ORES
Научно-практический электронный журнал
Оригинальные исследования (ОРИС), *ISSN 2222-0402*
www.ores.su
original-research.ru

УДК 004.15.415.2.031.43

ОНЛАЙН-РЕКОНФИГУРАЦИЯ В РАСПРЕДЕЛЕННЫХ СИСТЕМАХ NOSQL

Налисник Алексей Николаевич
Калужский филиал федерального государственного бюджетного образовательного учреждения высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
mkumum12@mail.ru

Белов Юрий Сергеевич
Калужский филиал федерального государственного бюджетного образовательного учреждения высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
ysbelov@bmstu.ru

Федоров Виктор Олегович
Калужский филиал федерального государственного бюджетного образовательного учреждения высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
fedorov_vo@bmstu.ru

Аннотация

Исследование фокусируется на сложностях реконфигурации в NoSQL-системах, ключевых для облачных вычислений. Оно описывает вызовы изменения параметров конфигурации баз данных и их воздействие на эффективность. Освещены ключевые аспекты, такие как проблемы глобальных операций реконфигурации и рекомендации по управлению ключами сегментации. Обсуждается проблема баланса нагрузки при перераспределении данных и приводятся два алгоритма Morphus, оптимизирующие сетевую передачу. Анализируются методы улучшения процесса реконфигурации, важные для повышения доступности и производительности систем.

Ключевые слова: NoSQL, реконфигурация, распределенные системы, онлайн-изменения, доступность данных.

ONLINE RECONFIGURATION IN DISTRIBUTED NOSQL SYSTEMS

Aleksei N. Nalisnik
Federal State Budgetary Educational Institution of Higher Education «Bauman Moscow State Technical University» (Kaluga Branch)
mkumum12@mail.ru

Yuri S. Belov
Federal State Budgetary Educational Institution of Higher Education «Bauman Moscow State Technical University» (Kaluga Branch)
ysbelov@bmstu.ru

Victor O. Fedorov
Federal State Budgetary Educational Institution of Higher Education «Bauman Moscow State Technical University» (Kaluga Branch)
fedorov_vo@bmstu.ru

ABSTRACT

The research focuses on the complexities of reconfiguration in NoSQL systems, crucial for cloud computing. It describes the challenges of altering database configuration parameters and their impact on efficiency. Key aspects are highlighted, such as the issues with global reconfiguration operations and recommendations for managing segmentation keys. The problem of load balancing during data redistribution is discussed, accompanied by two Morphus algorithms optimizing network transmission. Methods for improving the reconfiguration process are analyzed, essential for enhancing system availability and performance.

Keywords: NoSQL, reconfiguration, distributed systems, online changes, data availability.

Распределенные системы хранения NoSQL представляют собой одну из основных технологий современной революции облачных вычислений. Эти системы привлекательны тем, что обеспечивают высокую доступность и быстрое чтение и запись данных. Они используются в производственных средах для онлайн-покупок, управления контентом, архивирования, электронной коммерции, образования, финансов, игр, электронной почты и здравоохранения. Даже традиционные базы данных считали реконфигурации серьезной проблемой за последние два десятилетия. Операции реконфигурации связаны с изменениями параметров конфигурации на уровне таблицы базы данных или всей базы данных в целом таким образом, что это влияет на большое количество данных сразу. Примеры включают изменения схемы, такие как изменение сегмента/первичного ключа, который используется для разделения таблицы на блоки, или изменение самого размера блока. В современных сегментированных развертываниях NoSQL глобальные операции реконфигурации весьма неэффективны. Это связано с тем, что их выполнение зависит от специальных механизмов, а не от решения основных алгоритмических проблем и проблем проектирования системы.

Целью данного исследования является рассмотрение и преодоление проблем, связанных с операциями реконфигурации в распределенных NoSQL-системах. Исследование нацелено на изучение и анализ методов и механизмов для улучшения процесса изменения конфигурации данных в таких системах, обеспечивая при этом минимальное влияние на доступность и производительность.

Наиболее распространенное решение включает в себя сначала сохранение таблицы или всей базы данных, а затем повторный импорт всех данных в новую конфигурацию. Такой подход приводит к значительному периоду недоступности. Вторым вариантом может быть создание нового кластера серверов с новой конфигурацией базы данных, а

затем заполнение его данными из старого кластера. Этот подход не поддерживает одновременное чтение и запись во время миграции. Рассмотрим администратора, который хочет изменить ключ сегмента в сегментированном хранилище NoSQL, таком как MongoDB, например. Ключ шардирования используется для разделения базы данных на блоки, где каждый блок хранит значения для непрерывного диапазона ключей шардирования. Ответы на запросы выполняются намного быстрее, если они включают ключ шардирования в качестве параметра запроса (в противном случае запрос должен быть многоадресным). Современные системы настоятельно рекомендуют, чтобы администратор определял ключ сегмента во время создания базы данных, но не менял его впоследствии. Однако это сложно, потому что трудно предугадать, как будет развиваться рабочая нагрузка в будущем. В результате многие администраторы начинают свои базы данных с UID, сгенерированного системой, в качестве ключа сегмента, в то время как другие ограничивают свои ставки, вставляя суррогатный ключ в каждую запись, чтобы его можно было позже использовать в качестве нового первичного ключа. Первый подход с UID снижает полезность первичного ключа для пользователей-людей и ограничивает выразительность запроса, в то время как второй подход будет работать только с хорошим предположением для суррогатного ключа, который сохраняется в течение многих лет работы [1]. В любом случае, по мере того как модели рабочей нагрузки становятся более ясными в течение продолжительного периода работы, новый ключ сегмента для конкретного приложения (например, имя пользователя, URL-адрес блога и т. д.) может стать более подходящим, чем изначально выбранный ключ сегмента. Базы данных могут снижать производительность, а также приводить к сбоям в работе. В результате проблема реконфигурации является предметом жарких дискуссий в сообществе на протяжении многих лет.

Система автоматизации Morphus, например, позволяет вносить изменения в конфигурацию в режиме онлайн, то есть путем одновременной поддержки чтения и записи в таблице базы данных во время перенастройки ее данных. Morphus предполагает, что система NoSQL имеет функции репликации ведущий-ведомый, сегментирования на основе диапазона и гибкость в назначении данных. Этим предположениям удовлетворяют несколько баз данных, например, MongoDB, RethinkDB, CouchDB и т. д. Morphus решает три основные задачи: быстрота, миграция данных между серверами должна сопровождаться наименьшим объемом трафика; ухудшение задержек чтения и записи при реконфигурации должно быть небольшим по сравнению с задержками операций, когда реконфигурации нет; трафик миграции данных должен адаптироваться к топологии сети центра обработки данных [2].

Операция реконфигурации влечет за собой перераспределение данных, присутствующих на нескольких серверах. Информацию необходимо размещать на серверах таким образом, чтобы уменьшить общий объем сетевой передачи при реконфигурации и добиться баланса нагрузки [3]. Morphus использует два алгоритма размещения новых фрагментов в кластере. Первый алгоритм жадный и оптимален по общему объему сетевой передачи. Однако это может создать узкие места из-за кластеризации множества новых фрагментов на нескольких серверах. Второй алгоритм, основанный на двудольном сопоставлении, является оптимальным по объему сетевой передачи среди всех тех стратегий, которые обеспечивают балансировку нагрузки.

Жадный подход рассматривает каждый новый фрагмент независимо. Для каждого нового фрагмента NCi этот подход оценивает все N серверов. Для каждого сервера Sj вычисляется количество элементов данных WNCi, Sj фрагмента NCi, которые уже присутствуют в старых фрагментах на сервере Sj. Затем этот подход выделяет каждый новый фрагмент NCi тому серверу Sj, который имеет максимальное значение WNCi, Sj.

Поскольку фрагменты рассматриваются независимо, алгоритм выдает один и тот же результат независимо от порядка, в котором он рассматривает фрагменты. Вычисление значений WNCi, Sj может выполняться параллельно на каждом сервере Sj после того, как серверы будут осведомлены о новых диапазонах фрагментов. Централизованный сервер собирает все значения WNCi, Sj, запускает жадный алгоритм и информирует серверы о решениях о распределении. Жадный алгоритм оптимален по общему объему сетевой передачи [4].

Фрагменты балансировки нагрузки между серверами важны по нескольким причинам: они уменьшают задержки чтения/записи для клиентов за счет распределения данных и запросов по большему количеству серверов; уменьшает количество узких мест при чтении/записи; сокращает хвост времени повторной настройки, предотвращая выделение слишком большого количества фрагментов на какой-либо один сервер. Вторая стратегия обеспечивает балансировку нагрузки за счет ограничения количества новых фрагментов, выделяемых каждому серверу. Несмотря на то, что общие затраты такие же, как и при жадном подходе, он дополнительно обеспечивает преимущество новой конфигурации с балансировкой нагрузки, когда каждому серверу выделяется ровно один новый фрагмент. Венгерский подход к этому новому графику уравновешивал бы нагрузку по трафику, в то же время жертвуя оптимальностью [5].

Данное исследование подчеркивает критическую роль реконфигурации в распределенных NoSQL-системах и сложности, связанные с изменением конфигурации без остановки работы или серьезного снижения производительности. Проблемы, такие как изменение ключей сегментации и необходимость в реконфигурации на ходу, являются вызовами для администраторов баз данных в контексте эволюции рабочих нагрузок. Исследование показывает перспективное решение для сложной проблемы онлайн-реконфигурации в распределенных NoSQL-системах, обеспечивая баланс между производительностью, доступностью и эффективностью изменения конфигурации без значительного простоя или потери данных.

Список литературы:

1. D. Preuveneers, W. Joosen, «Automated Configuration of NoSQL Performance and Scalability Tactics for Data-Intensive Applications», in Proceedings of the 1st Workshop on Performance Analysis of Big Data Systems, Austin, TX, USA, 8 August 2020; pp. 5–10.
2. Mainak G., Wenting W., Gopalakrishna H. Morphus: Supporting Online Reconfigurations in Sharded NoSQL Systems // IEEE 12th International Conference on Autonomic Computing. 2021, pp. 1–10.
3. Das S., Nishimura S., Agrawal D. Albatross: lightweight elasticity in shared storage databases for the cloud using live data migration // Very Large Database Endowment. 2021, pp 494–505.
4. Налисник А.Н., Белов Ю.С. Автоматическая настройка конфигураций NoSQL баз данных для максимизации производительности и масштабируемости в приложениях с интенсивным использованием данных // E-Scio [Электронный ресурс]: Электронное периодическое издание «E-Scio.ru». 2023 г. – №1.
5. A. Mahgoub, P. Wood, A. Medoff, S. Mitra, F. Meyer, S. Chaterji, and S. Bagchi, “SOPHIA: Online reconfiguration of Clustered NoSQL Databases for Time-Varying Workloads,” in 2019 USENIX Annual Technical Conference ATC’19, Usenix, 2019, pp. 223–240.