**ОБЩЕЕ ОПИСАНИЕ СИСТЕМЫ**

**RT.KEYVALUE**

2021

ОГЛАВЛЕНИЕ

[1. НАЗНАЧЕНИЕ ДОКУМЕНТА 3](#_Toc86148600)

[2. ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ 4](#_Toc86148601)

[3. НАЗНАЧЕНИЕ RT.KEYVALUE 5](#_Toc86148604)

[4. ОПИСАНИЕ RT.KEYVALUE 7](#_Toc86148606)

[5. ОПИСАНИЕ RT.KEYVALUE 8](#_Toc86148608)

[5.1. Распределение данных 8](#_Toc86148610)

[5.2. Расширение узлов 8](#_Toc86148617)

[5.3. Разделы 9](#_Toc86148618)

[5.4. Репликация 9](#_Toc86148619)

[5.5. Согласованность 10](#_Toc86148620)

[5.6. Чтение и запись данных 11](#_Toc86148621)

НАЗНАЧЕНИЕ ДОКУМЕНТА

Настоящий документ включает описание **RT.KeyValue** и описание его компонент.

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

1.
2.

Таблица 1 — Термины и определения

| Термин/сокращение | Определение |
| --- | --- |
| **Atomicity, Consistence, Isolation, Durability, ACID** | Атомарность, целостность, изолированность, постоянство. |
| **Business Intelligence, BI** | Набор IT-технологий для сбора, хранения и анализа данных, позволяющих предоставлять пользователям достоверную аналитику в удобном формате, на основе которой можно принимать эффективные решения для управления бизнес-процессами компании. |
| **Bloom filter** | крайне быстрые алгоритмы, которые помогают быстро определить, относится ли определённый элемент к указанному множеству. К нему мы обращаемся после каждого запроса. |
| **Column family** | Семейство столбцов, именованная коллекция строк |
| **Commit log** | механизм аварийного восстановления, в которую записывается любая операция в Cassandra. |
| **Gossip**  | это группа протоколов в одноранговой компьютерной коммуникации в которых распространение информации проходит способом, сводящимся к тому, что каждый или некоторые из узлов могут передавать обновляемые данные известным этому узлу соседям.  |
| **IoT** | Интернет вещей |
| **Keyspace** | Пространство ключей, группа из нескольких семейств столбцов, собранных вместе.  |
| **Mem-table** |  структура, в которую записываются данные после Сommit Log |
| **SStable** |  файл на дисковом пространстве, в которую данные записываются после того, как количество данных достигла указанного “порога” |
| **Кластер** | Группа серверов и координирующего программного обеспечения, объединённых логически, способных обрабатывать идентичные запросы и использующихся как единый ресурс. |
| **Шардирование** | Техника масштабирования при работе с данными. Суть его в разделении (партиционирование) базы данных на отдельные части так, чтобы каждую из них можно было вынести на отдельный сервер. |

НАЗНАЧЕНИЕ RT.KEYVALUE

1.

**RT.KeyValue** – это NoSQL распределенная база, обладающая высокой масштабируемостью и доступностью без ущерба для производительности. Линейная масштабируемость и проверенная отказоустойчивость на стандартном оборудовании или облачной инфраструктуре делают её идеальной платформой для критически важных данных.

Основным назначением RT.KeyValue является:

1. Загрузка и хранение структурированных данных.
2. Анализ данных.
3. Работа с Business Intelligence (BI) приложениями.
4. Работа со высоконагруженными сервисами.

**RT.KeyValue** может выполнять автоматизацию следующих бизнес-процессов:

1. Использование RT.KeyValue в организации стриминговых сервисов, т.к. система обеспечивает очень низкую задержку срабатывания и максимально возможного количества операций при пиковой нагрузке, с возможностью отслеживать действия пользователей (например, аудио, видео, игры, подкасты).
2. Использование RT.KeyValue в сферах ритейла, интернет-маркетинга, интернет рекламы и прочих за счёт устойчивого хранения большого объёма постоянно изменяющихся данных в строгой структуре (например, рекомендательные системы, оптимизация каталогов продукции т.д.).
3. Использование RT.KeyValue в сферах IoT, ЖКХ и промышленности благодаря возможности быстрой обработки огромного количества разных данных (например, данные с IoT-датчиков и смарт устройств).

**RT. KeyValue** обладает следующими особенностями:

1. **Масштабируемость и надежность** в связи с отсутствием центрального сервера (Master Node), отказ которого может стать причиной сбоя всего кластера. Простое добавление новых узлов в кластер и обновление версии RT.KeyValue, без дополнительного ручного вмешательства и переконфигурации всего кластера.
2. **Гибкая схема данных**, основанная на комбинации столбцовых семейств (Column Family) в пространство ключей (keyspace), позволяя определять в строках одной и той же таблицы разные столбцы и таким образом эффективно хранить разряженные таблицы.
3. **Высокая пропускная способность**, особенно для операций записи – благодаря хранению в оперативной памяти ответственного узла, данные записываются быстрее, чем считываются. Кроме того, за счет использования LSM-деревьев, RT.KeyValue также позволяет достаточно быстро считывать данные. LSM-дерево (Log-structured merge-tree, журнально-структурированное дерево со слиянием) – это структура данных, которая хранит пары «ключ — значение» и предоставляет быстрый доступ по индексу в условиях частых запросов на вставку, например, при хранении журналов транзакций.
4. **CQL язык запросов (собственный SQL-подобный язык запросов)**, который позволяет выполнять простейшие запросы SELECT с выборкой по определённому условию. Добавление и обновление осуществляется через единое выражение UPDATE, операция INSERT отсутствует. Также CQL поддерживает пространство имён и семейств столбцов. Драйверы с поддержкой этого языка запросов реализованы для языков Python, Java, Ruby, PHP, JavaScript и Perl.
5. **Поддержка поиска за счет вторичных индексов**, создать которые можно с помощью CQL-выражения CREATE INDEX;
6. **Настраиваемая согласованность и поддержка репликации,**когда пользователи могут сами определить необходимое количество реплик и задать уровень согласованности данных по каждой операции хранения. В частности, при каждом чтении и записи клиент может указать желаемый уровень согласованности (консистентности).
7. **Автоматическое разрешение конфликтов** благодаря хранению временных меток (timestamp) для каждого столбца в любой записи. К примеру, если одновременно делаются два запроса на изменение значений разных столбцов одной записи, то возможен конфликт на узлах, отвечающих за хранение этой строки. Конфликт будет успешно разрешен по времени последнего изменения столбцов.
8. **Поддержка аутентификации, ролей** и работы с клиентом по защищенному криптографическому протоколу **SSL**.
9. **Поддержка ACID-транзакционности**на уровне одной записи, т.е. для набора столбцов с одним ключом.

ОПИСАНИЕ RT.KEYVALUE

1.

Компоненты **RT.KeyValue** представлены ниже (Рисунок 1).



Рисунок 1. Схема компонентов RT.KeyValue

RT.KeyValue включает в себя следующие функциональные компоненты, предназначенные для решения комплексных задач:

**Кластер** – компонент системы, включающий один и более дата центр.

**Центр обработки данных (дата центр)** – набор узлов, которые взаимодействуют друг с другом.

**Узел** – место хранения данных.

Узлы кластера RT.KeyValue равноценны и пользователи могут соединятся с любым из них для записи или чтения.

**Узел–координатор** – узел, выполняющий координацию запросов пользователя всем узлам-реплик.

**Узел-реплика** – узел, выбранный для сохранения записи.

ОПИСАНИЕ RT.KEYVALUE

1. 1. Распределение данных
2.
3.
4.
5.
6. 1.

Одним из важных атрибутов RT.KeyValue является то, что ее базы данных распределены.  Базы данных RT.KeyValue легко масштабируются, когда приложение находится в состоянии повышенной нагрузки, а распределение также предотвращает потерю данных из-за сбоя оборудования любого конкретного центра обработки данных. Распределенная архитектура также обеспечивает расширенные технические возможности; например, разработчик может настроить пропускную способность запросов чтения или запросов записи по отдельности.

"Распределенный" означает, что RT.KeyValue может работать на нескольких машинах, представляясь пользователям как единое целое. Нет особого смысла запускать RT.KeyValue как единый узел, хотя это очень полезно, чтобы помочь разобраться в том, как он работает. Но чтобы получить максимальную выгоду от RT.KeyValue, надо запускать ее на нескольких машинах.

Поскольку это распределенная база данных, RT.KeyValue может (и обычно имеет) иметь несколько узлов. Узел представляет собой один экземпляр RT.KeyValue. Эти узлы взаимодействуют друг с другом с помощью протокола под названием gossip, который представляет собой процесс одноранговой связи между компьютерами. RT.KeyValue также имеет архитектуру без мастера – любой узел в базе данных может обеспечивать ту же функциональность, что и любой другой узел, что способствует надежности и устойчивости RT.KeyValue. Несколько узлов могут быть логически организованы в кластер или "кольцо". У вас также может быть несколько центров обработки данных.

* 1. Расширение узлов

RT.KeyValue позволяет разработчикам динамически масштабировать свои базы данных, используя готовое оборудование, без простоев. Расширяться или сжиматься, когда нужно, если появляются такие требования.

RT.KeyValue позволяет легко увеличить объем данных, которыми она может управлять. Поскольку RT.KeyValue основан на узлах и можно сказать масштабируется горизонтально. Например, чтобы удвоить емкость или пропускную способность, удвойте количество узлов. Эта линейная масштабируемость может применяться практически бесконечно.

* 1. Разделы

В RT.KeyValue данные автоматически распределяются с (положительными) последствиями для производительности. Это достигается с помощью разделов. Каждый узел владеет определенным набором токенов, и RT.KeyValue распределяет данные на основе диапазонов этих токенов по всему кластеру. Ключ раздела отвечает за распределение данных между узлами и важен для определения местоположения данных. Когда данные вставляются в кластер, первым шагом является применение хеш-функции к ключу раздела. Выходные данные используются для определения того, какой узел (на основе диапазона токенов) получит данные.

Когда поступают данные, координатор базы данных берет на себя работу по назначению данного раздела. Любой узел в кластере может взять на себя роль координатора. Узлы сплетничают (gossip) друг с другом; во время этого они сообщают, какой узел отвечает за какие диапазоны Координатор выполняет поиск: на каком узле находится нужный токен? Когда он находит нужный, он пересылает эти данные на этот узел. Узел, которому принадлежат данные для этого диапазона, называется узлом-репликой. Один фрагмент данных может быть реплицирован на несколько узлов-реплик, обеспечивая надежность и отказоустойчивость. До сих пор наши данные были реплицированы только в одну реплику. Это представляет собой коэффициент репликации, равный единице, или RF = 1.

Узел координатора не является единым местоположением, если бы это было так, то система была бы хрупкой. Это просто узел, который получает запрос в данный конкретный момент. Любой узел может выступать в качестве координатора.

* 1. Репликация

Один фрагмент данных может быть реплицирован на несколько узлов (реплик), обеспечивая надежность и отказоустойчивость. RT.KeyValue поддерживает понятие коэффициента репликации (RF), который описывает, сколько копий данных должно существовать в базе данных. Если увеличить коэффициент репликации до двух (RF = 2), данные также должны храниться на второй реплике, и, следовательно, каждый узел становится ответственным за вторичный диапазон токенов в дополнение к своему первичному диапазону. Коэффициент репликации, равный трем, гарантирует, что существует три узла (реплики), охватывающих этот конкретный диапазон токенов, и данные хранятся еще на одном.

Распределенная структура RT.KeyValue делает ее более устойчивой и производительной, это действительно вступает в силу, когда есть несколько реплик для одних и тех же данных. Это помогает системе самовосстанавливаться, если что-то пойдет не так, например, если выйдет из строя узел или выйдет из строя жесткий диск. Репликация гарантирует, что данные не будут потеряны. Если поступает запрос на получение данных, даже если одна из наших реплик вышла из строя, две другие все еще доступны для выполнения запроса. Координатор также хранит “подсказку” для этих данных, и когда сбитая реплика вернется, он выяснит, что он пропустил, и ускорит работу с двумя другими репликами. Никаких ручных действий не требуется, это делается полностью автоматически.

Использование нескольких реплик имеет преимущества в производительности. RT.KeyValue автоматически реплицирует данные в разных ваших центрах обработки данных. Ваше приложение может записывать данные на узел RT.KeyValue в одной части страны и эти данные автоматически доступны в центрах обработки данных на другой части страны и даже больше. Это демонстрирует положительные преимущества в производительности, особенно если поддерживается огромная база пользователей. В мире, зависящем от облачных вычислений и быстрого доступа к данным, ни один пользователь не страдает от задержек из-за расстояния.

* 1. Согласованность

В RT.KeyValue можно настроить согласованность на основе каждого запроса. В этом контексте уровень согласованности представляет минимальное количество узлов RT.KeyValue, которые должны подтвердить операцию чтения или записи координатору, прежде чем операция будет признана успешной. Как правило, уровень согласованности (CL) выбирается на основе коэффициента репликации. В приведенном ниже примере данные реплицируются на три узла. У нас есть CL= QUORUM (Кворум, относящийся к большинству, в данном случае 2 реплики или RF/2 +1), поэтому координатору необходимо будет получить подтверждение от двух реплик, чтобы запрос считался успешным.

Как и в случае с другими вычислительными задачами, может потребоваться определенный навык, чтобы научиться настраивать эту функцию для обеспечения идеальной производительности, доступности и целостности данных. Тот факт, что можно управлять такой детализацией, означает, что позволено управлять развертываниями в мельчайших деталях.

RT.KeyValue все равно, где вы его разместите – on-prem, облачном провайдере или нескольких облачных провайдерах и т.д. Пользователь может использовать их комбинацию для одной базы данных, это дает разработчикам максимальную гибкость.

* 1. Чтение и запись данных

Во время операции чтения, RT.KeyValue получает данные и mem-table и проверяет Bloom Filter для поиска SSTable, которая содержит необходимые данные.

Каждая операция записи перехватывается Commit Log, которые записываются в узлы. Позже, данных перехватываются и сохраняются в mem-table. Когда mem-table заполнена, данные записываются в SSTable файл. Все записи автоматически разбиваются на части и реплицируются в кластере. С определённой периодичностью, RT.KeyValue консолидирует SS таблицы, убирая лишние данные.