Столыпинский вестник

Научная статья
Original article
УДК 004.056.2

ПРОБЛЕМА ЦЕЛОСТНОСТИ ДАННЫХ В МИКРОСЕРВИСНОЙ АРХИТЕКТУРЕ

THE PROBLEM OF DATA INTEGRITY IN MICROSERVICE ARCHITECTURE

Кучеренко Н. Ю., бакалавр, факультета информационных технологий и компьютерной безопасности, Воронежский государственный технический университет, Россия, г. Воронеж

Kucherenko N. Y., Bachelor, of Information Technology and Computer Security Faculty, Voronezh State Technical University, Russia, Voronezh

Аннотация: в статье будет проведено исследование задачи сохранения целостности данных в микросервисной архитектуре. Целостность данных, отвечающая за надежность и сохранность информации, является обязательным требованием в любой системе. С расширением систем на микросервисной архитектуре задача обеспечения согласованности данных при распределенных транзакциях становится нетривиальной задачей, так как готовых решений данной задачи в распределенных системах обычно не существует. Статья посвящена анализу существующих методов обеспечения целостности данных. Разбираются преимущества и недостатки протокола двухфазных фиксаций и паттерна «Сага» при задаче сохранения целостности данных в рамках набора

требований ACID, рассматривается методика устранения недостатков предложенного подхода.

Annotation: the article will investigate the problem of preserving data integrity in a microservice architecture. Data integrity, which is responsible for the reliability and safety of information, is a mandatory requirement in any system. With the expansion of systems on a microservice architecture, the task of ensuring data consistency in distributed transactions becomes a non-trivial task, since there are usually no ready-made solutions to this problem in distributed systems. The article is devoted to the analysis of existing methods of ensuring data integrity. The advantages and disadvantages of the two-phase commit protocol and the "Saga" pattern are analyzed for the task of preserving data integrity within the set of ACID requirements, and the method of eliminating the disadvantages of the proposed approach is considered.

Ключевые слова: целостность данных, микросервисная архитектура, распределенные транзакции, паттерн сага, двухфазные фиксации
Keywords: data integrity, microservice architecture, distributed transactions, saga pattern, two-phase fixations

Введение

Задача обеспечения целостности данных в монолитной системе обычно не вызывает особых сложностей, так как зачастую она решается механизмами, встроенными в базу данных. Но при переходе на микросервисную архитектуру появляются глобальные распределенные транзакции, контроль целостности данных в которых уже становится не самой простой задачей. При выходе из строя одной из частей распределенной транзакции возникают ситуации, когда необходимо откатить примененные изменения. Необходимо выбрать правильный подход, который менее всего будет влиять на производительность системы и удовлетворять требованиям целостности данных.

Принципы ACID

Когда речь идет об организации целостности данных, целесообразно привезти основополагающий набор принципов, которым должна отвечать транзакция – требования ACID.

Атомарность гарантирует, что транзакция может быть только в двух состояниях: выполнена полностью или не выполнена вообще.

Консистентность – согласованность данных друг с другом, результат выполнения транзакций должен являться суммарным эффектом. Наиболее сложный для реализации принцип, потому что при применении микросервисного паттерна database per service его выполнение лежит целиком на разработчиках конкретного сервиса.

Изолированность означает выполнение параллельных транзакций независимо друг от друга.

Долговечность означает, что изменения, полученные при успешном выполнении транзакции, хранятся вечно и не будут отменены ни при каких обстоятельствах.

Пока существует один сервис, задача обеспечения целостности данных и соответствия принципам ACID в транзакциях успешно выполняется базой данных. При транзакциях между несколькими сервисами уже нет какого-то готового решения этой проблемы. Необходимо реализовать механизм, отвечающий за целостность данных.

Протокол двухфазных фиксаций (2PC)

Одним из способов обеспечения целостности данных в распределенной микросервисной архитектуре является протокол двухфазной фиксации (2PC), который по сути и используется в транзакциях баз данных.

Принцип работы 2PC достаточно прост. Основным компонентом двухфазного коммита является координатор. Координатор регулирует ход транзакции, принимает решения о коммите или отмене транзакции. Все участники процесса получают оповещения о решении кординатора (рисунок 1).

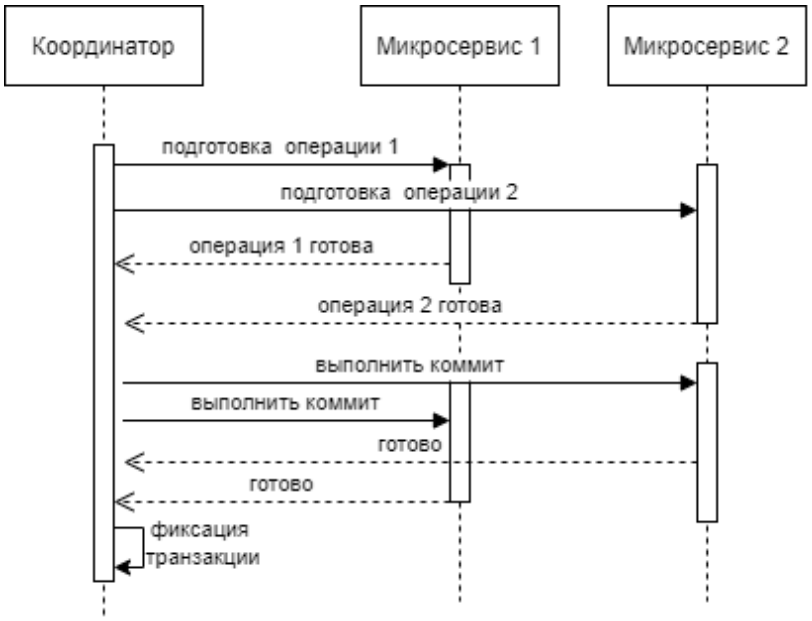

Рисунок 1. Алгоритм работы двухфазных коммитов

Если хотя бы один из участников не может успешно завершить транзакцию, то она отклоняется (рисунок 2) [1, с. 14].

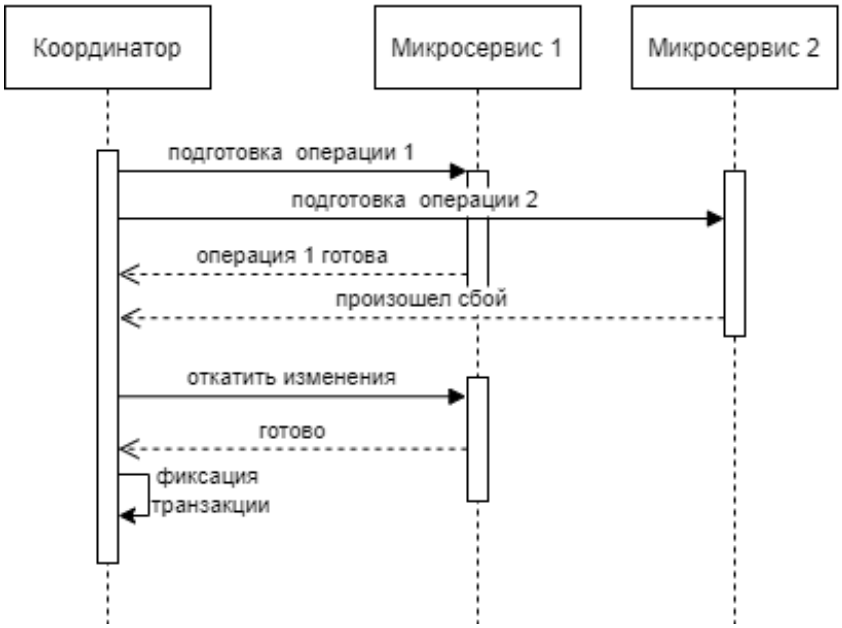

Рисунок 2. Откат изменений при ошибке в операции

С одной стороны, все правильно, если произошел сбой операция откатывается, соблюдается условие атомарности. Но здесь кроется проблема: пока микросервис 2 пытается применить операцию, микросервис 1 держит изменяемое операцией 1 значение заблокированным и ждет ответа от координатора. Узел системы не знает, что дальше делать с транзакцией, пока координатор не будет восстановлен. Если одна из операцией зависает, время

блокировки увеличивается. Еще хуже если посреди выполнения транзакции из строя выйдет координатор, полностью отвечающий за процесс.

Применение двухфазных фиксаций может значительно снизить производительность микросервисной системы при горизонтальном расширении вследствие того, что механизм является синхронным. На рисунке 3 изображены результаты исследований работы двухфазных коммитов с разным количеством сервером.

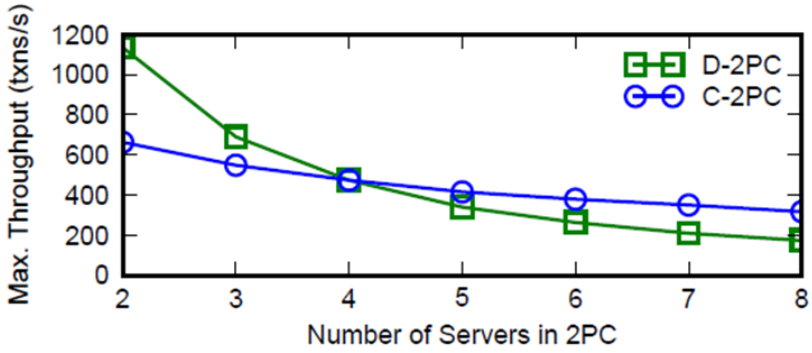

Рисунок 3. Пропускная способность транзакций при увеличении количества серверов

График отражает падение количества выполняемых операций в секунду при увеличении количества серверов, что является значительным недостатком для быстрорастущих систем [2, с. 3].

Поэтому необходимо рассмотреть альтернативные подходы к обеспечению целостности данных в микросервисной архитектуре.

Паттерн «Сага»

Сага – это последовательность операций в рамках одной транзакции, в которой каждый последующий шаг будет инициализирован предыдущим.

Не нужно контролировать каждую операцию. В отличие от 2PC, где координатор регулирует ход всей транзакции, в хореографической саге координатором является каждый участник процесса и отвечает он только за свои шаги.

Еще одним отличием от двухфазных фиксаций является то, что узлы саги не знают о схеме запроса. Им достаточно знать только предыдущий шаг. Эта особенность предоставляет возможность вносить в сагу новые узлы, не

редактируя схему где-то централизовано, что требовалось при введении новых этапов в двухфазных коммитах.

Одним из условий реализации саги является ключ идемпотентности, который предоставляется каждому запросу. По заданному ключу сервис определяет, был ли уже выполнен этот запрос раннее, и если был, то он просто игнорируется, что гарантирует единичное выполнение операции в транзакции.

Паттерн сага основан на событийно-ориентированной архитектуре. Соответственно, для его реализации также необходима единая шина событий. Если некий микросерсвис изменяет свое состояние, выполняя один из шагов саги, он создает событие в шине. Микросервисы постоянно прослушивают эти события, то есть являются подписчиками на них. При появлении прослушиваемого события они создают следующие события, добавляя его в шину. Если по какой-то причине на одном из шагов произошел сбой, создается соответствующее событие, которое прослушивают микросервисы, готовые сделать операцию отката, которая компенсирует затронутые изменения [3, с. 152].

Это можно наглядно рассмотреть на обобщенном примере покупки товара в интернет магазине (рисунок 4).

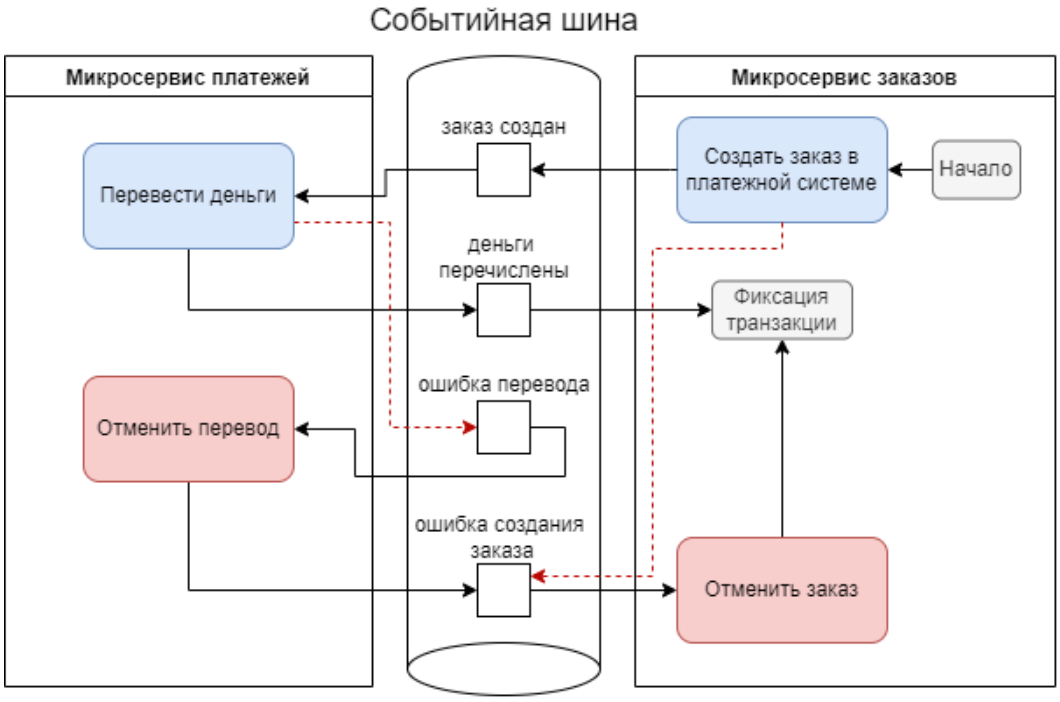

Рисунок 4. Пример хореографии саги с использованием событийной шины

Красными пунктирными линиями выделены стрелки, указывающие на альтернативные события при сбое, при получении которых микросервисы применяют операции компенсации (выделены красным).

На роль событийной шины хорошо подходит брокер сообщений Kafka, так как он обладает всеми необходимыми условиями для канала связи в паттерне саги: асинхронность, персистентность, каждое событие должно быть получено хотя бы один раз [4, с. 3]

Таким образом микросервисы, работающие по паттерну сага, самостоятельно регулируют последовательное выполнение транзакции, не нарушая атомарности. Долговечность достигается персистентностью шины, которая хранит события необходимый промежуток времени (несколько часов или день). Согласованность данных же является ответственностью разработчиков конкретных микросервисов, участников саги.

Однако условие изолированности сага не может реализовать, из-за чего в системе возможны аномалии «грязного чтения». Вспоминая о наличии версионности в канале данных, можно решить и эту проблему. Для этого достаточно использовать атомарную инструкцию CAS – compare and swap (сравнить и заменить).

Можно рассмотреть данный случай на примере создания нескольких заказов в интернет магазине и их оплаты (рисунок 5).

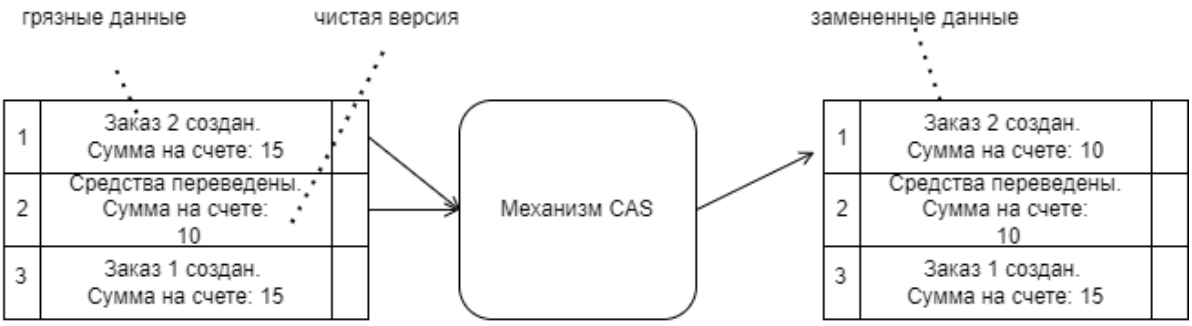

Рисунок 5. Устранение «грязного чтения» с помощью механизма CAS

Если при каком-то стечении обстоятельств при создании 2 заказа были взяты старые данные о счете, может возникнуть проблема отображения старого

состояния счета клиента. В этом случае платформа с алгоритмом синхронизации CAS сравнит предыдущую версию суммы на счете в канале данных с настоящей и при расхождении заменит грязные данные чистой версией [5, с. 2].

Заключение

Проводя небольшую ретроспективу изложенному материалу, можно выделить ряд основных преимуществ применения паттерна сага.

1. Горизонтальная масштабируемость системы больше не несет за собой просадок в производительности.
2. Сравнительно быстрое добавление новых шагов в саги, как следствие слабой связанности между операциями.
3. Поддержка целостности данных без распределенных транзакций и присущих им недостатков.

Но все же у паттерна есть два существенных недостатка, на которые необходимо обратить внимание, прежде чем применять этот подход на практике.

1. Отладка процесса саги очень проблематична. Анализировать процесс затруднительно, так как у системы нет способа получить доступ к централизованной схеме глобальной транзакции.
2. Введение паттерна может оказаться сложным, если происходит переход с монолитной архитектуры. Если микросервисная архитектура строится с нуля, то сага – хороший выбор.

Использованные источники:

1. Bernstein P. A., Principles of Transaction Processing (The Morgan Kaufmann Series in Data Management Systems) (2nd ed.) / P. A. Bernstein, E. Newcomer // Morgan Kaufmann – 2009 – 400 p.
2. Bailis P., Coordination Avoidance in Database Systems / P. Bailis, A. Fekete, M. J. Franklin, A. Ghodsi, J. M. Hellerstein, I. Stoica, // Proc. VLDB Endow. 8(3) – Nov. 2014 – P. 185–196.
3. Ричардсон К., Микросервисы. Паттерны разработки и рефакторинга / К. Ричардсон // СПб.: Питер – 2019 – 544 с.

4. Sekhar R. R., Microservices, Saga Pattern and Event Sourcing: A Survey / R. R. Sekhar, G. V. Gadad // International Research Journal of Engineering and Technology (IRJET) – 2020 – 4 p.
5. Dechev D., Pirkelbauer P., Stroustrup B., Understanding and Effectively Preventing the ABA Problem in Descriptor-Based Lock-Free Designs // ISORC '10: Proceedings of the 2010 13th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing – 2010 – P. 185–192

Used sources:

1. Bernstein P. A., Principles of Transaction Processing (The Morgan Kaufmann Series in Data Management Systems) (2nd ed.) / P. A. Bernstein, E. Newcomer // Morgan Kaufmann - 2009 - 400 p.
2. Bailis P., Coordination Avoidance in Database Systems / P. Bailis, A. Fekete, M. J. Franklin, A. Ghodsi, J. M. Hellerstein, I. Stoica, // Proc. VLDB Endow. 8(3) - Nov. 2014 - P. 185-196.
3. Richardson K., Microservices. Patterns of development and refactoring / K. Richardson // St. Petersburg: Peter - 2019 - 544 p.
4. Sekhar R. R., Microservices, Saga Pattern and Event Sourcing: A Survey / R. R. Sekhar, G. V. Gadad // International Research Journal of Engineering and Technology (IRJET) - 2020 - 4 p.
5. Dechev D., Pirkelbauer P., Stroustrup B., Understanding and Effectively Preventing the ABA Problem in Descriptor-Based Lock-Free Designs // ISORC '10: Proceedings of the 2010 13th IEEE International Symposium on Object/Component/Service- Oriented Real-Time Distributed Computing - 2010 - P. 185–192

Для цитирования: Кучеренко Н. Ю. ПРОБЛЕМА ЦЕЛОСТНОСТИ ДАННЫХ В МИКРОСЕРВИСНОЙ АРХИТЕКТУРЕ// Научный сетевой журнал «Столыпинский вестник» №4/2022