УДК 004.658 **DOI: 10.33042/2522-1809-2024-4-185-2-6**

Р.В. Белоус, Є.В. Крилов

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Україна

МІНІМІЗАЦІЯ МЕРЕЖЕВОГО ТРАФІКУ В RAFT-LIKE CONSENSUS ALGORITHM

Розроблено новий метод мінімізації мережевого трафіку для алгоритму консенсусу типу RAFT. Базуючись на попередньому обміні ключовими векторами та кардинальностями між нодами, він дозволяє зменшити обсяг даних, що передаються, шляхом уникнення дублювання і передачі тільки необхідних даних. Застосування цього методу дає змогу значно підвищити ефективність системи та знизити навантаження на мережу.

Ключові слова: *розподілені бази даних, RAFT, мережевий трафік, кардинальності, Big Data, IoT.*

Постановка проблеми

Сьогодні, в еру Big Data, коли дані збільшуються в геометричній прогресії, у розподілених системах щосекунди обробляються та передаються між вузлами величезні об'єми даних, унаслідок чого виникають питання оптимізації та покращення процесів всієї системи, зокрема процесу передачі мережевого трафіку. Особливо актуальною ця проблема є для систем, які використовують алгоритми консенсусу RAFT [1], що забезпечують узгодженість і надійність даних через постійний обмін інформацією між вузлами. Підвищений обсяг мережевого трафіку може значно знизити продуктивність всієї системи, збільшити час виконання запитів та створити додаткові витрати на інфраструктуру.

Алгоритм RAFT, який широко використовується для досягнення консенсусу в розподілених системах [2], забезпечує узгодженість даних шляхом синхронізації журналів змін між нодами (рис. 1).

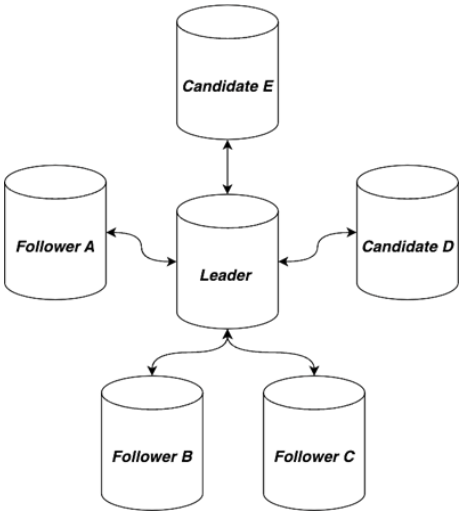

Рис. 1. Схема взаємодії нод в RAFT-like Consensus Algorithms

Однак традиційні методи обміну даними в таких системах можуть призводити до значного навантаження на мережу, особливо у разі частих змін даних або великих обсягів інформації. У цій статті ми пропонуємо новий метод мінімізації мережевого трафіку в розподілених базах даних, що використовують алгоритм консенсусу типу RAFT.

Аналіз останніх досліджень і публікацій

Існує багато методів, що демонструють значний інтерес наукової спільноти до цієї проблематики. У контексті розподілених систем та баз даних ефективне управління мережевим трафіком стає все більш важливим завданням, особливо з огляду на постійне зростання обсягів даних і вимоги до їх швидкої обробки [3]. Оптимізація мережевого трафіку є критично важливою задачею, оскільки дозволяє зменшити затримки, підвищити продуктивність та забезпечити ефективне використання ресурсів. Існує кілька методів, які використовуються для досягнення цієї мети.

Метод Semi-Join був запропонований як ефективний підхід до оптимізації мережевого трафіку при виконанні запитів на об'єднання в розподілених базах даних. Він був введений у 1980-х роках для вирішення проблеми надмірного обсягу даних, які передаються між нодами у розподілених системах [4]. Метод є одним із найвідоміших підходів для зменшення мережевого трафіку при виконанні запитів на об'єднання в розподілених базах даних. Його основна ідея полягає в тому, щоб передавати лише ключі рядків, необхідних для об'єднання, а не всі рядки таблиць. Це дозволяє значно зменшити обсяг переданих даних.

Процес виконання має наступні кроки:

1) нода-ініціатор запиту надсилає запит на отримання ключів до інших нод;
2) інші ноди повертають відповідні ключі;
3) на основі отриманих ключів нода-ініціатор

запиту вибирає відповідні рядки і передає їх для остаточного об'єднання.

До основних переваг можна віднести:

– зниження мережевого трафіку;

– підвищення продуктивності запитів на об'єднання великих таблиць.

До недоліків слід віднести наступне:

– складність реалізації;

– може бути менш ефективним при об'єднанні малих таблиць або при частих змінах даних.

Алгоритм штучної бджолиної колонії (Artificial Bee Colony, ABC) – це метаевристичний алгоритм оптимізації, натхненний поведінкою бджіл у пошуку їжі. Він використовується для вирішення складних оптимізаційних задач, включно із оптимізацією запитів у розподілених базах даних [5]. Алгоритм штучної бджолиної колонії складається з трьох основних типів бджіл: робочих бджіл, спостерігачів та скаутів. Кожен тип бджіл виконує певні ролі у процесі пошуку і оптимізації. Employed Bees (робочі бджоли) відповідають за пошук нових джерел їжі (рішень) навколо відомих джерел і оновлення їхніх позицій. Onlooker Bees (спостерігачі) обирають джерела їжі на основі інформації, наданої робочими бджолами, і досліджують їхні околиці для покращення якості рішень. Scout Bees (скаути) шукають нові джерела їжі, коли старі джерела вже не можуть бути покращені.

Алгоритм штучної бджолиної колонії ефективно застосовується для оптимізації запитів у розподілених базах даних завдяки своїй здатності знаходити оптимальні рішення. Він має наступні кроки:

– ініціалізація запитів – початковий набір планів виконання запитів генерується випадковим чином;

– оптимізація планів – робочі бджоли модифікують плани виконання запитів, спостерігачі вибирають і покращують плани на основі їхньої якості, а скаути генерують нові плани, коли поточні плани вже не можуть бути покращені;

– оцінка ефективності – плани оцінюються за критеріями продуктивності, серед яких час виконання запиту і обсяг мережевого трафіку;

– вибір оптимальних планів – найкращі плани обираються для виконання запитів у розподіленій базі даних [6].

До переваг ABC-алгоритму можна віднести:

– високу ефективність пошуку оптимальних рішень;

– завдяки паралельному пошуку і використанню різних стратегій пошуку він може бути легко адаптований для вирішення різних типів оптимізаційних задач;

– здатність виходити з локальних мінімумів та використання скаутів допомагає уникнути застрявання у локальних мінімумах.

До недоліків можна віднести високу складність налаштування, яка потребує налаштування багатьох параметрів, як-от кількість бджіл і критерії зупинки, а також значний час виконання для досягнення глобального оптимуму, особливо для великих і складних задач.

Оптимізація мережевого трафіку в розподілених базах даних є складним, але критично важливим завданням. Існує декілька методів, кожен з яких має свої переваги та недоліки. Метод Semi-Join ефективний при об'єднанні великих таблиць, але може бути складним у реалізації, алгоритм штучної бджолиної колонії пропонує гнучкість і високу ефективність, але може бути складним в налаштуванні. Вибір конкретного методу залежить від специфіки розподіленої системи і вимог до продуктивності та ефективності [7].

Мета статті

Метою цього дослідження є розробка та аналіз методу KVCE, а також його порівняння із вже наявними методами оптимізації мережевого трафіку. Практичне застосування методу продемонстровано на прикладі системи онлайн-журналу оцінок учнів, розробленої за допомогою Laravel з використанням реляційної бази даних MySQL.

Виклад основного матеріалу

Огляд методів для порівняння використання мережевого трафіку. У розподілених базах даних мережевий трафік є ключовим фактором, що впливає на продуктивність та ефективність системи. На сьогодні існують декілька основних методів мінімізації мережевого трафіку, які застосовуються у провідних розподілених системах, зокрема Google Spanner [8] використовує глобально розподілений консенсус на основі алгоритму Paxos для забезпечення ACID-транзакцій. Він підтримує глобальні транзакції з використанням глобальних часових міток для синхронізації даних. Цей метод дозволяє зменшити мережевий трафік завдяки чіткій синхронізації даних між нодами та оптимізації запитів через попередній обмін метаданими.

У контексті невеликого проєкту онлайн-журналу оцінок учнів було прийнято рішення розробити новий метод мінімізації мережевого трафіку, оскільки вже наявні методи мають низку недоліків, які роблять їх менш ефективними для потреб проєкту. Новий метод обміну ключовими векторами та кардинальностями є більш придатним рішенням завдяки своїм перевагам [9].

Математична модель. Метод KVCE передбачає попередній обмін метаданими між нодами для мінімізації обсягу переданих даних.

Для мінімізації мережевого трафіку між нодами спочатку виконується обмін ключовими векторами та кардинальностями. Обмін ключовими векторами KV:

$$KV_{i\to j} = KV_i \text{ , для всіх } i,j \in \{1,2,\dots,N\}, \quad (1)$$

де N – кількість нод у розподіленій системі;

KV_i – ключовий вектор на ноді i, який представляє унікальні ключі даних, що зберігаються на ноді i.

Нода i передає свій ключовий вектор ноді j. Обмін кардинальностями C:

$$C_{i\to j} = C_i \text{ , для всіх } i,j \in \{1,2,\dots,N\}, \quad (2)$$

де C_i – кардинальність на ноді i, що представляє кількість записів для кожного ключа в наборі даних F_i.

Після обміну метаданими кожна нода може визначити, які дані потрібно передавати, щоб уникнути дублювання і мінімізувати мережевий трафік:

$$D_{i\to j} = F_i(KV_j \cap KV_i), \text{ для всіх } i,j \in \{1,2,\dots,N\}, \quad (3)$$

де $D_{i\to j}$ – розмір даних, які потрібно передати з ноди i на ноду j;

F_i – набір даних, що зберігається на ноді i. Нода i передає ноді j тільки ті дані, ключі яких не присутні в KV_i.

Загальний мережевий трафік T_{ij} між нодами i та j:

$$T_{ij} = \sum_{i=1}^{N}\sum_{j=1}^{N}(|KV_i| + |C_i| + |D_{i\to j}|). \quad (4)$$

Математична модель методу KVCE дозволяє чітко формалізувати процес обміну ключовими векторами та кардинальностями для мінімізації мережевого трафіку в розподілених базах даних, що використовують алгоритми консенсусу RAFT. Ця модель є ефективним інструментом для оптимізації мережевого трафіку та підвищення продуктивності систем з рідкісними змінами даних.

Матеріалізації. Після виконання запиту в розподіленій системі результати запиту зберігаються на ноді, з якої був ініційований запит. Цей процес називається матеріалізацією. Матеріалізація дозволяє забезпечити швидкий доступ до результатів запиту в майбутньому, знижуючи необхідність повторного виконання того ж запиту, у такий спосіб зменшуючи мережевий трафік. Основні етапи процесу матеріалізації включають:

1) ініціація запиту – користувач або система ініціює запит на конкретній ноді;

2) обробка запиту – запит виконується, дані збираються з різних нод у розподіленій системі;

3) збереження результатів – після обробки запиту результати зберігаються (матеріалізуються) на ноді, з якої був ініційований запит.

Після матеріалізації даних на конкретній ноді виникає необхідність забезпечення узгодженості даних у всій розподіленій системі. Узгодженість даних означає, що всі копії даних на різних нодах системи повинні бути однаковими. Це досягається через механізми реплікації та узгодження змін.

Числові результати. Для більш детального аналізу на 5 нодах було проведено дослідження ефективності методу KVCE залежно від кількості запитів з різними обсягами запитів для оцінки мережевого трафіку, часу виконання запитів, часу синхронізації та кількості переданих даних (рис. 2).

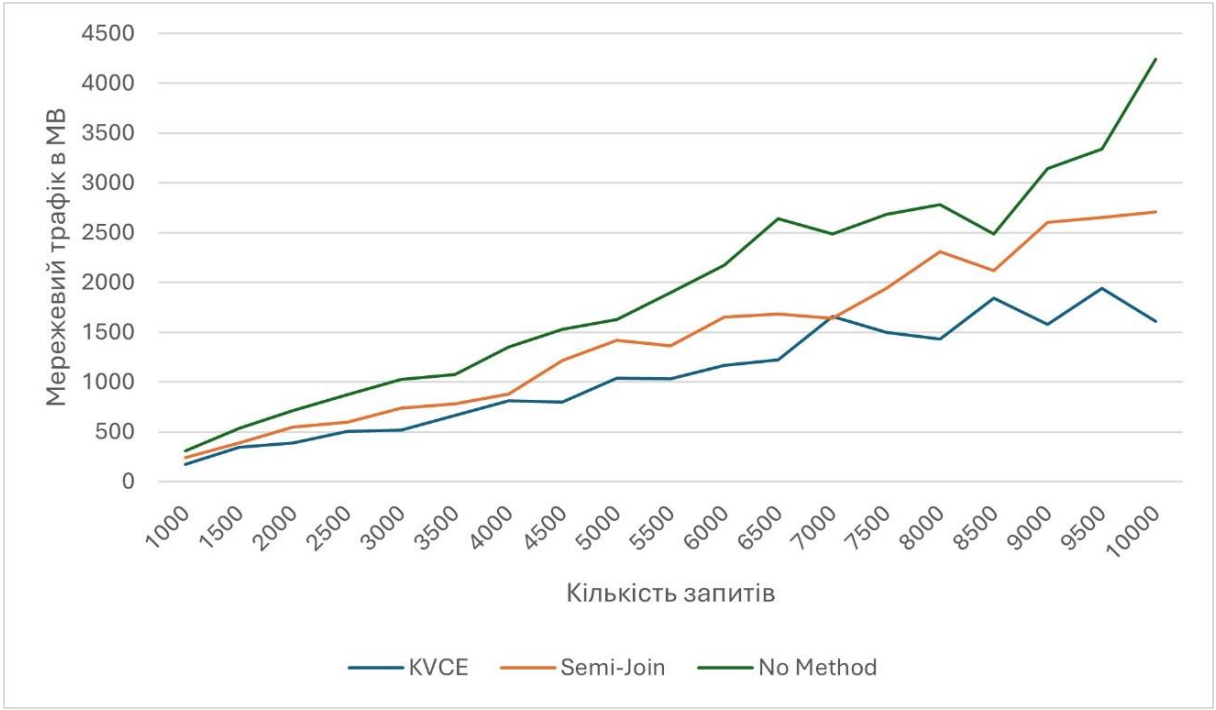

Рис. 2. Порівняння використання мережевого трафіку

Висновки

Дослідження методу обміну ключовими векторами та кардинальностями показало значні переваги у мінімізації мережевого трафіку для розподілених баз даних. Розглянуті методи демонструють значний потенціал для оптимізації мережевого трафіку в розподілених базах даних. Використання методів Semi-Join та ABC може значно підвищити продуктивність і ефективність систем. Проте кожен з цих методів має свої переваги та недоліки, що слід враховувати при виборі конкретного підходу для оптимізації [10].

Метод KVCE особливо ефективний для систем з рідкісними змінами даних, що робить його ідеальним рішенням для нашого проєкту. Недоліками методу є потенційні обмеження при великих обсягах даних або частих змінах, що потребує додаткових досліджень в таких умовах. Загалом метод KVCE забезпечує значні покращення у мінімізації мережевого трафіку та підвищенні продуктивності системи. Простота реалізації та ефективність роблять його оптимальним вибором для невеликих проєктів з розподіленими базами даних, як-от онлайн-журнал оцінок учнів. Результати дослідження підтверджують, що метод KVCE може бути успішно інтегрований у різні розподілені системи, забезпечуючи високі показники продуктивності та ефективності.

Література

1. Ongaro D., Ousterhout J. In Search of an Understandable Consensus Algorithm (Extended Version). Raft Consensus Algorithm. 2014. Available at: https://raft.github.io/raft.pdf

2. C. Mohan, D. Haderle, B. Lindsay, H. Pirahesh, P. Schwarz, "ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging," ACM Transactions on Database Systems (TODS), vol. 17, no. 1, pp. 94-162, 1992. Available at+: https://dl.acm.org/doi/10.1145/1365815.1365816

3. F. Li, V. Venkataraman, S. Rajamanickam, and J. Zhang, "Spanner: Google's globally distributed database," ACM Transactions on Computer Systems (TOCS), vol. 31, no. 2, pp. 8-15, 2013.

4. D. B. Lomet, K. Tzoumas, C. Weng, P. Larson, S. Blanas, C. Curino, and A. J. Elmore, "Cosmos DB: A System Overview," IEEE Data Engineering Bulletin, vol. 42, no. 1, pp. 36-45, 2019.

5. B. Warnke, S. Fischer, S. Groppe, "Using Machine Learning and Routing Protocols for Optimizing Distributed SPARQL Queries in Collaboration," Computers, vol. 12, no. 10, 2023. Available at: https://www.mdpi.com/2073-431X/12/10/210https://www.mdpi.com/2076-3417/14/2/846

6. IDC. Global Data Volume Trends [EB/OL]. 2018. Available online: https://www.seagate.com/files/www-content/our-StoryAmazon/trends/files/idc-seagate-dataage-chine-whitepaper.pdf (accessed on 16 October 2018).

7. Zeng, K.; Yang, J.; Wang, H.; Shao, B.; Wang, Z. A distributed graph engine for web scale RDF data. Proc. VLDB Endow. 2013, 6, 265–276.

8. Rohloff, K.; Schantz, R.E. High-performance, massively scalable distributed systems using the MapReduce software framework: The SHARD triple-store. In Programming Support Innovations for Emerging Distributed Applications; Association for Computing Machinery: New York, NY, USA, 2010; pp. 1–5.

9. Zeng, K.; Yang, J.; Wang, H.; Shao, B.; Wang, Z. A distributed graph engine for web scale RDF data. Proc. VLDB Endow. 2013, 6, 265–276.

10. Rohloff, K.; Schantz, R.E. High-performance, massively scalable distributed systems using the MapReduce software framework: The SHARD triple-store. In Programming Support Innovations for Emerging Distributed Applications; Association for Computing Machinery: New York, NY, USA, 2010; pp. 1–5.

References

1. Ongaro D., Ousterhout J. In Search of an Understandable Consensus Algorithm (Extended Version). Raft Consensus Algorithm. 2014. Available at: https://raft.github.io/raft.pdf

2. C. Mohan, D. Haderle, B. Lindsay, H. Pirahesh, P. Schwarz, "ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging," ACM Transactions on Database Systems (TODS), vol. 17, no. 1, pp. 94-162, 1992. Available at+: https://dl.acm.org/doi/10.1145/1365815.1365816

3. F. Li, V. Venkataraman, S. Rajamanickam, and J. Zhang, "Spanner: Google's globally distributed database," ACM Transactions on Computer Systems (TOCS), vol. 31, no. 2, pp. 8-15, 2013.

4. D. B. Lomet, K. Tzoumas, C. Weng, P. Larson, S. Blanas, C. Curino, and A. J. Elmore, "Cosmos DB: A System Overview," IEEE Data Engineering Bulletin, vol. 42, no. 1, pp. 36-45, 2019.

5. B. Warnke, S. Fischer, S. Groppe, "Using Machine Learning and Routing Protocols for Optimizing Distributed SPARQL Queries in Collaboration," Computers, vol. 12, no. 10, 2023. Available at: https://www.mdpi.com/2073-431X/12/10/210https://www.mdpi.com/2076-3417/14/2/846

6. IDC. Global Data Volume Trends [EB/OL]. 2018. Available online: https://www.seagate.com/files/www-content/our-StoryAmazon/trends/files/idc-seagate-dataage-chine-whitepaper.pdf (accessed on 16 October 2018).

7. Zeng, K.; Yang, J.; Wang, H.; Shao, B.; Wang, Z. A distributed graph engine for web scale RDF data. Proc. VLDB Endow. 2013, 6, 265–276.

8. Rohloff, K.; Schantz, R.E. High-performance, massively scalable distributed systems using the MapReduce software framework: The SHARD triple-store. In Programming Support Innovations for Emerging Distributed Applications; Association for Computing Machinery: New York, NY, USA, 2010; pp. 1–5.

9. Zeng, K.; Yang, J.; Wang, H.; Shao, B.; Wang, Z. A distributed graph engine for web scale RDF data. Proc. VLDB Endow. 2013, 6, 265–276.

10. Rohloff, K.; Schantz, R.E. High-performance, massively scalable distributed systems using the MapReduce software framework: The SHARD triple-store. In Programming Support Innovations for Emerging Distributed Applications; Association for Computing Machinery: New York, NY, USA, 2010; pp. 1–5.

Рецензент: д-р техн. наук, проф. Я.І. Корнага, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Україна.

Автор: БЕЛОУС Роман Володимирович
PhD, асистент кафедри інформаційних систем та технологій
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
E-mail – belous22@ukr.net
ID ORCID: https://orcid.org/0000-0002-7588-941X

Автор: КРИЛОВ Євген Володимирович
кандидат технічних наук, доцент, доцент кафедри інформаційних систем та технологій
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
E-mail – ekrylov1964@gmail.com
ID ORCID: https://orcid.org/0000-0003-4313-938X

MINIMISATION OF NETWORK TRAFFIC IN THE RAFT-LIKE CONSENSUS ALGORITHM

R. Belous, Ye. Krylov
National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, Ukraine

In distributed databases, network traffic is a critical factor that affects system performance and efficiency. The article develops a new method for minimising network traffic in the RAFT-like Consensus Algorithm. The result of using this method is a reduction in network traffic and query execution time in a distributed database. The authors demonstrate its practical application with the example of an online student gradebook system developed with Laravel using a MySQL relational database.

The developed network traffic optimisation method relies on the preliminary exchange of key vectors and cardinalities between nodes. Such an approach reduces the amount of data transferred by avoiding duplication and transmitting only the necessary data. Applying this method increases system efficiency and lowers network load, which is particularly important for distributed databases with high traffic volumes.

The data materialisation process after query execution allows for storing query results on the nodes that initiate these queries. It ensures quick access to already obtained data when performing similar queries in the future, reducing their execution time and improving system performance. Materialisation also helps to reduce the number of repeated data processing, decreasing the system load and enhancing the overall efficiency of the distributed database.

One of the main advantages of this method is its simplicity of implementation and ability to significantly reduce network traffic, particularly in systems containing a small number of infrequent changes. Compared to existing methods, such as the Semi-Join Query Optimisation method, this method shows advantages in systems with small and infrequent changes.

A significant feature of the new method is its ability to provide high data consistency in a distributed system. The use of key vector exchange allows for more efficient data synchronisation between nodes, lowering the likelihood of conflicts and ensuring the relevance of data across the entire system. It is essential for systems requiring high reliability and data accuracy.

Due to its simplicity of implementation and high efficiency, this method is a promising solution for improving the performance of distributed systems in various fields.

Keywords: *distributed databases, RAFT, network traffic, cardinalities, Big Data, IoT.*