Георaспределённые транзакции в YTSaurus

Магистерская диссертация

Смородинов Александр, MCKH231

Научный руководитель:
руководитель службы разработки динамических таблиц,
ООО "Яндекс.Технологии",
Савченко Руслан Алексеевич

Современные Компьютерные Науки
ФКН НИУ ВШЭ (Москва)

Июнь 2025

Содержание

Содержание

Транзакции

- **Транзакция** - последовательность из одной или нескольких операций с базой данных (БД), рассматриваемых как единое целое

Транзакции

- **Транзакция** - последовательность из одной или нескольких операций с базой данных (БД), рассматриваемых как единое целое
- Основные свойства:
  - Атомарность (atomicity)
  - Консистентность (consistency)
  - Изоляция (isolation)
  - Устойчивость (durability)

Транзакции

- **Транзакция** - последовательность из одной или нескольких операций с базой данных (БД), рассматриваемых как единое целое
- Основные свойства:
  - Атомарность (atomicity)
  - Консистентность (consistency)
  - Изоляция (isolation)
  - Устойчивость (durability)
- Виды транзакций:
  - Последовательные
  - Параллельные
  - Распределённые
  - Георасределённые

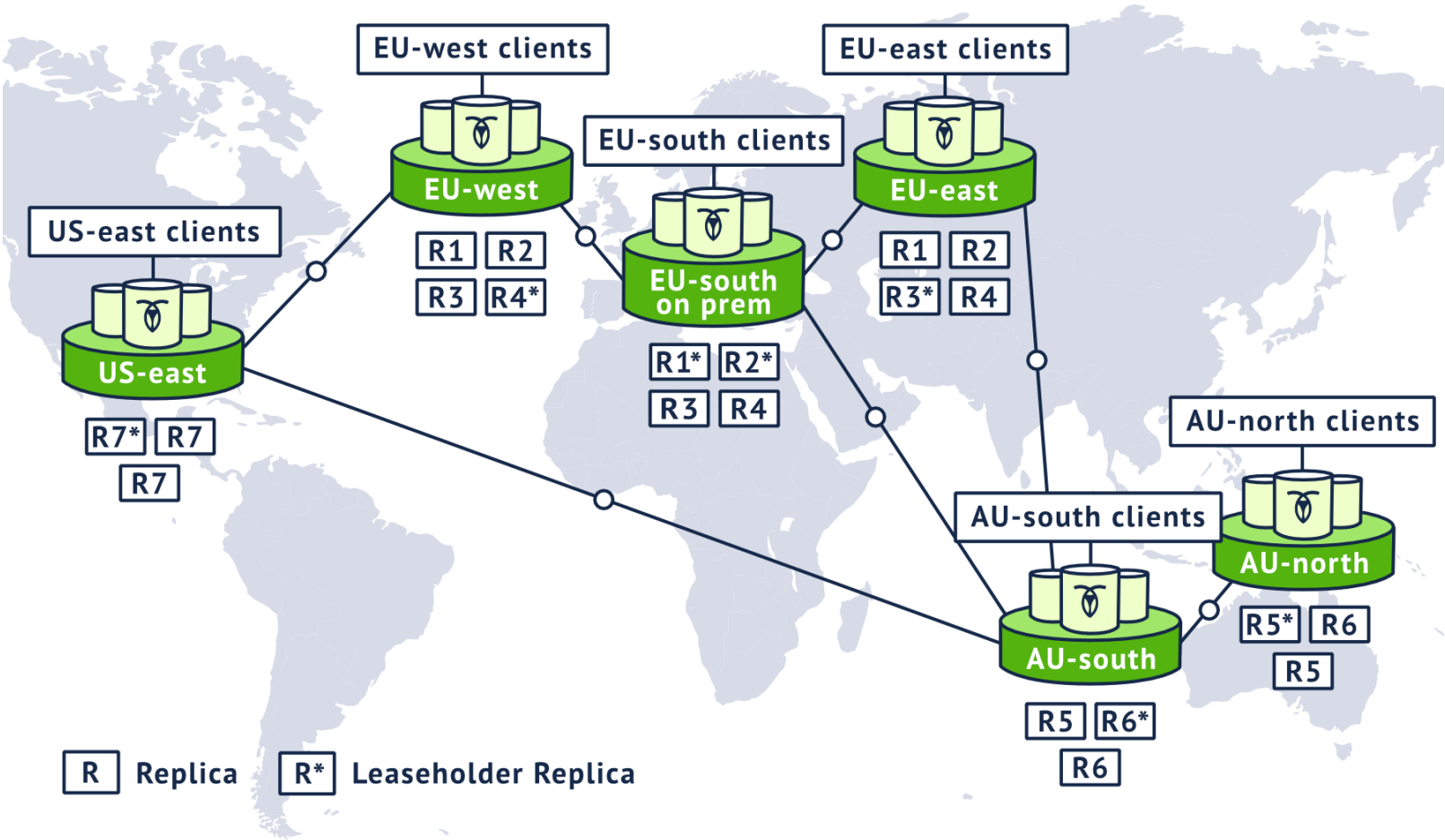
Шардирование и репликация

- Если хранить все данные на одном сервере, то возникают следующие проблемы:
  - Ограничения по ресурсам одного сервера
  - Высокая задержка
  - Нарушение требований по хранению персональных данных
  - Единая точка отказа

Шардирование и репликация

- Если хранить все данные на одном сервере, то возникают следующие проблемы:
  - Ограничения по ресурсам одного сервера
  - Высокая задержка
  - Нарушение требований по хранению персональных данных
  - Единая точка отказа
- Поэтому, необходимо:
  - Разделить таблицу на несколько частей (шардов, или таблетов)
  - Реплицировать данные

Пример георасределённой базы данных



Cockroachdb: The resilient geo-distributed sql database / Rebecca Taft, Irfan Sharif, Andrei Matei и др.

Геораспределённые транзакции

- Необходимо гарантировать свойства ACID для распределённых транзакций
- Походы по сети дорогие $\Rightarrow$ необходимо минимизировать походы в другие локации

ACID

- Атомарность - с помощью двухфазного коммита (2PC)
- Изоляция - с помощью Multiversion Concurrency Control (MVCC) + двухфазной блокировки (2PL)
- Консистентность - следует из свойств транзакций и изоляции
- Устойчивость - с помощью репликации и записи данных на диск (не рассматривается в данной работе)

Multiversion Concurrency Control

- Multiversion concurrency control (MVCC) - один из распространённых методов для изоляции транзакций
- Основная идея MVCC - хранить для каждого ключа историю его изменений
- Нужно на каждое чтение и коммит генерировать временные метки
- Временные метки должны быть уникальны и монотонны

Генерация временных меток

- YTSaurus - единый источник (timestamp_provider)
- Google Spanner - TrueTime
- CockroachDB - гибридные логические часы (HLC)

Содержание

Цели работы

- Изучить, реализовать и протестировать георасперделённые транзакции
- Добавить поддержку в YTSaurus локального источника времени (HLC), избавиться от timestamp_provider
- Сравнить производительность методов

Задачи работы

- Изучить материалы по георасперделённым транзакциям
- Реализовать алгоритмы для обеспечения ACID с использованием HLC: 2PC, 2PL+MVCC
- Подготовить тестовый стенд для симуляции георасперделённой системы
- Написать тесты для проверки корректности и производительности
- Написать текст диссертации и сопроводительную документацию (readme файл)

Содержание

Актуальность задачи

1. Востребованность:
   - Транзакции значительно упрощают написание надёжных и производительных сервисов
   - Всё больше и больше приложений становятся георасперделёнными

Актуальность задачи

1. Востребованность:
   - Транзакции значительно упрощают написание надёжных и производительных сервисов
   - Всё больше и больше приложений становятся георасперделёнными
2. Нерешённость:
   - Георасперделённые транзакции эффективно реализованы в CockroachDB и в Spanner, но не в YTSaurus

Значимость задачи

1. YTSaurus используется в Яндексе во многих внутренних сервисах $\Rightarrow$ оптимизация транзакций позволит улучшить задержку запросов и уменьшить потребление ресурсов
2. У Яндекса есть 5 датацентров, хотя они расположены относительно близко друг к другу
3. YTSaurus имеет открытый исходный код $\Rightarrow$ любая компания может поднять свою инсталяцию (например Yango, Nebius)

Датацентры Яндекса

https://yandex.ru/jobs/services/datacenters

Содержание

Обзор существующих решений

- Google Spanner
  - (2PL (RW) + MVCC (RO)) × 2PC
  - Генерация временных меток: TrueTime
  - TrueTime - позволяет синхронизировать часы с небольшим ($\sim$ 7 мс) расхождением.

Обзор существующих решений

- Google Spanner
  - (2PL (RW) + MVCC (RO)) × 2PC
  - Генерация временных меток: TrueTime
  - TrueTime - позволяет синхронизировать часы с небольшим ($\sim$ 7 мс) расхождением.
- YTSaurus
  - (2PL (RW) + MVCC (RO)) × 2PC
  - Генерация временных меток: timestamp_provider

Обзор существующих решений

- Google Spanner
  - (2PL (RW) + MVCC (RO)) × 2PC
  - Генерация временных меток: TrueTime
  - TrueTime - позволяет синхронизировать часы с небольшим ($\sim$ 7 мс) расхождением.
- YTSaurus
  - (2PL (RW) + MVCC (RO)) × 2PC
  - Генерация временных меток: timestamp_provider
- CockroachDB
  - (2PL (RW) + MVCC (RO)) × 2PC
  - Генерация временных меток: HLC

Обзор существующих решений

- Google Spanner
  - (2PL (RW) + MVCC (RO)) × 2PC
  - Генерация временных меток: TrueTime
  - TrueTime - позволяет синхронизировать часы с небольшим ($\sim$ 7 мс) расхождением.
- YTSaurus
  - (2PL (RW) + MVCC (RO)) × 2PC
  - Генерация временных меток: timestamp_provider
- CockroachDB
  - (2PL (RW) + MVCC (RO)) × 2PC
  - Генерация временных меток: HLC
- YugabyteDB
  - Аналогично CockroachDB

Отличия от существующих решений

- Идея использовать HLC взята из CockroachDB, в этом плане, работа не имеет теоретической новизны
- Основные результаты:
  1. Обзор существующих решений
  2. Имплементация HLC
  3. Сравнение производительности и тесты корректности
  4. Стенды для симуляции георасределённой БД
- В тестовом окружении есть допущения, например, что сервера не отказывают

Содержание

Полученные результаты

1. Код на C++ для клиентской и серверной части БД
   https://github.com/asmorodinov/distributed_transactions
2. Реализованные алгоритмы:
   1. Двухфазный коммит (2PC)
   2. Двухфазная блокировка (2PL)
   3. Мультиверсионное хранилище (MVCC)
   4. Предотвращение дедлоков (Wait-Die)
   5. Гибридные логические часы (HLC)
   6. Неблокирующие чтения
3. Конфигурация docker и docker compose для симуляции георасперделённой системы
4. Стресс-тесты
5. Техническая документация (readme файл)
6. Текст диссертации

Полученные результаты

1. На текущий момент, код в YTSaurus ещё не закоммичен
2. Вместо этого, HLC реализация была написана в отдельном тестовом окружении
3. Коммит в YTSaurus остаётся для будущей работы

Содержание

Детали реализации

- Приложение на C++
- Зависимости: yt/yt/core
- Система сборки: yatool (ya make)
- Кроме самого приложения написаны тесты и конфигурация для docker и docker compose
- Репозиторий: https://github.com/asmorodinov/distributed_transactions

Содержание

1 тест (инкременты)

1. Начальное состояние: $X_1 = X_2 = \ldots = X_N = 0$
2. Транзакция: $X_1 \mathrel{+}= x, X_2 \mathrel{+}= x, \ldots, X_N \mathrel{+}= x$, где x - случайное число
3. Проверка консистентности: $X_1 = X_2 = \ldots = X_N$

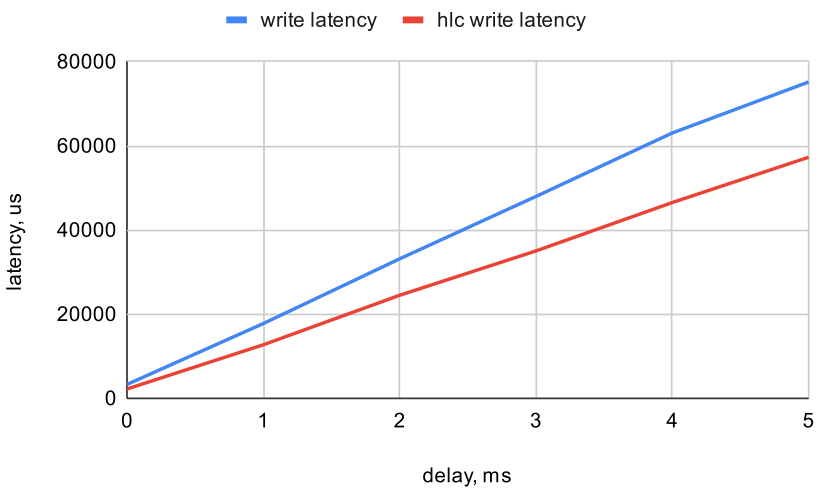
2 тест (переводы)

1. Начальное состояние: $X_1 = X_2 = \ldots = X_N = 0$
2. Транзакция: $X_i -= x, X_j += x$, где x - случайное число, $i \neq j$ - случайные индексы
3. Проверка консистентности: $X_1 + X_2 + \ldots + X_N = 0$

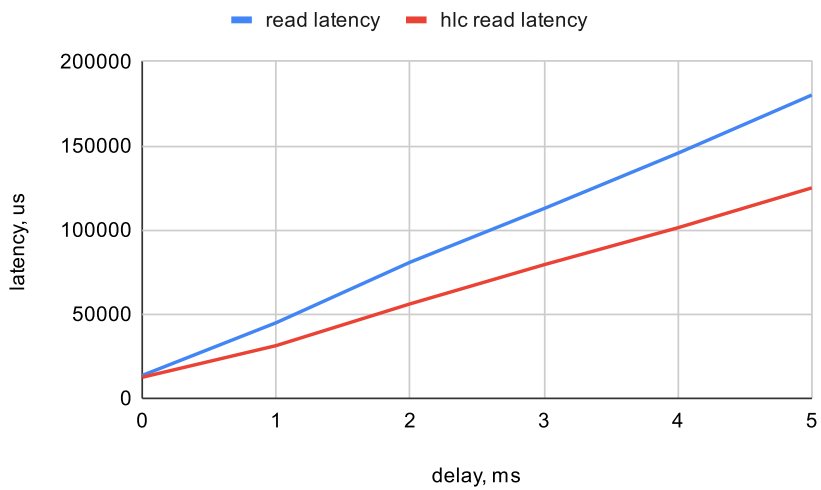
Конфигурация сети

1. Для каждого контейнера можно настроить исходящую задержку
2. Было протестировано 3 сценария:
   1. Между всеми контейнерами задержка t
   2. У timestamp_provider задержка t, у других контейнеров нулевая
   3. У клиента задержка t, у остальных контейнеров нулевая

1 сценарий



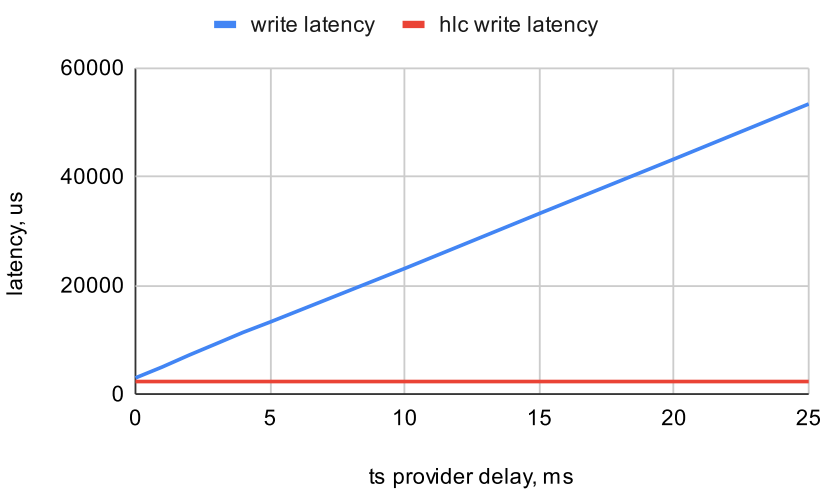
(a) Задержка записей



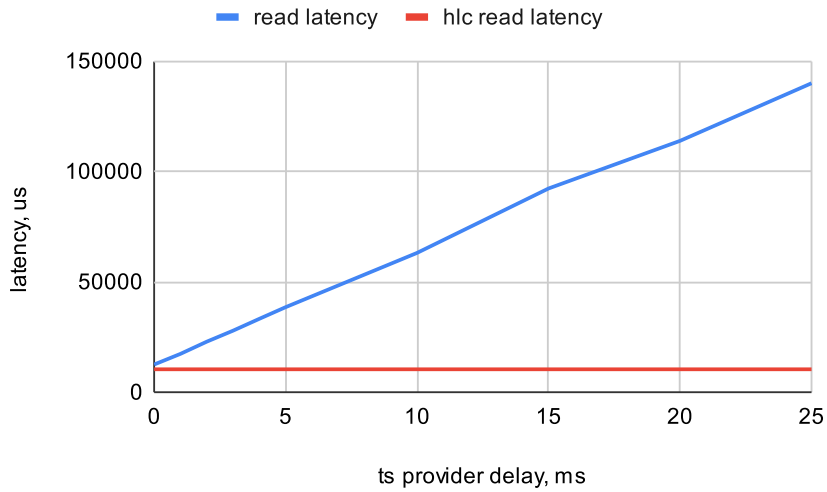
(b) Задержка чтений

Зависимость задержки записей и чтений от задержки между контейнерами

2 сценарий



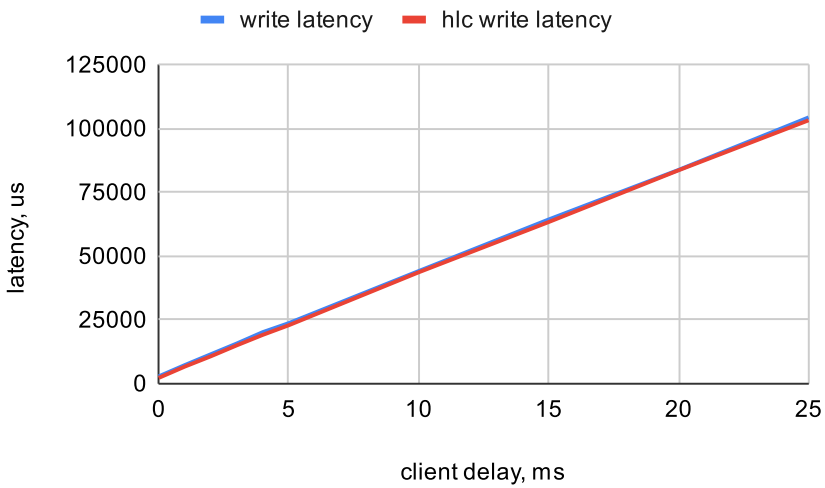
(c) Задержка записей



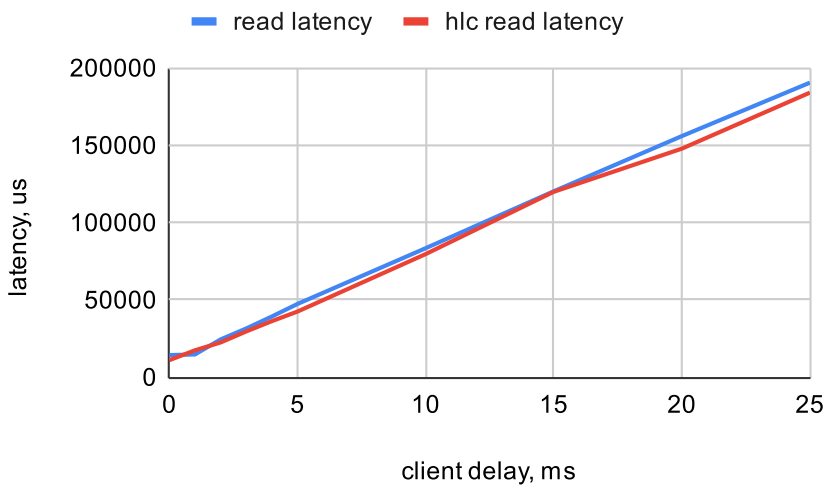
(d) Задержка чтений

Зависимость задержки записей и чтений от задержки между кластером и timestamp_provider

3 сценарий



(e) Задержка записей



(f) Задержка чтений

Зависимость задержки записей и чтений от задержки между клиентом и кластером

Содержание

Заключение

1. Были реализованы алгоритмы для георасределённых транзакций
2. По итогам тестов, было показано, что метод с HLC более эффективный, при этом корректность не нарушается
3. Тесты можно запускать с помощью docker compose и настраивать задержку между контейнерами
4. Коммит в YTSaurus остаётся для будущей работы

Спасибо за внимание
Готов ответить на вопросы