UDC 004.65

VALERII NIKITIN
EVGEN KRYLOV

CONSISTENCY OPTIMIZATION METHODS IN DISTRIBUTED NOSQL DATABASES

Анотація: Розподілена база даних представляє собою об'єднання за допомогою комп'ютерних мереж екземляри баз даних одного чи різних видів. Управління такими системами відбувається прозоро для кінцевих користувачів, що не можно сказати про аварійні ситуації та певні зміни у кількості вузлів. До глобально визначених властивостей відносяться консистенція, доступність та толерантність до розподілу. Вони з'являються внаслідок необхідності горизонтального розширення, що тягне за собою потребу у наявності копій зберігаємих даних. Це обумовлено не тільки питанням продуктивністі, але й питанням доступності. Ці дві властивості є діаметрально різними: технології та способи, які покращують одну з них, автоматично погіршують стан іншої.

Окрім цього, будь-яка існуюча інформаційна система використовує великий набір алгоритмів. Кожен алгоритм є необхідним для вирішення тієї чи іншої задачі. Останні бувають достатньо різноманітними: сортування, структуризація та пошук даних, отримання унікального цифрового відбитку з набору даних. Можливості застосування не обмежені певним напрямком і тільки спонукають дослідників на пошук нових. До цього можно віднести алгоритми хешування, які знайшли широке використання у базах даних, у перевірці на цілісність файлів та мережевих пакетів. Хешування має широке використання і не обмежується використанням тільки для перевірки цілісності, а може бути використаний в якості аналогу для індексації замість збалансованих дерев за рахунок побудови хеш-таблиць [1].

Не дивлячись на велике різноманіття, виникають нові проблеми, які потребують вирішення. З розвитком технологій передачі даних та їх зберіганням, виникає потреба у покращенні підтримки консистентності у розподілених нереляційних базах даних. Існуючі алгоритми хешування є детермінованими та засновані на побітових операціях, які унеможливлюють прогнозування колізій. Таким чином, основною метою розробки нового алгоритму є ідея створення такого алгоритму, який покращить колізійну стійкість при зміні розміру вхідних даних та дозволить оцінити можливу кількість колізій.
КЛЮЧОВІ СЛОВА: розподілені бази даних, розподілені системи, хешування, хеш-функції, узгодженість, стійкість до колізій, узгодженість даних.

Abstract: A distributed database is a combination of copies of databases of one or different types using computer networks. Management of such systems is transparent to end users, that cannot be said about emergency situations and certain changes in the number of nodes. Globally defined properties include consistency, availability, and allocation tolerance. They appear as a result of the need for horizontal extension, which entails the need for copies of the stored data. This is due not only to the issue of productivity, but also to the issue of availability. These two properties are diametrically opposite: technologies and methods which improve one of them, worsen automatically the condition of the other.

In addition, any existing information system uses a large set of algorithms. Each algorithm is necessary to solve some problem. The latter is quite diverse: sorting, structuring and searching for data, obtaining a unique digital fingerprint from a data set. The possibilities of usage are not limited to a certain direction and only encourage researchers to seek for new ones. This includes hashing algorithms, which are widely used in databases, in checking the

integrity of files and network packets. Hashing has a wide range of usage and is not limited to use only for checking integrity, but can be used as an analogue for indexing instead of balanced trees by building hash tables [1].

Despite a great diversity, new problems arise that need to be solved. With the development of data transmission and storage technologies, there is a nicessity to improve consistency support in distributed NoSQL databases. Existing hashing algorithms are deterministic and based on bitwise operations, which make it impossible to predict collisions. Thus, the main goal of developing a new algorithm is the idea of creating such an algorithm that will improve collision resistance when changing the size of input data and which will allow estimating the possible number of collisions.

Keywords: distributed databases, distributed systems, hashing, hash functions, consistency, collision resistance, data consistency.

Introduction. There is a CAP theorem that states that a distributed system can support only two of the three desired properties [2]. These properties include consistency, availability, and partitio tolerance (fig. 1).

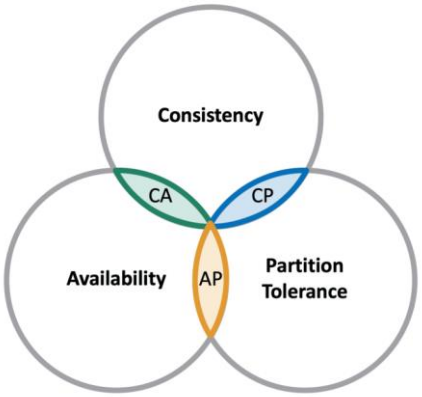

Figure 1. The CAP theorem

A high level of consistency causes clients to read the most updated data regardless of the node. A high level of availability ensures a stable and continuous response to customer requests, regardless of the relevance of data. The ability to partition ensures stable and continuous operation of the system even in the presence of communication problems of nodes or their failure.

It should be noted that the properties are interrelated and the improvement of one property leads to the deterioration of another.

The task of maintaining consistency is of great importance in distributed systems. It is difficult to imagine the normal operation of critical services due to the inconsistency of data on different nodes. Having up-to-date information as soon as possible can save lives and save money that may be lost due to outdated information.

Main part. Several methods are used to reconcile data in distributed systems. One method involves the use of a central node that captures changes and sends updated data to other nodes. This approach is called centralized and resembles a "star" topology (fig. 2).

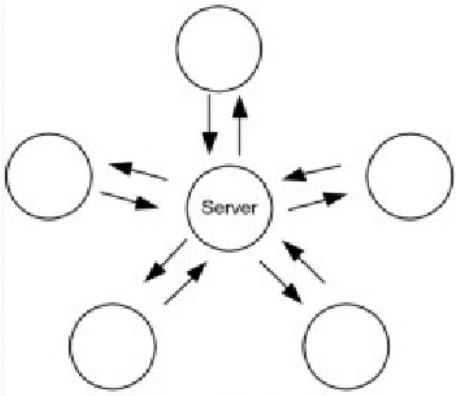

Figure 2. Centrilized approach

This approach allows you to maintain high consistency if clients send requests only to the central node for writing, and others are used as backups and work to read data. Also, it should be noted that this approach provides a high level of availability, since there are always agreed nodes that can replace in case when the central one fails, but there will be a certain delay until a new central node is selected.

Another method is based on the fact that there is no primary node and the nodes coordinate data directly with each other (fig. 3).

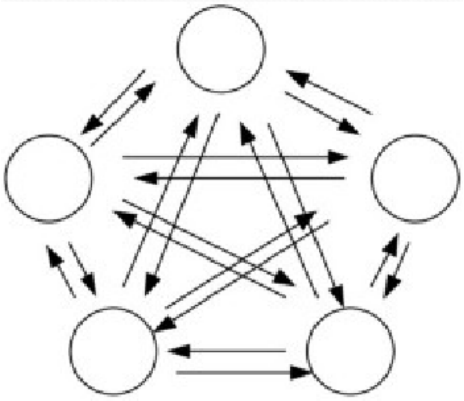

Figure 3. No centralized node approach

When receiving updates, a node notifies everyone else about it. This is a fairly reliable method, but requires an additional mechanism to resolve conflicts when updating the same pieces of data, if they have been updated before. It provides a high level of availability, but can have a negative impact on consistency because communication between multiple nodes may be missing. On the one hand, reconciliation occurs almost instantaneously, and on the other hand, an update may not be available at all on a particular node.

The last method of maintaining consistency is based on the principle of gossip protocol (fig. 4). Each node sends an update to a randomly chosen other one and they propagate it to each other. In this case, the situation is completely opposite to the previous method. There is a possibility that clients will try to retrieve inconsistent data, although there is less risk that a node will not be updated due to a lack of communication between two nodes because it will be notified by a third node [3].

Regardless of how a system reconciles data, it may have additional hashing mechanisms, such as a consistent hashing algorithm to calculate data's node.

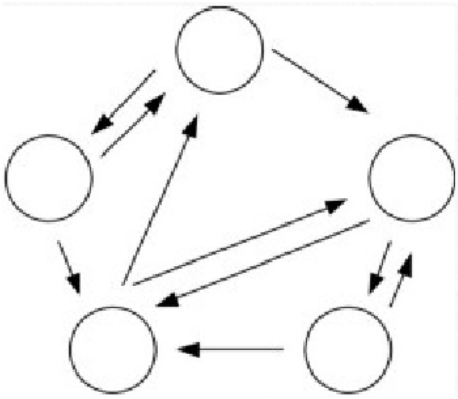

Figure 4. Gossip protocol based approach

In addition to the agreed hashing algorithm, Active Anti Entropy technology can be activated. This mechanism constantly corrects conflicting data in the background. Where Read Repair fixes data as it is read, AAE fixes all data regardless of whether it is actively being accessed by running a background process that constantly checks for inconsistencies. AAE uses a Merkle Tree hash exchange to find these inconsistencies. When a difference is detected at the top of the tree, the database recursively scans the tree until it finds the exact values with the difference, and then sends the least amount of data necessary to restore balance. Riak has such a mechanism and SHA1 is used as a hashing algorithm.

The problem is that the most common hashing algorithms are based on bitwise operations, which in turn can lead to collisions when hashing the data itself in the Merkle tree. To solve this problem, it is possible to use an algorithm that will be aimed at hashing data, and to obtain the resulting hash - an algorithm that is already in use.

In most cases, the most common hashing algorithms are used, such as MD5, SHA1, SHA2. They refer to cryptographic algorithms, in which one of the main criteria is the impossibility of reproducing the input array of data [4].

Conclusions.

Currently, there are many hashing algorithms, but for the most part, their main purpose is to obtain a cryptographic fingerprint. This means that in some sense the question of uniqueness is neglected, which has no place in the question of ensuring consistency in distributed NoSQL databases.

Modern distributed databases require new technologies to maintain consistency in the background. This, in turn, requires the availability of new hashing methods that will have the necessary properties to quickly recognize data inconsistencies.

REFERENCES

1. Combined indexing method in nosql databases / V. Nikitin та ін. // Adaptive Systems of Automatic Control Interdepartmental scientific and technical collection. 2021. № 38 (1). С.3-9 URL: https://doi.org/10.20535/1560-8956.38.2021.232948;

2. Gilbert S., A. Lynch N. Perspectives on the CAP Theorem // Computer. 2012. № 45 (2). P.1-10 URL: https://groups.csail.mit.edu/tds/papers/Gilbert/Brewer2.pdf

3. RRG: redundancy reduced gossip protocol for real-time N-to-N dynamic group communication / V. Wing-Hei Luk та ін. // Journal of Internet Services and Applications. 2013. № 4 (14). С.1-19 URL: https://rdcu.be/cZ1Fk

4. Comparison of hashing methods for supporting of consistency in distributed databases / V. Nikitin та ін. // Adaptive Systems of Automatic Control Interdepartmental scientific and technical collection. 2022. No 1 (40). P.48-53 URL: http://asac.kpi.ua/article/view/261646/258069;