна здатність мережі. Це дозволяє системі ефективно розподіляти користувацькі запити між доступними вузлами на основі актуальних даних. Крім того, сучасні динамічні алгоритми можуть використовувати прогнозування навантаження, машинне навчання або адаптивне зважування (Chawla, 2024, Yang, 2023, Parida, 2018), що дає змогу ще точніше розподіляти ресурси. Такі підходи стають критично важливими для хмарних сервісів, великих вебдодатків, де затримки та збої можуть призводити до значних фінансових та репутаційних втрат.

Аналіз останніх досліджень та публікацій

Проблема ефективного розподілу навантаження в інформаційних системах є предметом численних досліджень у галузях обчислювальної техніки, телекомунікацій та хмарних обчислень. Традиційні алгоритми балансування, такі як Round Robin, Least Connections та Weighted Round Robin, широко розглядаються в літературі як базові підходи до розподілу запитів між серверними вузлами. Вони мають низьку обчислювальну складність та легко впроваджуються, однак виявляють обмежену ефективність (Sharma, 2023, Rao, 2023) у випадках динамічно змінюваного навантаження. У роботах дослідників було показано, що для більш гнучкого розподілу ресурсів доцільно враховувати поточний стан кожного вузла, зокрема рівень використання процесора, обсяг доступної пам’яті та затримки відповіді. Це стало поштовхом до створення динамічних алгоритмів балансування навантаження (Dewangan, 2023, Devi, 2016, Parida, 2018), які можуть адаптуватися до змін у середовищі в режимі реального часу. Окремий напрям досліджень пов’язаний із використанням прогнозних моделей та методів машинного навчання для аналізу навантаження і формування рішень щодо розподілу запитів. Такі підходи дозволяють передбачити потенційні «вузькі місця» та мінімізувати ризики перевантаження за рахунок превентивного перенаправлення трафіку. У деяких публікаціях розглядаються моделі з рейтингом вузлів, який формується на основі історичних даних або агрегованих метрик ефективності (Sharma, 2021, Tiwari, 2023). Проте більшість з них не враховують швидкоплинність змін у навантаженні або потребують значних обчислювальних ресурсів для реалізації. Таким чином, аналіз наукових джерел демонструє актуальність створення методів, які поєднують простоту реалізації з адаптивністю до реального навантаження. Саме такий підхід і покладено в основу даного дослідження — через формування автоматизованої системи оцінки вузлів, що дозволяє забезпечити більш точний та гнучкий розподіл запитів у складних динамічних середовищах.

Формулювання цілі статті

Метою статті є розроблення та обґрунтування методу динамічного балансування навантаження серверних вузлів на основі автоматизованої оцінки їх рейтингу, що враховує поточні показники продуктивності, зокрема рівень завантаження процесора, використання оперативної пам’яті, кількість активних з’єднань та середній час відповіді. Запропонований підхід спрямований на під-

вищення ефективності розподілу запитів в умовах динамічного та непередбачуваного трафіку, зниження пікових навантажень та забезпечення стабільності роботи системи. Для досягнення мети передбачено аналіз обмежень традиційних статичних алгоритмів балансування, розроблення математичної моделі рейтингової оцінки вузлів, а також експериментальне порівняння запропонованого методу з класичними підходами в умовах симульованого середовища.

Виклад основного матеріалу

Балансування навантаження (load balancing) — це процес розподілу обчислювальних задач або запитів користувачів між декількома серверними вузлами або ресурсами з метою забезпечення стабільності системи, зменшення часу відповіді та підвищення ефективності використання обчислювальних ресурсів. Найбільш поширеними методами є:

- **Round Robin** – запити рівномірно розподіляються по черзі між усіма вузлами. Простий у реалізації, однак не враховує навантаження.
- **Least Connections** – обирається вузол з найменшою кількістю активних з'єднань. Працює краще в умовах нерівномірного навантаження.
- **Weighted Round Robin** – кожному вузлу присвоюється вага відповідно до його потужності. Запити розподіляються пропорційно до ваг.

Ці методи належать до статичних методів балансування навантаження, але попри свою простоту та легкість у реалізації, мають низку суттєвих недоліків, які обмежують їх ефективність у сучасних інформаційних системах. Насамперед, такі методи не враховують поточний стан обчислювальних ресурсів, зокрема рівень завантаження процесора, обсяг доступної пам'яті чи кількість активних з'єднань на кожному серверному вузлі. Це призводить до ситуацій, коли запити можуть надсилатися на вже перевантажені сервери, що негативно впливає на загальну продуктивність системи. Крім того, статичні підходи не здатні адаптуватися до змін у навантаженні (Yadav, 2023), що робить їх неефективними в динамічних або масштабованих середовищах, таких як хмарні платформи. Ще однією проблемою є те, що ці методи розподіляють запити однаково між усіма вузлами, незалежно від їх обчислювальної потужності, внаслідок чого слабші сервери можуть перевантажуватися, тоді як більш потужні залишаються недозавантаженими.

До того ж, статичні алгоритми не мають механізму зворотного зв'язку і не враховують результати виконання запитів, що унеможливлює корекцію стратегії розподілу в разі змін ситуації. Вони також не здатні передбачати пікові навантаження чи сплески активності користувачів, що значно знижує їх ефективність у реальних умовах експлуатації складних систем. Таким чином, попри свою інженерну простоту, статичні методи не відповідають вимогам високонавантажених або адаптивних інформаційних середовищ.

Саме тому в останні роки спостерігається активний розвиток динамічних методів балансування, які здатні враховувати змінні параметри роботи системи в режимі реального часу та забезпечувати більш точний і стабільний розподіл запитів. Насамперед, такі методи враховують поточний стан серверних вузлів — рівень завантаження процесора, обсяг доступної оперативної пам'яті, мережеву активність, затримки відповіді та інші ключові метрики. Завдяки цьому розподіл запитів відбувається з урахуванням реального стану системи, що дозволяє уникнути перевантаження окремих вузлів і забезпечити більш рівномірне використання ресурсів (Malathi, 2022, Karimi, 2009). Крім того, динамічні підходи адаптуються до змін навантаження в режимі реального часу, що є особливо актуальним для хмарних середовищ, в яких кількість користувачів і обсяг трафіку можуть змінюватися щосекунди. Такі системи здатні миттєво реагувати на збільшення кількості запитів, перенаправляючи трафік на менш завантажені вузли.

У цьому дослідженні розроблено метод балансування навантаження на основі рейтингу кожного вузла, який розраховується на основі показників у реальному часі. Це дає змогу не лише підвищити точність розподілу навантаження, а й забезпечити більшу гнучкість у керуванні

ресурсами, особливо в умовах динамічно змінюваного середовища. Запропонований метод ґрунтується на ідеї оцінки кожного вузла за допомогою агрегованого рейтингу, який розраховується на основі поточних показників стану системних ресурсів. Балансування навантаження здійснюється через вибір вузла з найменшим рейтингом у момент надходження нового запиту.

Як модель формування рейтингу вузлів використано модель зваженої суми з нормалізацією, що дозволяє зменшити вплив різниці в шкалі метрик і забезпечити більш справедливу оцінку. У цій моделі кожному критерію (наприклад, завантаження процесора, використання оперативної пам'яті, середній час відповіді) присвоюється певна вага відповідно до його впливу на продуктивність сервера.

Оцінка рейтингу вузла $Score_i$ розраховується за формулою:

$$Score_i = \omega_R \cdot R_{i,normalized} + \omega_C \cdot C_{i,normalized} + \omega_{CPU} \cdot CPU_{i,normalized} + \omega_{RAM} \cdot RAM_{i,normalized}$$

де:

- $R_{i,normalized}$ – нормалізоване значення середнього часу відповіді
- $C_{i,normalized}$ – нормалізована кількість активних з'єднань
- $CPU_{i,normalized}$ – нормалізований рівень завантаження процесора
- $RAM_{i,normalized}$ – нормалізований рівень використання оперативної пам'яті для вузла *i*
- $\omega_R, \omega_C, \omega_{CPU}, \omega_{RAM} \in [0,1]$ – вагові коефіцієнти, які визначають вплив кожної метрики на загальний рейтинг та мають сумарне обмеження $\omega_R + \omega_C + \omega_{CPU} + \omega_{RAM} = 1$.
- Нормалізація кожної метрики відбувається за формулою:

$$P_{normalized} = \frac{P - P_{min}}{P_{max} - P_{min}}$$

де:

- P – поточне значення метрики для вузла,
- P_{max}, P_{min}– максимальне та мінімальне значення цієї метрики серед усіх вузлів на момент оцінки.

Процес балансування реалізується за допомогою наступного алгоритму:

1. **Отримання метрик вузлів**.
   Отримати поточні значення R_i, C_i, CPU_i , RAM_i для кожного вузла.
2. **Нормалізація значень.**
   Нормалізувати значення з попереднього кроку. Це переводить всі значення в діапазон [0, 1].
3. **Обчислення рейтингу.**
   Для кожного вузла розраховується рейтинг за вищезазначеною формулою. Чим менший $Score_i$, тим менше завантажений вузол.
4. **Вибір оптимального вузла**
   Обрати вузол з мінімальним значенням рейтингу $Score_i$ для обслуговування нового запиту.
5. **Надсилання запиту.**
   Запит надсилається на обраний вузол.
6. **Оновлення стану.**
   Після завершення опрацювання оновлюються показники навантаження для відповідного вузла.
7. **Повторення**.
   Кроки 1–5 виконуються для кожного нового запиту.

Описаний алгоритм можна зобразити наступною блок-схемою

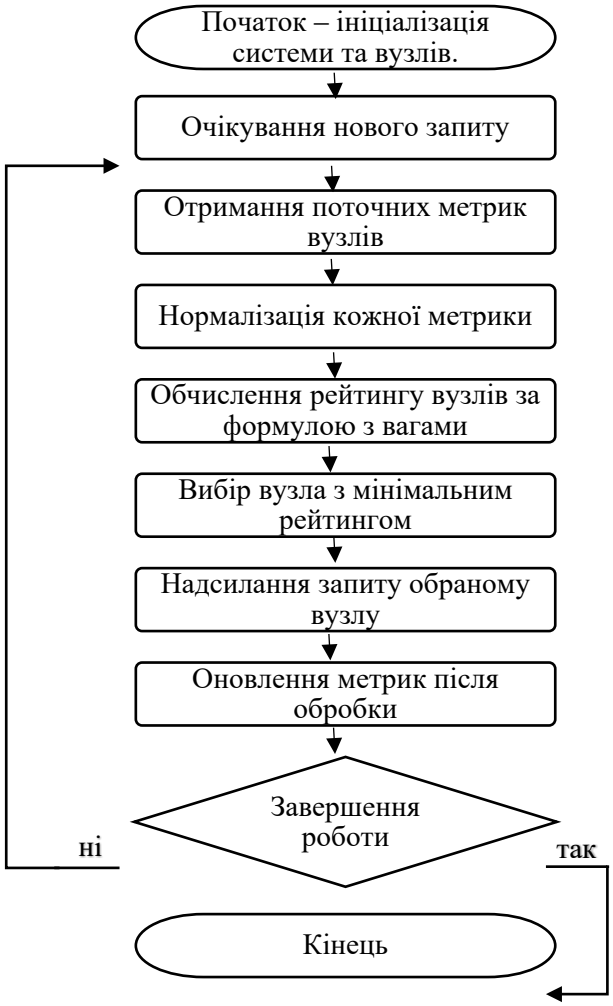

Рис. 1. Блок-схема описаного алгоритму

Для оцінки ефективності запропонованого методу балансування навантаження на основі автоматизованої оцінки рейтингу вузлів було проведено симуляційне тестування з використанням 100 реалістичних запитів та трьох серверних вузлів. Результати було порівняно з класичним методом найменшого завантаження, який є одним з найпоширеніших у реальних інформаційних системах. Для кожного запиту вказувались параметри, що визначають навантаження на CPU, RAM та очікуваний час відповіді, що дозволило реалізувати більш реалістичну симуляцію. Для симуляції використовувався набір запитів у форматі CSV з такими параметрами:

Приклади вхідних запитів для симуляційного тестування

CPU	RAM	ProcessingTimeMs
3.71	1.2	200
0.72	0.55	281
…	…	…

Ці дані були згенеровані для імітації реального трафіку до серверів, із врахуванням навантаження на ресурси (CPU, RAM) та реалістичних часів відповіді, які мали вплив на динамічну оцінку рейтингу кожного вузла.

Симуляційне тестування реалізовано в середовищі Visual Studio з використанням фреймворку .NET та мови програмування C#. Після проведення симуляційного тестування з трьома серверами було отримано наступні результати:

```
--- Least Connections ---
Server 0: Requests=34, TotalTime=5753ms, AvgResponse=169,21ms, MaxCPU=97,62, MaxRAM=38,22
Server 1: Requests=33, TotalTime=6450ms, AvgResponse=195,45ms, MaxCPU=73,89, MaxRAM=30,92
Server 2: Requests=33, TotalTime=5828ms, AvgResponse=176,61ms, MaxCPU=99,60, MaxRAM=36,38
--- Rating-Based Method ---
Server 0: Requests=30, TotalTime=5609ms, AvgResponse=186,97ms, MaxCPU=86,92, MaxRAM=34,70
Server 1: Requests=37, TotalTime=6412ms, AvgResponse=173,30ms, MaxCPU=92,48, MaxRAM=37,26
Server 2: Requests=33, TotalTime=6010ms, AvgResponse=182,12ms, MaxCPU=91,71, MaxRAM=33,56
```

Рис. 2. Результати симуляційного тестування з трьома серверами

Метод найменшої кількості з'єднань розподіляє запити відповідно до поточної кількості активних з'єднань на кожному сервері. Його перевага – простота реалізації та низька обчислювальна складність. У тестуванні цей метод забезпечив відносно рівномірний розподіл по 33–34 запити на кожен сервер. Однак аналіз навантаження виявив низку недоліків. Сервер із найменшим числом з'єднань іноді отримував нові запити, попри те, що був близький до критичних значень CPU або RAM. Зокрема, один із серверів (Server 2) досяг пікового процесорного навантаження у 99,60 %, що потенційно може спричинити деградацію продуктивності або навіть збої у стабільності роботи. Хоча середній час відповіді залишався у межах 169–195 мс, він був нестабільним, із розкидом понад 25 мс між серверами.

У противагу цьому, метод на основі рейтингу вузлів, враховував не лише кількість активних запитів, але й поточний стан кожного вузла, включаючи CPU, RAM та середній час відповіді. Це дало змогу більш точно оцінювати реальну продуктивність кожного сервера та адаптивно перенаправляти запити. Результати показали кращу стабільність розподілу. Попри більшу кількість запитів на одному з вузлів (37 проти 30 на іншому), максимальне навантаження на CPU не перевищувало 92,48 %, що суттєво знижує ризики перевантаження. Середній час відповіді залишився в межах 173–186 мс, при цьому спостерігалась менша варіативність між вузлами, що вказує на більш рівномірне використання ресурсів. Після проведення симуляційного тестування з п'ятьма серверами було отримано наступні результати:

```
--- Least Connections ---
Server 0: Requests=20, TotalTime=3832ms, AvgResponse=191,60ms, MaxCPU=57,43, MaxRAM=20,95
Server 1: Requests=20, TotalTime=4016ms, AvgResponse=200,80ms, MaxCPU=54,62, MaxRAM=23,34
Server 2: Requests=20, TotalTime=3056ms, AvgResponse=152,80ms, MaxCPU=54,69, MaxRAM=19,74
Server 3: Requests=20, TotalTime=3843ms, AvgResponse=192,15ms, MaxCPU=52,51, MaxRAM=18,87
Server 4: Requests=20, TotalTime=3284ms, AvgResponse=164,20ms, MaxCPU=51,86, MaxRAM=22,62
--- Rating-Based Method ---
Server 0: Requests=20, TotalTime=3697ms, AvgResponse=184,85ms, MaxCPU=55,44, MaxRAM=20,81
Server 1: Requests=24, TotalTime=3943ms, AvgResponse=164,29ms, MaxCPU=62,87, MaxRAM=24,30
Server 2: Requests=20, TotalTime=3403ms, AvgResponse=170,15ms, MaxCPU=57,60, MaxRAM=24,05
Server 3: Requests=19, TotalTime=3360ms, AvgResponse=176,84ms, MaxCPU=57,63, MaxRAM=20,15
Server 4: Requests=17, TotalTime=3628ms, AvgResponse=213,41ms, MaxCPU=37,57, MaxRAM=16,21
```

Рис. 3. Результати симуляційного тестування з п'ятьма серверами

Метод найменшої кількості з'єднань забезпечив рівномірний розподіл запитів (по 20 на сервер), проте не враховував стан ресурсів, що призвело до коливань середнього часу відповіді (152,80–200,80 мс) і ризику локальних перевантажень. Натомість метод на основі рейтингу вузлів розподіляв запити нерівномірно (від 17 до 24), але з урахуванням CPU, RAM та часу відповіді, що дозволило досягти більш збалансованого використання ресурсів. Попри трохи вищий піковий рівень CPU (62,87 % проти 57,43 %), він забезпечив стабільнішу роботу системи та зберіг продуктивність на прийнятному рівні (164,29–213,41 мс).

Ще однією важливою перевагою рейтингового методу є його здатність запобігати виникненню "вузьких місць" у системі шляхом реагування на зміни навантаження. Завдяки гнучкому механізму перерахунку рейтингу вузлів у режимі реального часу, метод динамічно пристосовується до змін навколишнього середовища – наприклад, зростання кількості одночасних запитів чи зниження доступної пам'яті на одному з вузлів. Це дозволяє досягти вищої надійності системи та уникати критичних збоїв, що можуть виникати при статичних підходах.

Таким чином, метод автоматизованої оцінки рейтингу вузлів демонструє перевагу над класичними алгоритмами в динамічному середовищі, де навантаження змінюється в реальному часі. Хоча реалізація такого підходу вимагає дещо більших обчислювальних ресурсів, загальні переваги у стабільності, прогнозованості часу відповіді та зниженні ризиків перевантаження роблять його доцільним для використання в сучасних масштабованих інформаційних системах.

Висновки

У цій статті було проведено аналіз сучасних підходів до балансування навантаження в інформаційних системах, а також запропоновано метод, що базується на динамічній оцінці вузлів за допомогою агрегованого рейтингу. Метод поєднує в собі кілька ключових параметрів, зокрема рівень використання CPU, оперативної пам'яті, кількість активних з'єднань та середній час відповіді сервера. Такий підхід дозволяє здійснювати гнучкий та обґрунтований вибір вузла для опрацювання запиту, адаптуючись до змін у навантаженні системи в реальному часі.

Результати експериментального моделювання показали, що метод на основі рейтингу забезпечує більш збалансований розподіл навантаження порівняно з класичним методом найменшого завантаження. Хоча кількість запитів, опрацьованих кожним вузлом, залишалася подібною в обох випадках, рейтинговий підхід продемонстрував значно менші пікові навантаження на процесор і пам'ять. Це вказує на зменшення ризику перевантаження окремих вузлів, що є критично важливим для систем з високою інтенсивністю запитів. Додатковою перевагою запропонованого методу є його масштабованість та універсальність. Він може бути легко адаптований до інших параметрів системи або типів інфраструктур. Таким чином, результати дослідження підтверджують доцільність використання рейтингових механізмів у сучасних системах управління навантаженням, особливо в умовах динамічного та непередбачуваного трафіку.

У подальших дослідженнях доцільно розглянути інтеграцію методів прогнозування навантаження, зокрема із застосуванням машинного навчання, а також тестування моделі в умовах реального середовища.

СПИСОК ЛІТЕРАТУРИ

Bamnele, B., & Bhargava, R. (2023). Review on load balancing in cloud computing using ant colony optimization. *International Journal of Innovation in Engineering Research & Management, 10*(1), 128–133. Retrieved from https://journal.ijierm.co.in/index.php/ijierm/article/view/1274

Chawla, K. (2024). Reinforcement learning-based adaptive load balancing for dynamic cloud environments. *arXiv*. https://doi.org/10.48550/arXiv.2409.04896

Devi, D. C., & Uthariaraj, V. R. (2016). Load balancing in cloud computing environment using improved weighted round robin algorithm for nonpreemptive dependent tasks. *The Scientific World Journal, 2016*, Article 3896065. https://doi.org/10.1155/2016/3896065

Karimi, A., Zarafshan, F., Jantan, A. R., & Saripan, M. I. (2009). A new fuzzy approach for dynamic load balancing algorithm. *arXiv*. https://doi.org/10.48550/arXiv.0910.0317

Malathi, V., & Venkatesh, K. (2022). Energy aware load balancing algorithm for upgraded effectiveness in green cloud computing. In *Proceedings of the International Conference on Sustainable Computing* (pp. 100–110). Retrieved from https://www.researchgate.net/publication/304197438_Efficient_load_Balancing_in_Cloud_Computing_using_Fuzzy_LogicResearchGate

Parida, S., & Panchal, B. (2018). An efficient dynamic load balancing algorithm using machine learning technique in cloud environment. *International Journal of Scientific Research in Science, Engineering and Technology, 4*(4), 1184–1186. Retrieved from https://www.academia.edu/37082445/An_Efficient_Dynamic_Load_Balancing_Algorithm_Using_Machine_Learning_Technique_in_Cloud_EnvironmentAcademia

Rao, D. C., Sharma, S., Nayak, S. K., Srichandan, S. K., & Dash, A. (2023). A novel modified and optimized meta-heuristic load-balancing technique for cloud computing system. *International Journal of Intelligent Systems and Applications in Engineering, 11*(9s), 598–611. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/3209IJISAE

Sharma, A., & Sharma, K. K. (2023). Cloud computing: Hybrid load balancing algorithm proposal. *International Journal of Intelligent Systems and Applications in Engineering, 11*(10s), 859–864. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/3356IJISAE+1IJISAE+1

Sharma, H. C., & Semwal, P. (2021). A review of load balancing algorithms in cloud computing. *International Journal of Creative Research Thoughts, 9*(3), 2786–2790. Retrieved from https://www.researchgate.net/publication/357332048_A_REVIEW_OF_LOAD_BALANCING_ALGORITHMS_IN_CLOUD_COMPUTINGResearchGate

Syed, D., Muhammad, G., & Rizvi, S. (2024). Systematic review: Load balancing in cloud computing by using metaheuristic based dynamic algorithms. *Intelligent Automation & Soft Computing, 39*(3), 437–476. https://doi.org/10.32604/iasc.2024.050681

Tiwari, S., & Bhatt, C. (2023). Performance evaluation on load balancing algorithms in cloud computing environment: A comparative study. *Harbin Gongcheng Daxue Xuebao/Journal of Harbin Engineering University, 44*(5), 50–60. Retrieved from https://harbinengineeringjournal.com/index.php/journal/article/view/195harbinengineeringjournal.com

Yadav, J., & Richariya, P. (2023). Performance evaluation of load balancing algorithms in cloud environment. *International Journal for Research Publication and Seminar, 14*(1), 144–154. Retrieved from https://jrps.shodhsagar.com/index.php/j/article/view/352jrps.shodhsagar.com

Yang, P., Zhang, L., Liu, H., & Li, G. (2023). Reducing idleness in financial cloud services via multi-objective evolutionary reinforcement learning based load balancer. *arXiv*. https://doi.org/10.1007/s11432-023-3895-3ю

A METHOD FOR BALANCING SERVER NODE LOADS BASED ON DYNAMIC RANKING EVALUATION

Taras Milianets[1], Andrii Pukach[2]

Lviv Polytechnic National University,
department of automated control systems, Lviv, Ukraine
[1] E-mail: Taras.M.Milianets@lpnu.ua, ORCID: 0009-0004-6693-6757
[2] E-mail: Andriy.I.Pukach@lpnu.ua, ORCID: 0009-0001-8563-3311

This paper addresses the challenge of achieving effective load distribution in information systems operating under dynamic and unpredictable traffic conditions. The study examines traditional static load balancing algorithms, such as Round Robin and Least Connections, and highlights their limitations in adaptive environments. To overcome these constraints, a node rating–based approach is proposed, which dynamically evaluates server states by considering CPU utilization, memory load, and average response time. A comparative simulation of the proposed method against the Least Connections algorithm was conducted using a representative request dataset. The results show that the rating-based method mitigates peak loads, enhances system stability, and ensures more efficient resource utilization. Furthermore, the approach demonstrates scalability and suitability for deployment in modern cloud infrastructures.

Keywords: load balancing, node rating, adaptive algorithms.