Яндекс Деньги

Дао миграции нагруженного сервиса с Oracle на PostgreSQL

Василий Созыкин, ведущий Java-разработчик

Начало пути

Решение о миграции

2 года назад руководство компании Яндекс.Деньги приняло окончательное решение о миграции с Oracle на PostgreSQL.

Аргументы экономические

- Oracle дорогой как в лицензиях, так и в поддержке
- Рост количества микросервисов и команд, который потребовал большое количество небольших баз данных на боевой и дев. средах
- Подходил плановый срок замены оборудования,
  решили убить сразу двух зайцев

Аргументы технические

- Бесплатная, активно развивающаяся инфраструктура вокруг PostgreSQL
- Отказоустойчивость
- Для перехода от интеграционного тестирования к компонентному потребовалась функциональность быстрого поднятия нового экземпляра БД
- Возможность офлайн работы с локальной БД

1 год назад…

Преграда на пути

Серьезный соперник

Небольшие сервисы уже переведены на PostgreSQL.

Мы столкнулись с настоящим сервисом-тяжеловесом и

старые подходы уже не работали,

потребовалось промышленное решение.

Непростая миграция

Сервис «Профиль пользователя»:

- Не допустимы простой сервиса и потеря данных
- 55 таблиц, самая большая - 1.5 Tb,
  нагрузка на БД больше 800 rps
- Legacy код, почти без тестов, 5-10 лет давности
- Релизное окно - 1 раз в неделю и
  сервис активно развивается – 40 комитов в день

Месяц спустя

Всё та же преграда

Нужен другой путь

Миграция по частям в разных командах не дала никаких результатов

Для миграции выделяется отдельная команда из 4 разработчиков и 1 DBA

Разрабатывается план миграции для минимизации сроков и трудозатрат

Кандидаты

- **Ora2Pg**
- **SymmetricDS**
- Oracle GoldenGate
- xDB Replication Server
- ORQ
- Full Convert

- Oracle to PostgreSQL Migration
- ESF Database Migration Toolkit
- OraDump-to-PostgreSQL
- Oracle-to-PostgreSQL
- SQLData Tool

Автоматизация процесса миграции

Миграция DDL:

- Ora2Pg — http://ora2pg.darold.net

Миграция данных:

- SymmetricDS — https://www.Symmetricds.org

Миграция кода:

- copy-paste и топором, напильником, скальпелем

Взгляд за горизонт

Оценка трудозатрат

Приблизительные оценки:

- 1 таблица перенос кода, тесты, pull request – 1-2 дня
- Релиз – длительность 3-5 дней
- Время миграции данных - не больше 1-2 часов
- За один раз можно мигрировать 3-5 таблиц
- Порядка 7 сложных таблиц в БД с большими рисками

Возможность оптимизации

- Релиз 1 раз в неделю был узким местом.
  При 2 релизах успевали переводить 5-7 таблиц.
- Использовали возможность получения опыта на простых таблицах. Сложные оставили на конец.
- Не делать ручного тестирования.
  100% покрытие юнит-тестами с БД и приемка.
- Избегать миграции с «самописной» синхронизацией.
  По оценке и по факту она в 3 раза дольше.

Итого по оценкам в неделю

- Минимум 1 релиз в неделю
- 3-7 таблиц на бою переключены на PostgreSQL
- 12 таблиц переведены по коду для PostgreSQL
- 1 день команды - на улучшение инфраструктуры: ускорение релизного цикла,
  автоматизация,
  стабилизация тестов.

Наивное представление

- Пишем код для работы с PostgreSQL под выключенным «рубильником»,
- Выкатываем код на продакшен,
- Мигрируем данные таблиц,
- Переключаем «рубильник»,
- PROFIT!

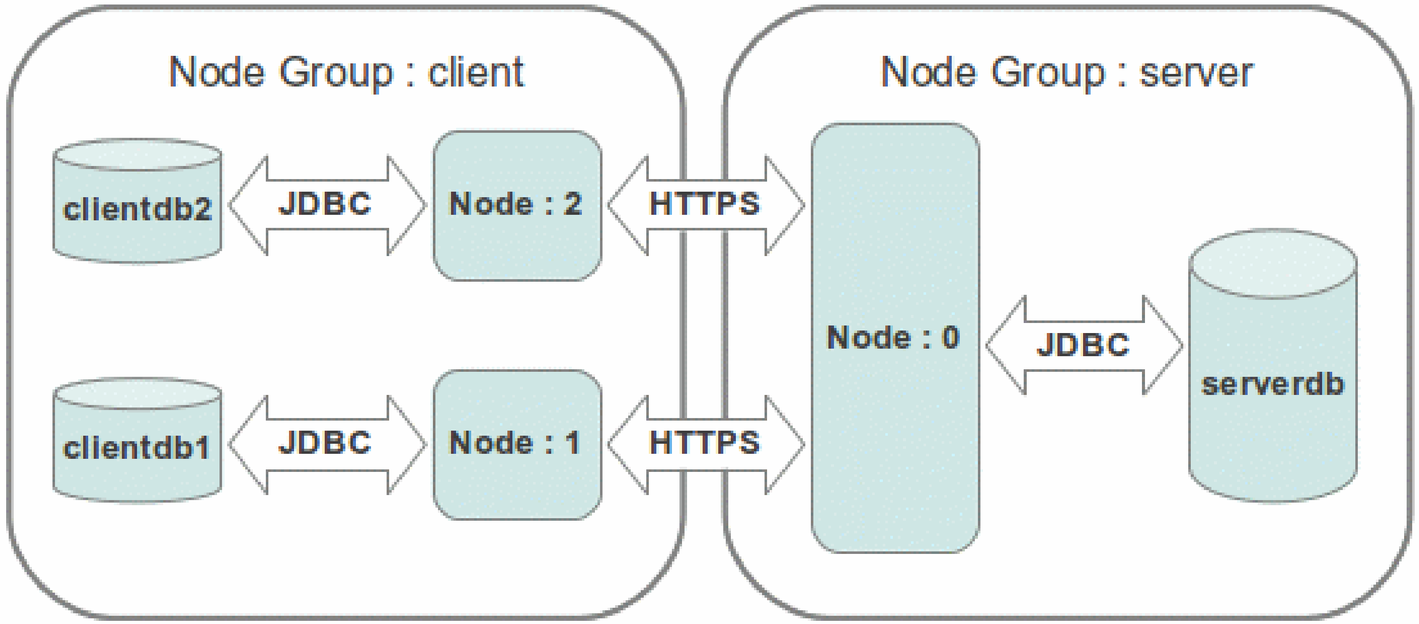
Месяц спустя

Поехали

Шаг 1. Подготовка

- DBA настраивает SymmetricDS на продакшен зоне
- Перед миграцией разработчик релизит новый код с поддержкой работы на PostgreSQL, под «рубильником»
- DBA получает от разработчика DDL таблицы в PostgreSQL, проверяет DDL на корректность
- DBA создаёт таблицу и её окружение в PostgreSQL.

SymmetricDS inside



Шаг 2. Миграция данных

Создание триггера на синхронизацию:

```
INSERT INTO sym_trigger (trigger_id, source_table_name,
channel_id, last_update_time, create_time)
VALUES (‘sometable_trigger', ‘sometable', ‘sample_db',
current_timestamp, current_timestamp)
```

Шаг 2. Миграция данных

Создание связки роутера и триггера синхронизации:

```
INSERT INTO sym_trigger_router (trigger_id, router_id,
initial_load_order, last_update_time, create_time)
VALUES ('sometable_trigger', 'ora_2_pg', 100,
current_timestamp, current_timestamp)
```

Шаг 2. Миграция данных

Создание задачи на initial load:

```
INSERT INTO sym_data (node_list, table_name, event_type,
row_data, trigger_hist_id, channel_id, create_time)  SELECT
'001', t.source_table_name, 'R', '1=1', h.trigger_hist_id,
t.channel_id, current_timestamp
FROM sym_trigger t INNER JOIN sym_trigger_router tr
ON t.trigger_id = tr.trigger_id
INNER JOIN sym_trigger_hist h ON h.trigger_hist_id = (SELECT
max(trigger_hist_id) FROM sym_trigger_hist WHERE trigger_id
= t.trigger_id) WHERE channel_id = 'some_db' AND
tr.router_id LIKE 'ora_2_pg' AND (t.source_table_name LIKE
'some_table') ORDER BY tr.initial_load_order ASC
```

Шаг 2. Миграция данных

Убедиться что данные начали синхронизироваться

Примерно через 30 мин. – 1 час проверить,

что initial load закончился

Шаг 2. Окончание initial load

Шаг 3. Переключение рубильника

Увеличить значение sequence для PK в PostgreSQL.

Переключиться на работу с PostgreSQL,
внимательно наблюдать за производительностью приложения и ошибками в логах.

Шаг 3. Возможный возврат

В зависимости от таблицы будет 5-10 минут,

чтобы переключиться обратно на Oracle

Последний шаг

Переименовать таблицу в Oracle.

Следим за логами!

Месяц спустя

Первые победы

Первая ретроспектива

Мигрировано уже около 20 таблиц.

Командой выработан checklist миграции.

Выработаны правила валидации миграции.

Checklist мигратора

- Стараться мигрировать связанные таблицы пачкой
- Держать под общим переключателем связанные по SQL таблицы

Checklist мигратора

- Написать тесты на все ветки работы с БД в коде
- Проверить соответствие типов данных при передаче из кода в БД

Checklist мигратора

- Подумать как восстановить данные в случае отката
- Не забыть сдвинуть sequence для PK перед переключением таблицы

Валидация миграции

- Сверка количества записей по количеству записей
- Получение из двух БД выборки с n произвольными соответствующими записями,
  их программное сравнение

Месяц спустя

Набираем обороты

Вторая ретроспектива

Мигрировано уже около 40 таблиц.

Набили уже много шишек из-за отличий

PostgreSQL от Oracle.

Начали использовать advanced возможности SymmetricDS.

Отличия PostgreSQL от Oracle

- Транзакционный DDL. Транзакция в 8 часов.
- Переход с rownum = 1 на limit 1 без order by.
  Oracle – index scan, PostgreSQL – full scan.
- PostgreSQL открывает очень много file handlers.
- Oracle - настраиваемый размер страницы памяти,
  PostgreSQL – 8Kb.

Advanced SymmetricDS

- Конвертация данных при миграции.
- Миграция части данных с ограничением по условию.
- Миграция таблиц с BLOB > 4 Kb.

Две недели спустя

Внезапная остановка

Идём на ощупь

Начались самые сложные таблицы.

SymmetricDS не справляется с объемами,

мигрируем на самописных очередях в БД.

Миграция выполняется с полной обратной совместимостью, риск отката велик.

Миграция в ручном режиме

SymmetricDS на нашем железе падает при миграции таблиц > 100Gb с OutOfMemory.

Написали собственных механизм миграции:

- При вставке и изменении записи, создаем в БД задачу на синхронизацию данных.
- Выполняем физическую репликацию записи.
- Initial load делаем через этот же механизм, но пачками по диапазону идентификаторов.

Недостатки ручной миграции

- Возможны ошибки.
  Совместили очередь задач синхронизации измененных записей и initial load.
  Синхронизация – быстрая и там много задач,
  initial load – просто долгая задача.
  В итоге количество задач выросло до миллиона.
- Нужно тестирование. Дополнительные трудозатраты.

Последние 1.5 месяца

Сложный участок пути

Колдобины и выбоины на пути

- Deadlock в двух БД
- Неучтенный SQL Join
- Ошибка дублирования для SymmetricDS

Колдобины и выбоины на пути

- Откат на Oracle
- Изменения в неявной транзакции
- Другой сервис

98%

История платежей пользователей

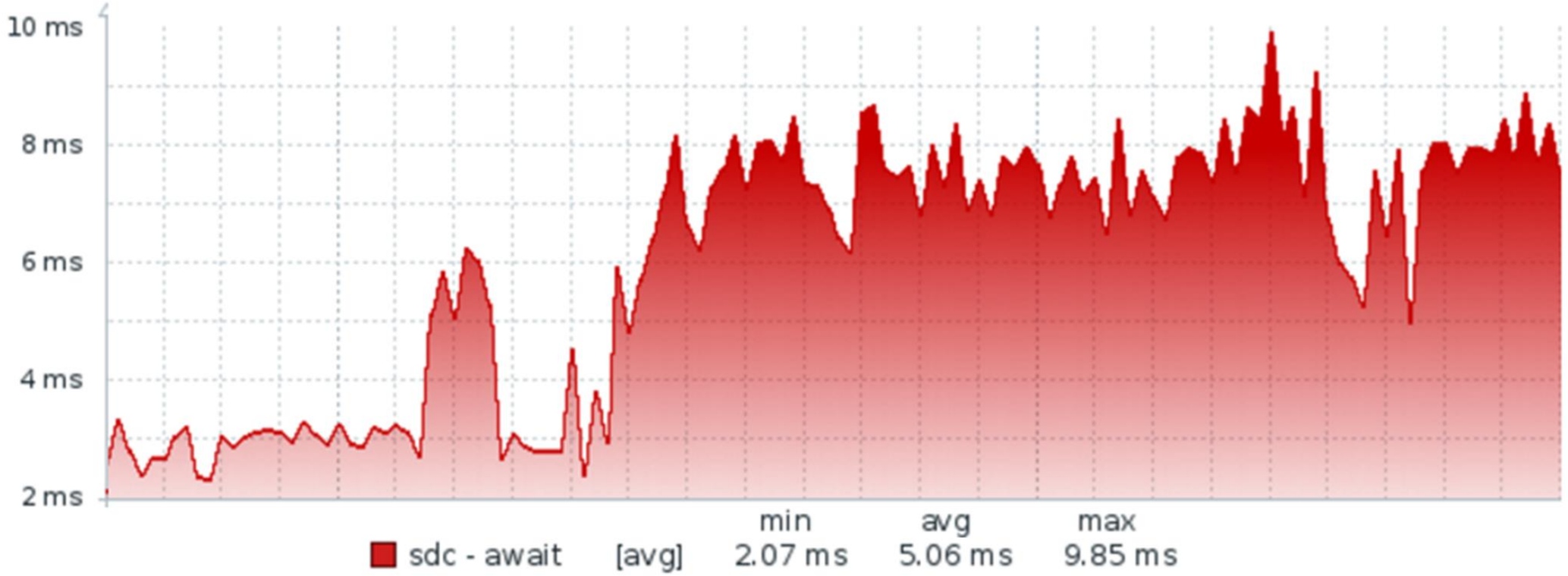
Самая проблемная таблица размером 1.5 Tb.

При переключении на PostgreSQL получили 20 кратную деградацию производительности, откатились.

Проблема

Все запросы были index scan, как и в Oracle.
Уперлись в дисковые чтения.

Диск загружен на 100%

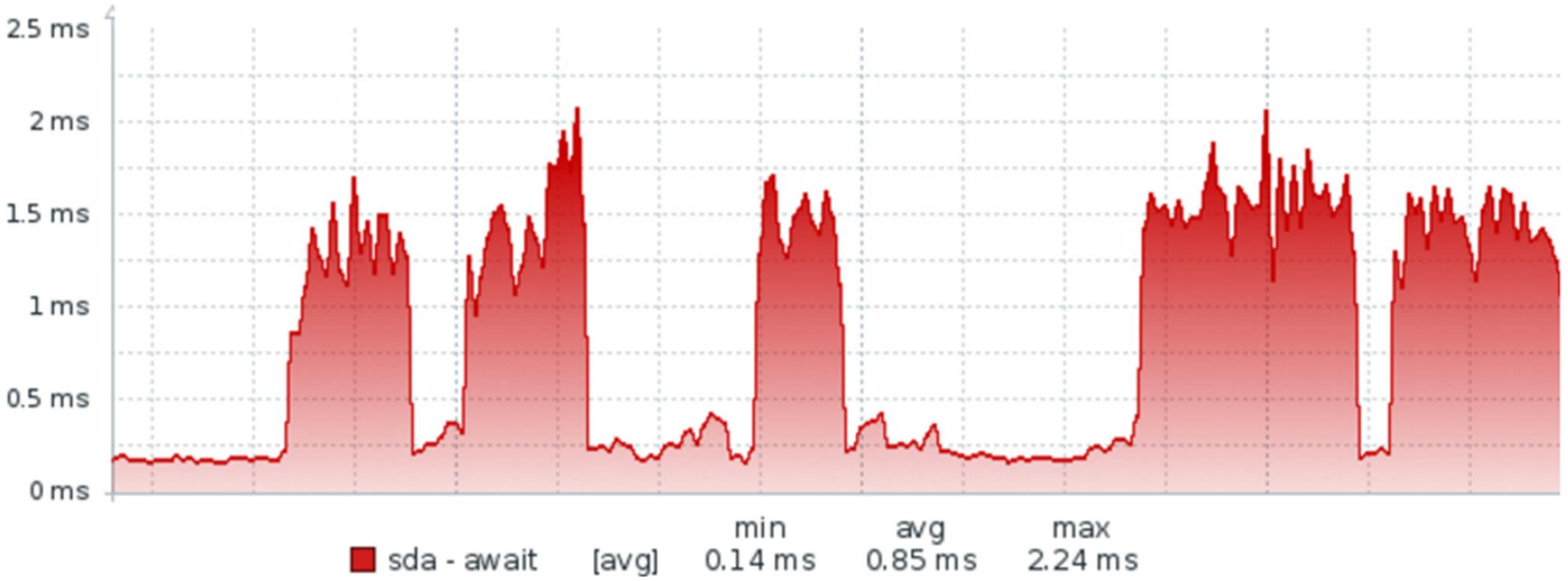


Триумф или позор?

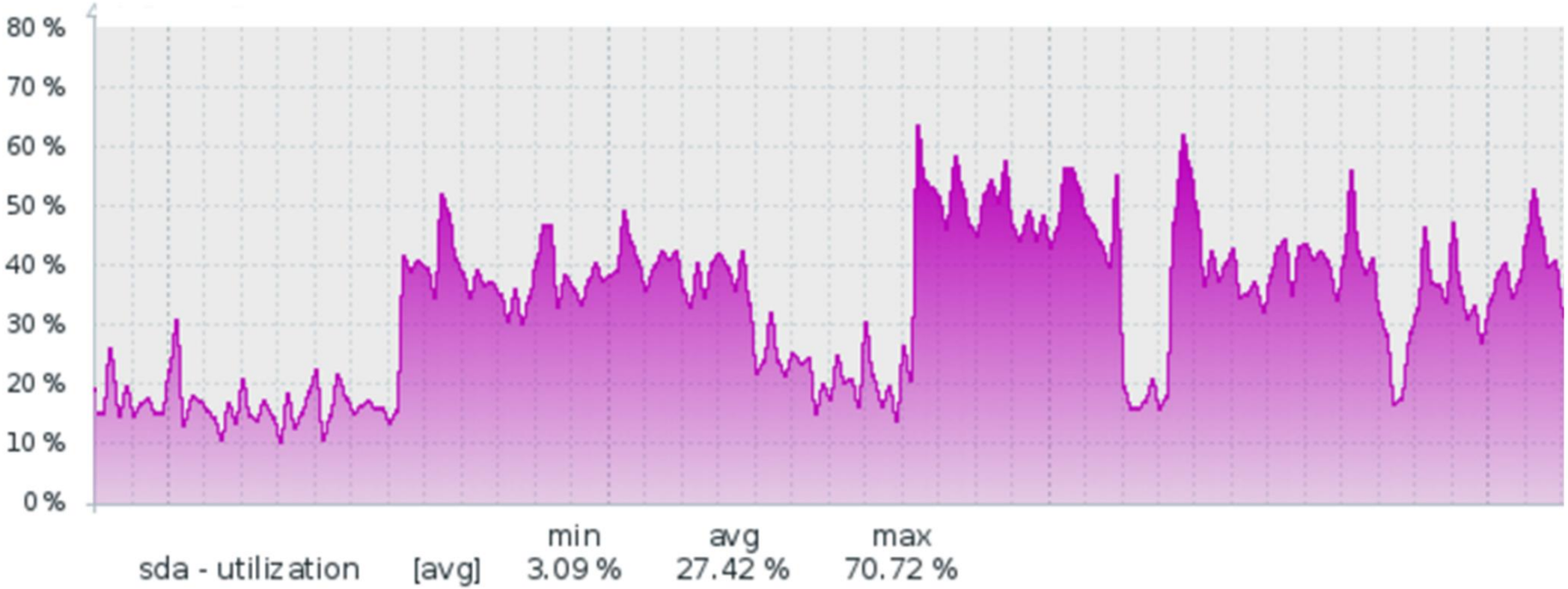
Решение

В таблице был CLOB для неструктурированных данных,
который использовался только в части запросов.
Вынесли CLOB в отдельную таблицу,
перенесли основную таблицу на SSD.

Диск успевает отвечать



Загрузка диска < 30%



Победа

Сервис «Профиль пользователя» успешно переключен на работу с PostgreSQL

Итоги пути

Начало

Oracle

- Version - 11.2.0.2.
- RAM – 64Gb.
- Intel(R) Xeon(R) CPU E5630 / 2.53GHz / 2 CPU * 4 ядра с HT.
- SAS DELL/JBOD/ASM normal redundancy.
- Размер БД - 3Tb

Окончание

PostgreSQL

- Version - 9.5.4.
- RAM – 128Gb.
- Intel(R) Xeon(R) CPU E5-2690 v4 / 2.60GHz / 2 CPU * 14 ядер.
- Storage - SAS DELL/RAID 10/FS xfs для data/index/xlog,
  SSD RAID 10 для индексов и таблиц с высоким доступом OLTP.
- Размер БД - 2Tb

Выводы

- Миграция на другой тип СУБД реальна.
- PostgreSQL – production ready. PROVED!
- SymmetricDS - silver bullet для типовых сценариев, но надо уметь его готовить.
- Правило 80/20 работает и для миграции.
- Мониторинг, мониторинг, мониторинг!

Спасибо за внимание!

Контакты

Созыкин Василий

Ведущий Java-разработчик

 vsozykin@yamoney.ru

 +7 904 551 41 33