Yandex Cloud

Миграция приложения с PostgreSQL на Yandex Database

Производительность. Стоимость. Риски

Александр Смирнов

Эксперт разработки систем хранения и обработки данных

Предпосылки к исследованию

- **YDB** — консистентная распределенная SQL БД, предоставляется в Yandex.Cloud в виде Dedicated- или Serverless-сервиса.
- YDB развивается как полноценная альтернатива традиционным RDBMS, обладающая дополнительными преимуществами.
- **PostgreSQL** — самая распространенная традиционная RDBMS, доступная в виде Managed Service у множества cloud-провайдеров.
- Что может дать YDB Serverless по сравнению с PostgreSQL сегодня? С какими проблемами столкнется разработчик? Как нам приоритизировать задачи по дальнейшему развитию YDB?

Вопросы

- Возможна ли переделка с минимальным рефакторингом и в чем она будет состоять?
- Какие характеристики задержек и производительности?
- Какие характеристики стоимости?
- Какие риски появляются для клиента?

Выбор тестового сценария

Требования

- Критичность к консистентности
- Неравномерная транзакционная нагрузка на чтение и запись
- Масштабируемость
- Использование SQL для работы с данными
- Востребованность вторичных индексов
- Типичность для множества бизнес-кейсов
- Простота

Ответ: Python E-commerce application

E-commerce application

Модель данных

Orders — заказы

ID
Customer CreatedDa teTime ProcessedDateTime

Index: Customer, CreatedDateTime

OrderLines — позиции

OrderID Product
Quantity

Stock — остатки

Product
Quantity

Функции

Регистрация нового заказа

1. Запись строки в таблицу orders, множества связанных строк в таблицу orderLines.
2. Исполнение заказа: проверка достаточности товара в остатках, уменьшение остатков, пометка заказа как исполненного.

Получение истории заказов клиента

› Выборка первых N строк из таблицы orders для заданного клиента, отсортированных в порядке убывания даты создания.

Инфраструктура тестирования

PostgreSQL

- Managed PostgreSQL Yandex.Cloud в двух вариантах узлов (2CPU/8GB RAM, 4CPU/16 GB RAM)
- Два узла в разных AZ с синхронной репликацией
- 10GB Network-attached SSD storage
- Стоимость в Yandex.Cloud – 7000 (2CPU), 14000 (4CPU) рублей в месяц

YDB

- Serverless с синхронными копиями данных в трех AZ
- Стоимость по выполненным запросам (13 руб. за 1 M RU) + объем хранения (13 руб. за 1GB)

Приложение

- VM в той же AZ где master node PostgreSQL

PostgreSQL-версия

- Пакет psycopg2 для работы с PostgreSQL из Python
- Read Committed
- Генерация идентификатора заказа из последовательности
- Многопоточное приложение с пулом соединений
- Фиксированный набор товаров с остатками
- Функция инициализации модели данных и остатков

Изменения взаимодействия с SDK

- Connection PostgreSQL = Session YDB, ThreadedConnectionPool PostgreSQL = SessionPool YDB
- Управление транзакциями в PostgreSQL методами Connection,
  в YDB транзакция — отдельный объект, создаваемый из Session.
- В YDB нет класса Cursor, execute() является методом Session
  и возвращает множество выборок.
- YDB execute() принимает параметры только по именам.
- Дополнительные возможности YDB SDK по сравнению с psycopg2:
  - Autocloseable-конструкции для работы с ресурсами (не требуют
    явного освобождения их в секции finally)
  - Session.execute() принимает флаг Commit для исключения лишнего
    запроса к серверу на подтверждение транзакции
  - Конструкции-обертки для реализации стратегий повторных попыток SessionPool.retry_operation()
  - Режимы транзакций со сниженными гарантиями консистентности (например, StaleReadOnly)

Изменения в SQL

- Параметры требуют явной декларации типов (DECLARE в тексте SQL)
- Нет последовательных генераторов, все ID — random 64bit
- Обращение к таблице по вторичному индексу должно быть явно указано в SELECT (view <index_name>)
- Serializable является основным уровнем изоляции в YDB
- В YDB не видны свои изменения посередине транзакции
- Дополнительные возможности YQL по сравнению с PostgreSQL:
  - UPSERT как метод слепой записи
  - Передача списков структур как параметров
  - Возможность объединения нескольких SQL-выражений в одном запросе к серверу, в том числе с возвращением нескольких выборок

Пример переработанного метода

PostgreSQL

```
def getOrderHistory_pg( pool, customer, limit=10 ):
  conn = pool.getconn()
  try:
    query = """
      select * from orders
      where customer = %s
      order by customer desc, created desc
      limit %s
    """
    cur = conn.cursor()
    cur.execute(query, ( customer, limit ))
    result = cur.fetchall()
    return result
 finally:
   cur.close()
   pool.putconn( conn )
```

Yandex Database

```
def getOrderHistory( pool, customer, limit=10 ):
  result = [None]
  def getit( session ):
    query = session.prepare( """
      DECLARE $cust as Utf8;
      DECLARE $limit as UInt32;
      select * from orders view ix_cust
      where customer = $cust
      order by customer desc, created desc
      limit $limit;
    """ )
    result[0] = session.transaction( ydb.StaleReadOnly())
      .execute( query,
      { "$cust": customer, "$limit": limit },
      commit_tx=True
    )[0].rows

  pool.retry_operation_sync( getit )

return result[0]
```

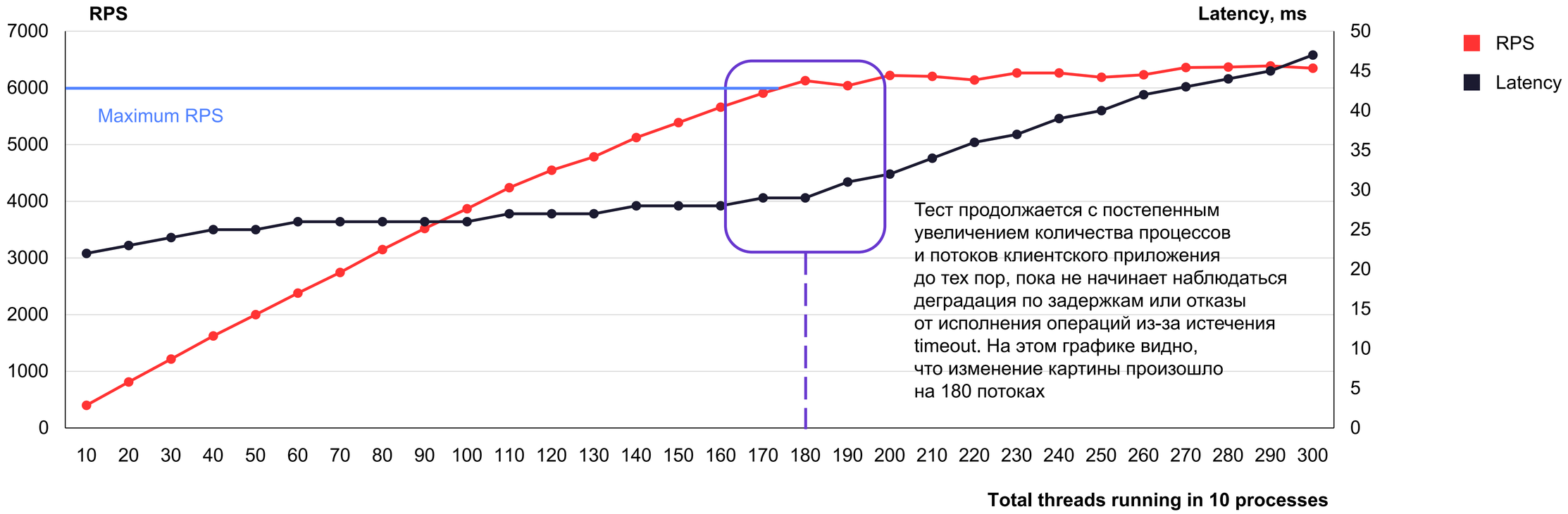
Тестовые сценарии

> **Цель — поиск максимальной производительности, определение ограничивающих производительность факторов**

- Однопоточное исполнение
- Многопоточное исполнение на одном ключе — проверка concurrency на запись и на чтение
- Эмуляция реальной нагрузки:
  - Генерация случайных заказов: количество позиций в заказе — распределение Парето, выбор товара в позицию — нормальное распределение, выбор клиента — равномерное распределение
  - Получение истории заказов случайного клиента: равномерное распределение
  - Смешанная нагрузка
- Клиентское приложение содержит прикладной код + генератор нагрузки

Методика определения максимальной производительности

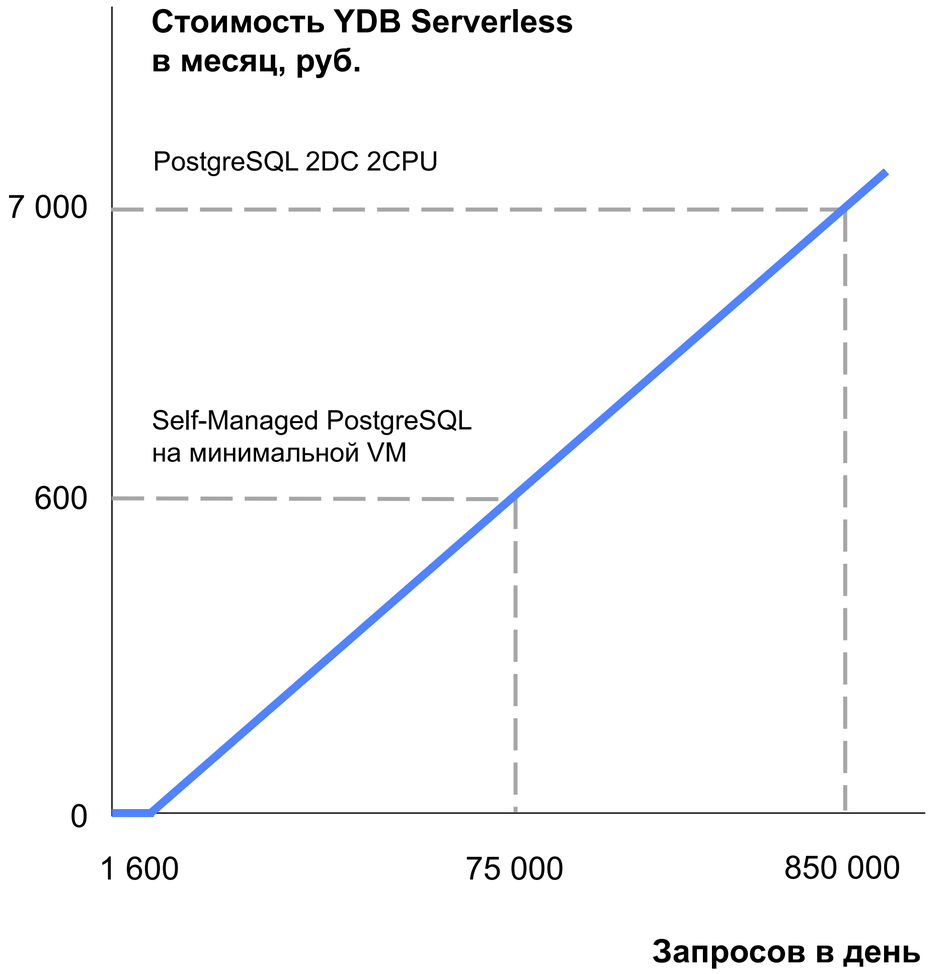
Пропускная способность и задержка в зависимости от количества потоков: чтение последних 10 записей о заказах одного клиента



Производительность и задержки

Сценарий	PostgreSQL 2CPU/4CPU	Yandex Database Serverless
Добавление новых случайных заказов без исполнения	1200 TPS 30ms/ 2800 TPS 24 ms	10000+ BTPS 100ms
Однопоточное создание заказа	30 BTPS 28ms	6 BTPS 170ms
Однопоточное получение истории заказов	1000 RPS 1ms 80 RPS 13ms (клиент в другом ЦОД)	44 RPS 23ms
Создание заказов с одним одинаковым товаром (проверка contention на update остатков)	75 BTPS 80ms	18 BTPS 280ms (geodistributed commit)
Получение истории по одинаковому клиенту	2300/5000 RPS 3ms	6000 RPS 32ms (max 1 shard, no read replicas)
Создание случайного заказа на 10000 товаров, 1000 клиентов	450/900 BPS, 80ms, pauses	500 BPS 200ms, contention < 0,02% 1900 BPS 250ms, contention < 1% 2500 BPS 380ms, contention < 2% …
Максимум производительности получения истории заказов случайных клиентов из 1000	2300/4700 RPS, 4ms	15000+ RPS 32ms
Смешанная случайная нагрузка: • Получение истории заказов + • Регистрация нового заказа	1000/2500 RPS 30/20ms 160/550 BTPS 200/100ms	5400 RPS 50ms 1300 BTPS 300ms …

Стоимость



Параметр	Значение
Добавление заказа с 1 позицией	12 RU
Добавление заказа с 5 позициями	38 RU
Получение 10 последних заказов клиента	20 RU
Добавление случайного заказа	20 RU

1M RU = 13 рублей

1M RU в месяц бесплатно

Риски

- YDB — очень быстрая БД при выполнении тяжелых запросов. Вы можете потратить много RU за очень маленькое время
- Использование ненужных гарантий значительно влияет на показатели

Выводы

1. Использование Serverless YDB на самом деле позволяет платить пропорционально активности пользователей, не тратя никаких усилий на администрирование БД
2. Транзакционное приложение, написанное для PostgreSQL с использованием SQL, может быть адаптировано под YDB за разумное время с возможностью сохранения модели данных
3. Для реальных нагрузок в E-commerce приложениях стоимость Serverless YDB может быть существенно меньше, чем стоимость сопоставимых решений на базе PostgreSQL
4. YDB Serverless позволяет успешно обработать пиковые нагрузки, не требуя оплаты аренды большой машины под PostgreSQL

Заключение

- Мы развиваемся в направлении соответствия стандартам. В ближайшее время выходит поддержка JDBC, далее в Roadmap ODBC, ANSI SQL, поддержка популярных ORM, что в пределе приведет к сокращению трудоемкости рефакторинга приложений вплоть до нуля
- Мы развиваемся в направлении снижения рисков. В нашем Roadmap интерактивные клиентские квоты, сниженные тарифы по прогнозируемой нагрузке
- Мы развиваемся в плане улучшения нефункциональных характеристик. В ближайших релизах – поддержка MVCC.
- Но уже сегодня рефакторинг под YDB может дать вам многократный выигрыш в стоимости эксплуатации транзакционного приложения в production по сравнению с PostgreSQL или другими классическими RDBMS
- Следите за новостями

Контакты Yandex Database

Телеграм-чат

t.me/yandexdatabase_ru

Twitter

twitter.com/yandexdatabase

Yandex Cloud

Спасибо!

Александр Смирнов

Эксперт разработки систем хранения и обработки данных

t.me/yandexdatabase_ru

twitter.com/yandexdatabase