**2. Особенности распределенных баз данных**

Нереляционные подходы хранения данных в качестве главной цели ставят функционирование на большом кластере, следовательно, главный инструмент распределения – **горизонтальное масштабирование**.

Преимущества:

* Возможность обработки больших объемов данных;
* Возможность обработки более интенсивного трафика запросов;
* Избегаем перегрузки.

Недостаток: сложность организации и управления базой данных.

Основные инструменты распределения данных: **репликация и фрагментация.** Репликация – это копирование одних и тех же данных на нескольких узлах. Фрагментация означает размещение разных данных на разных узлах. Методы могут работать как порознь, так и вместе. Но может и использоваться односерверная репликация (т.е. отсутствие репликации вообще).

**Проблемы организации фрагментации:**

В идеальном случае при фрагментации разные пользователи будут обращаться к разным узлам (каждый пользователь с одним сервером). Таким образом, главный вопрос – как сгруппировать данные, чтобы один пользователь преимущественно обращался к одному серверу данных. Здесь удобной является агрегатная ориентация. Агрегаты располагаются по серверам с учетом географического расположения или в целях равномерного распределения рабочей нагрузки. Может производиться логическая фрагментация исходя из приложения (ответственность разработчиков приложения) или автоматическая фрагментация (производится СУБД).

Фрагментация хороша для улучшения производительности, причем и чтения, и записи.

Фрагментация почти не влияет на отказоустойчивость, даже может ухудшать ее, так как кластеры обычно используют слабые компьютеры.

Для повышения отказоустойчивости более важна репликация, которая бывает двух видов: «ведущий-ведомый» и одноранговая.

**Характеристики репликации «ведущий-ведомый»**

Один узел назначается ведущим, который является источником данных и несет ответственность за модификацию данных. Остальные узлы – ведомые (вторичные). Процесс репликации синхронизирует данные ведущего и ведомых узлов.

Такая репликация очень полезна для масштабирования чтения, но никак не помогает при интенсивном трафике на операции записи, так как требуется репликация изменений.

Преимущество репликации «ведущий-ведомый» - отказоустойчивость чтения (наличие копии на других серверах и быстрое переназначение ведущего сервера – односерверная система с оперативным резервом).

Проблема: несогласованность – существует вероятность, что чтение одних и тех же данных с нескольких ведомых серверов даст разные результаты (недавно произошла запись и новые данные еще не до всех ведомых узлов дошли).

**Характеристики одноранговой репликации**

Устраняется ведущий узел – все узлы равноправны. Устраняется узкое место (единственное место отказа) – ведущий узел. Легко повысить производительность, добавляя новые узлы. Проблема – несогласованность записи (попытка двумя пользователями произвести запись одних и тех же данных на двух серверах), несогласованность чтения.

**Главная проблема распределенных баз данных – проблема согласованности**

Реляционные базы данных обеспечивали строгую согласованность. Базы данных NoSQL выделяют более мягкие виды согласованности, которые дают возможность появления некоторых ошибок данных.

1. **Согласованность обновлений.**

Попытка изменить одну и ту же запись двумя пользователями одновременно может привести к конфликту «запись-запись». Различают пессимистический и оптимистический подходы. Пессимистический подход ориентирован на предотвращение возможных конфликтов (например, с помощью блокировок, хотя, например, при одноранговой репликации это очень сложно). Оптимистический подход ориентирован на идентификацию и применение мер по устранению конфликтов (но несогласованность какое-то время существует в системе) (используются алгоритмы, аналогичные алгоритмам управления версиями – сохранение конфликтующих данных обновления и записи, последующее объединение с помощью пользователя или автоматически). Однако проблема обеспечения согласованности обновлений – алгоритмы не всегда позволят обеспечить должную производительность плюс существует вероятность взаимных блокировок. Репликация усугубляет проблему согласованности «запись-запись».

1. **Согласованность чтения.**

Проблема несогласованного («грязного») чтения приводит к конфликту «чтение-запись». Логическая согласованность гарантирует, что разные элементы данных будут изменяться вместе (в реляционных БД это достигается за счет транзакций). Но не все NoSQL базы данных используют транзакции. В ряде случаев поддерживается атомарность обновления отдельных единиц данных (в пределах агрегата, но не между агрегатами).

Другая проблема – согласованность реплик. Обязательно будет существовать окно несогласованности (на одном сервере данные изменились, на другом – нет). Впоследствии изменение будет распределено по репликам и ситуация будет приведена к **итоговой согласованности**.

Более важной является обеспечение сессионной согласованности (т.е. пользователь, который изменил данные увидит именно их даже после разрыва сессии). При ее отсутствии изменение может коснуться одной реплики, а за счет чтения с другой реплики (например, за счет перераспределения нагрузки) пользователь не увидит своего изменения. Обычно для сессионной согласованности используется механизм «липкой» сессии (обслуживание одной сессии с одного узла), но тогда будет существовать проблема балансировки рабочей нагрузки. Другой способ – штампы версий (любое взаимодействие должно содержать штамп последней версии в рамках сессии и при чтении с другого сервера должно быть гарантировано, что версия обеспечивается).

Есть сложности с производительностью при обеспечении сессионной согласованности, особенно при репликации «ведущий-ведомый». Варианты решений – переключение записи на ведомый узел для обеспечения сессии (ведомый узел перешлет обновленную информацию на ведущий узел); временное переключение клиента на ведущий узел на время работы с операциями обновления до момента, когда обновления достигнут всех реплик.

**Таким образом, приходим к ослаблению требования согласованности раде повышения производительности.**

**Теорема САР** (consistensy, availability, partition tolerance – согласованность, доступность и устойчивость к разделению). Сформулирована Эриком Брюером в 2000 г. и доказана Сетом Гильбертом и Нэнси Линч через несколько лет. Основная идея – из трех свойств (согласованность, доступность и устойчивость к разделению) можно обеспечить не более двух. Доступность здесь понимается как способность работающего узла обеспечить выполнение любого поступившего запроса. Устойчивость к разделению понимается как возможность кластера восстанавливать обмен данными после обрыва связей.

Типовой пример СА – односерверная система (нет проблемы разделения). Таким образом, для обеспечения всех трех свойств требуется ослабить два из свойств (обычно согласованность и доступность).

Результат – NoSQL базы данных обеспечивают вместо ACID-свойств BASE-свойства (Basically Available, Soft state, Eventual consistency – доступность в большинстве случаев, неустойчивое состояние, итоговая согласованность). Часто проблемы решают в зависимости от предметной области. Например, электронная биржа требует актуальных данных, а в службе новостей актуальность не обязательна (пользователь может получить сообщение с опозданием).

**Проблема долговечности**

Существует проблема несохраненных обновлений за счет достижения повышения производительности. Так, изменения в основном проводятся в оперативной памяти и периодически сбрасываются на диск. Сбой может привести к потере последних обновлений. В ситуации применения репликаций возможен сбой ведущего сервера до момента синхронизации с ведомыми серверами. Компромиссом в этом случае является обеспечение отклика клиенту только после получение ответа от ведомых узлов о получении обновления – опять ухудшение производительности.

**Кворум – один из общепринятых вариантов компромиссов**

Кворум – для обеспечения строгой согласованности при репликации не требуется опрос всех узлов.

Кворум записи – требуется подтверждение записи более чем половиной реплик (W>N/2, где N – количество узлов, обслуживающих чтение, W – количество узлов, обслуживающих запись). Количество реплик для чтения называют коэффициентом репликации.

Кворум чтения более сложно понятие: R+W>N (R – количество узлов, с которыми необходимо связаться, W – количество узлов, которые подтвердили запись, N – коэффициент репликации).

Крайний случай – требовать подтверждения со всех узлов. Тогда читать можно с одного узла.

**Определение возникшей несогласованности**

Возникновение несогласованности диагностируют с помощью оптимистической автономной блокировки – форма обновления, при которой есть гарантия, что изменения не будут основываться на устаревших данных. Оптимистическая автономная блокировка – это ситуация, когда операция клиента считывает любую информацию, от которой зависит логическая транзакция, и проверяет, изменилась ли информация с момента ее чтения (метод «сравнить и установить»). Основной метод – штампы версий (дополнительное поле, которое содержит версию: счетчик версии (через ведущий узел), создание идентификатора на основе комбинаций дат, состояний аппаратного обеспечения (может генерировать и ведомый узел, но сложно обеспечить корректное сравнение), хэширование содержимого записи, метки последнего обновления).

Проблема – обеспечение и синхронизация версий при репликациях. Используется вектор версий, анализ которых может привести к выявлению несогласованности. Особенно популярен для одноранговой репликации. При обмене данными узлы синхронизируют свои вектора версий. Тут имеется простой способ сравнения, на какой машине более новая версия. Не исключена ситуация, когда версии совпадают, а данные нет – это как раз и диагностирует конфликт «запись-запись», что требует отдельных процедур для обеспечения согласованности