УДК 004.65
DOI https://doi.org/10.24144/2616-7700.2023.42(1).188-192

Є. В. Крилов[1], В. А. Нікітін[2]

[1] Київський політехнічний інститут ім. Ігоря Сікорського,
доцент кафедри інформаційних систем та технологій,
кандидат технічних наук
ekrylov1964@gmail.com
ORCID: https://orcid.org/0000-0003-4313-938X

[2] Київський політехнічний інститут ім. Ігоря Сікорського,
аспірант кафедри інформаційних систем та технологій
19valeranikitin96@gmail.com
ORCID: https://orcid.org/0000-0002-4509-1204

ВИКОРИСТАННЯ ТРАНЗАКЦІЙНОГО ГОДИННИКА ДЛЯ ПРИШВИДШЕННЯ ПРОЦЕСУ УЗГОДЖЕННЯ ДАНИХ В РОЗПОДІЛЕНИХ СИСТЕМАХ

Однією з найважливіших властивостей розподіленої системи, які використовують NoSql бази даних, є **узгодженість даних** (consistency). Якщо кількість вузлів розподіленої системи велика, то процес узгодження даних може займати значний час. Для пришвидшення цього процесу в даній статті запропоновано використовувати **транзакційний годинник.** Його особливість полягає в тому, що в процесі виконання запитів до залежних баз даних на різних вузлах розподіленої системи, фіксуються всі транзакції, які в режимі реального часу передаються в транзакційний годинник на головному вузлі. Транзакції обробляються та формується результуюча транзакція, яка розповсюджується на всі залежні бази даних системи з врахуванням пріоритетів. Серед усіх даних розподіленої бази даних визначаються **критичні дані,** для яких швидкість узгодження є найбільш важливим.

Ключові слова: бази даних, розподілені системи, NoSql, узгодженість даних, узгодженність, транзакції, транзакційний годинник.

1. Вступ. Створення розподіленої системи є необхідністю у випадку широкого розповсюдження та великого навантаження. Окрім цього, це дозволяє підтримувати продуктивність сервісу на високому рівні незалежно від географічного розташування, зберігаючи час користувачів та розвантажуючи канали зв'язку. Даний підхід дозволяє відносно легко масштабувати систему додаючи нові вузли, що у свою чергу покращує доступність, оскільки при відмові одного вузла існують інші, які здатні виконувати необхідні функції.

Не дивлячись на суттєві переваги, такі системи можуть мати неузгодженість даних між репліками [1]. Існують різні механізми, які дозволяють покращити це, використовуючи синхронізацію подій у таких системах або за рахунок архітектурних рішень [2]. Одним із таких механізмів є логічні годинники, які дозволяють підтримувати послідовний порядок подій у системі, фіксуючи хронологічні та причинно-наслідкові зв'язки [3]. Яскравими прикладами можуть бути логічний годинник Лемпорта та векторні годинники [4]. Особливістю таких годинників є те, що не потрібно використовувати фізично синхронний глобальний годинник.

2. Постановка задачі. Процес узгодження даних в розподілених системах виконується зазвичай наступним чином. На різних вузлах системи в базах даних відбуваються зміни. В певний час, коли ці зміни треба узгодити виконується

процедура синхронізації даних двох залежних вузлів. Для цього використовуються алгоритми PULL, PUSH та PULL and PUSH. Недоліком цих алгоритмів є те, що вони:

1. Починають працювати в певний визначений час.
2. Обмін інформацією виконується без врахування критично важливих даних.
3. Вони не враховують транзакції, які відбуваються з даними, які зберігаються на різних вузлах залежних баз даних.

Існують задачі, для яких узгодженість даних є критично важливим. Наприклад, платіжна система. Якщо клієнт знімає кошти зі свого рахунку на сервері в Києві, то ця інформація повинна миттєво розповсюджуватися на інші вузли, які знаходяться в Варшаві чи Вашингтоні. З іншого боку, якщо клієнт змінює свою адресу, є можливість трохи затримати узгодженість цих даних. Другий приклад, система оренди номерів в готелях. Процедура бронювання номеру є критично важливою з точки зору узгодженості, а зміна внутрішньої службової інформації готелю не має такої критичності.

Тому задачу **пришвидшення швидкодії процесу узгодження даних** в розподілених системах треба розглядати як завдання **максимально швидкого узгодження критично важливих даних** в залежних базах даних. Для цього необхідно визначити критично важливі дані в системі та розробити алгоритм, який дозволяє максимально швидко їх узгоджувати.

3. Загальний алгоритм узгодження даних в розподілених системах з використанням транзакційного годинника. Загальна схема процесу узгодженості даних з використанням транзакційного годинника показана на рисунку 1.

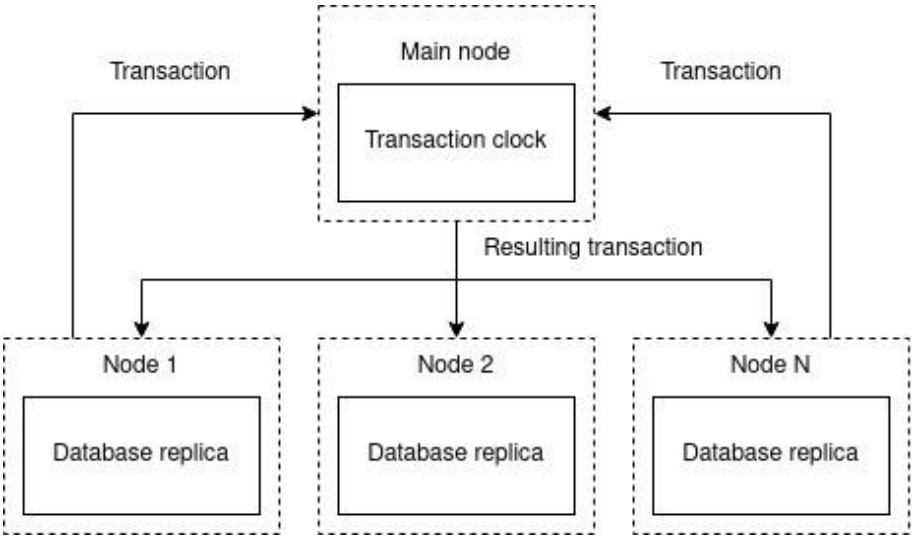

Рис. 1. Загальна схема процесу узгодженості даних з використанням транзакційного годинника.

Узгодження даних між різними вузлами системи виконується за наступним алгоритмом:

1. Всі операції зміни даних (Update, Delete) на залежних базах даних оформлюються в транзакції та миттєво передаються у чергу транзакційного годинника, яка знаходиться на головному вузлі.

2. Всі транзакції, які змінюють репліковані дані, зберігаються у відповідному стеку та обробляються за допомогою транзакційної логіки. Після цього отримуємо результуючу транзакцію, яку передаємо на всі залежні вузли.
3. Для прискорення процесу узгодження **критичних даних** та враховуючи можливий стан гонки, кожен запис бази даних отримує свій пріоритет. В цьому випадку критичні дані завжди будуть мати максимальний пріоритет. Послідовність обробки складних транзакцій в транзакційному годиннику буде виконуватися відповідно до пріоритетів змінених записів.

Слабким місцем даного механізму є головний вузол, на якому знаходиться транзакційний годинник, оскільки при його недоступності втрачається можливість створення результуючої транзакції. У такому випадку, система все-одно залишається працеспроможною на читання, а транзакції залишаються у локальній черзі на кожному вузлі до моменту відновлення головного вузла або переключенням на резервний.

4. Залежні та незалежні бази даних. В складній розподіленій системі можуть бути залежні та незалежні бази даних. Будемо називати бази даних залежними, якщо вони зберігають репліковані дані. Наприклад, дані про банківський рахунок однієї людини чи дані про бронювання номерів готелю, які можна продивлятися та змінювати на різних вузлах розподіленої системи.

Залежні бази даних потребують узгодження, а незалежні — ні. Тому для розмежування залежних та незалежних баз даних введемо поняття матриці залежності **MS,** яку математично можна представити у вигляді (1).

$$MS_{i,j} = \begin{cases} 1, \text{ якщо } i \text{ та } j \text{ БД залежні,} \\ 0, \text{ якщо } i \text{ та } j \text{ БД незалежні,} \end{cases} \tag{1}$$

де i — номер вузла, j — номер запису.

5. Математична модель транзакційного годинника. Математично транзакційний годинник (TS) можна представити у вигляді (2).

$$TS_{i,j} = f(P_{g_n},\, N,\, T_a,\, O,\, SZ,\, NZ), \tag{2}$$

де P_{g_n} — динамічний пріоритет n-транзакції,

N — номер запису,

T_a — вектор абсолютного часу транзакції,

O — операція транзакції,

SZ — вектор старих значень,

NZ — вектор нових значень.

Вектор абсолютного часу транзакції певного вузла можна розрахувати за формулою (3).

$$T_{a_i} = T_{S_i} - T_{O_i}, \tag{3}$$

де T_{S_i} — час транзакції i-го вузла, T_{O_i} — відносна похибка i-го вузла.

Динамічний пріоритет транзакції можна представити формулою (4).

$$P_{g_n} = f(P_{S_n},\, T_{a_n}), \tag{4}$$

де P_{S_n} — статичний пріоритет n-го запису, T_{a_n} — вектор абсолютного часу n-ої транзакції.

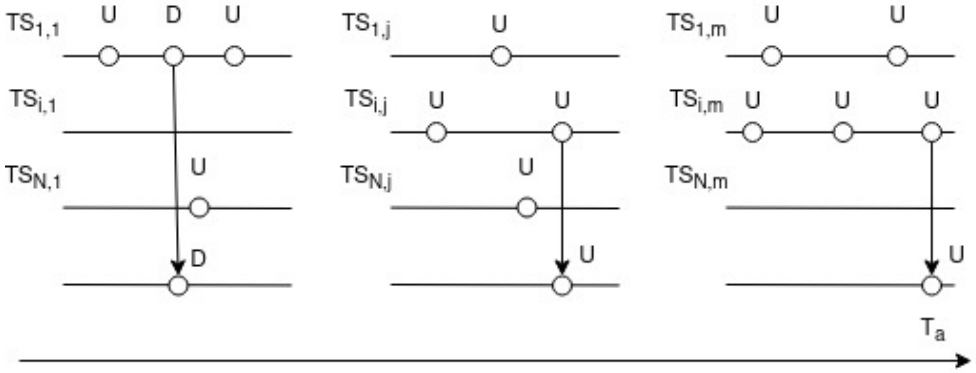

Рис. 2. Приклад роботи транзакційного годинника.

6. Логіка транзакційного годинника. На рисунку 2 можна побачити приклад роботи транзакційного годинника.

На даному рисунку $TS_{i,j}$ — транзакція, яка відбулася на i-му вузлі, j-запису. Під час обробки в транзакційному годиннику їх може буде декілько. Наприклад, для транзакції $TS_{1,1}$ — відбулися послідовно операції Update, Delete, Update. Транзакційна логіка повинна проаналізувати всі складні транзакції різних вузлів для одного запису та підготувати результуючу транзакцію. Результуючу транзакцію розраховуємо відповідно формули (5).

$$\begin{aligned} U_1 &\to U_2 \to \ldots \to U_N = U_N, \\ U_1 &\to D \to \ldots \to U_N = D. \end{aligned} \tag{5}$$

Слід зазначити, що транзакції накопичуються на кожному вузлі по мірі надходження і передаються на головний вузол, якщо значення приоритету транзакцій найбільше. Особливу увагу має операція Delete, яка має більший пріоритет, оскільки видалення відбувається на всіх вузлах системи і через це, її пріоритет більше, ніж у транзакцій з іншими операціями.

7. Висновки. Таким чином, у даній статті запропоновано новий алгоритм узгодження даних у розподілених системах. Він відрізняється від існуючих тим, що процес узгодження даних в залежних базах даних працює безперервно. Цей алгоритм базується на безперервній обробці транзакцій, які надходять від залежних вузлів. Результатом обробки є результуюча транзакція, яка отримується внаслідок роботи транзакційної логіки. Вона розповсюджується до усіх залежних вузлів для узгодження реплікованих даних. Запропонований механізм приорітетів записів залежних баз даних дає можливість змінити послідовність розгляду транзакцій і таким чином забезпечити пришвидшення узгодження критично важливих даних. Визначення конкретного коефіцієнту пришвидшення є подальшим напрямком дослідження.

Список використаної літератури

1. Andrew S. T., van Steen M. Distributed Systems: Principles and Paradigms (2nd Edition). Upper Saddle River, NJ : Pearson Prentice Hall, 2007. 702 p.
2. Nikitin V., Krylov E. A collision-resistant hashing algorithm for maintaining consistency in distributed NoSQL databases. *Adaptive Systems of Automatic Control Interdepartamental scientific and technical collection.* 2022. Vol. 2, No. 41. P. 45–57. DOI: https://doi.org/10.20535/1560-8956.41.2022.271338

3. Raynal M. About logical clocks for distributed systems. *ACM SIGOPS Operating Systems Review*. 1992. Vol. 26, No. 101. P. 41–48. DOI: https://doi.org/10.1145/130704.130708
4. Lamport L. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*. 1978. Vol. 21, No. 7. P. 558–565. DOI: https://doi.org/10.1145/3335772.3335934

Krylov E. V., Nikitin V. A. Usage of transaction clock to speed up the data consistency process in distributed systems.

One of the most important properties of a distributed system that uses NoSql databases is data consistency. If the number of nodes of the distributed system is large, then the process of data consistency can take a significant amount of time. To speed up this process, this article suggests using a transactional clock. This feature is that in the process of executing requests to dependent databases on different nodes of the distributed system, all transactions are recorded, which are transmitted in real time to the transaction clock on the main node. Transactions are processed and the resulting transaction is formed, which is distributed to all dependent databases of the system, taking into account priorities. Among all the data in a distributed database, critical data is identified for which the speed of consistency is most important.

Keywords: databases, distributed systems, NoSql, consistency of data, consistency, transactions, transaction clock.

References

1. Tanenbaum, A. S., & van Steen, M. (2007). *Distributed Systems: Principles and Paradigms (2nd Edition)*. Upper Saddle River, NJ: Pearson Prentice Hall.
2. Nikitin, V., & Krylov, E. (2022). A collision-resistant hashing algorithm for maintaining consistency in distributed NoSQL databases. *Adaptive Systems of Automatic Control Interdepartamental scientific and technical collection,* 2(41), 45–57. https://doi.org/10.20535/1560-8956.41.2022.271338
3. Raynal, M. (1992). About logical clocks for distributed systems. *ACM SIGOPS Operating Systems Review,* 26(101), 41–48. https://doi.org/10.1145/130704.130708
4. Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM,* 21(7), 558–565. https://doi.org/10.1145/3335772

Одержано 30.04.2023