PostgreSQL как ядро биржи интернет рекламы

Adsterra.com

Юрий Соболев

PgConf.Russia 2016

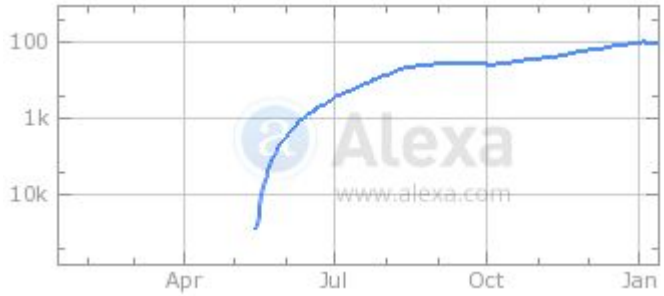
ADSTERRA NETWORK

PostgreSQL

Mediatech

Информация об adsterra.com

1. До 150 млн показов баннеров в сутки. До 15 млн показов в час.
2. Запись в Postgresql до 10000 событий в секунду.
3. 15 отдельных серверов под БД.
4. Команда разработчиков: 19 человек. Разработчики БД: 2.5 человека.



ADSTERRA NETWORK

PostgreSQL

Mediatech

Об авторе:

Об авторе:

ADSTERRA NETWORK

PostgreSQL

Mediatech

История создания

1. Май 2014. Adsterra 1.0: WEB Интерфейс к другому ад-серверу(SAAS).
2. Пришел заказ: до конца 2014 года сделать все полностью на своем решении.
3. Есть 30 млн показов баннеров в сутки и очень слабое представление, как это все работает.
4. Есть 1 Back-end разработчик, 1 Front-end и я.
5. Смотри предыдущий слайд:).

ADSTERRA NETWORK

PostgreSQL

Mediatech

Но у нас уже был опыт HighLoad!

1. Были сайты до 5 млн просмотров в сутки. Суммарно больше 10 млн с нескольких сайтов.
2. Как работали с данными: Mysql+Sphinx+Redis.
3. Плюсы: это работало.
4. Минусы: неконсистентность, несколько точек отказа, у каждого свои геморрои с настройкой.
5. В общем, боль.

ADSTERRA NETWORK

PostgreSQL

Mediatech

Для каких проектов такое подходит:

ADSTERRA NETWORK

PostgreSQL

Mediatech

Что хотелось от новой разработки:

1. Использовать максимально готовые решения.
2. Хранить данные в 1 месте. В 1 надежном месте.
3. При этом, работать с данными быстро: как на чтение, так и на запись.

ADSTERRA NETWORK

PostgreSQL

Mediatech

Что стали делать:

1. Набирать команду мечты.
2. Читать интернет.
3. Ходить на конфы: Спасибо Avito и Михаилу Тюрину за идеи архитектуры!
4. Ходить в отпуск.
5. В августе начали активную разработку(меньше 5 месяцев до релиза...).

Почему же PostgreSQL?

ADSTERRA NETWORK

PostgreSQL

Mediatech

Почему был выбран PostgreSQL:

Особенность	Зачем это нам
Бесплатность	Необходимое условие для любого стартапа
Понятная документация	Возможность быстро начать работу с базой
Наличие средств логической репликации	Londiste позволяет организовать кластер из серверов с различными ролями
Удобные типы данных:array+jsonb	Хорошо подходят для структур данных рекламного сервера

ADSTERRA NETWORK

PostgreSQL

Mediatech

Почему был выбран PostgreSQL:

Особенность	Зачем это нам
Материализованные представления	Консистентные подготовленные таблицы для быстрого чтения
Надежная потоковая репликация	Файловер+отдельный сервер под тяжелые запросы аналитиков
Партиционирование	Удобный способ организации хранения аналитических данных
PL/pgSQL	Размещение логики внутри БД позволяет убрать сетевые издержки

ADSTERRA NETWORK

PostgreSQL

Mediatech

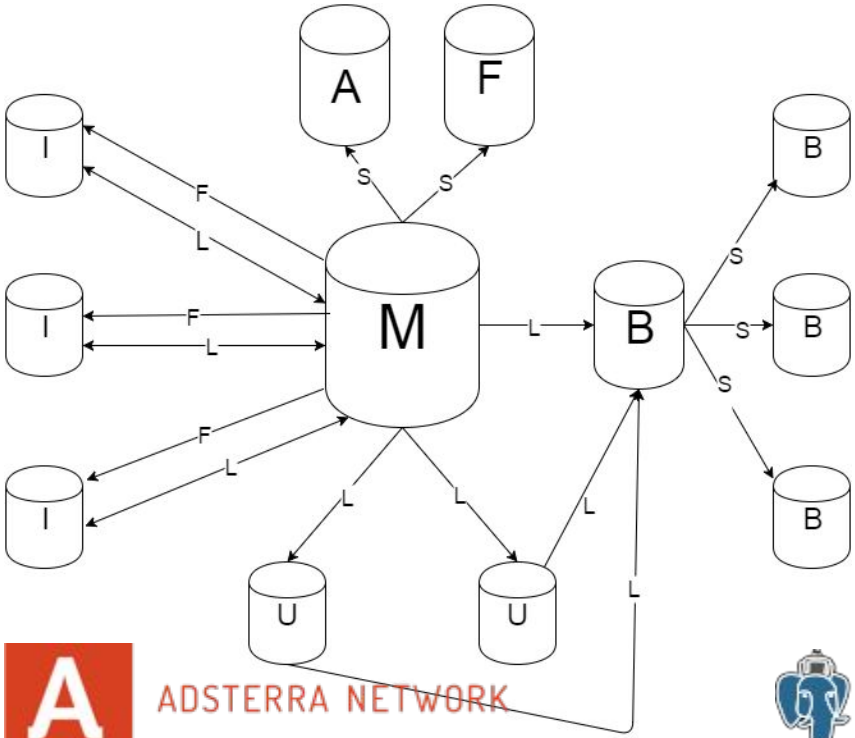
Что в итоге получилось

ADSTERRA NETWORK

PostgreSQL

Mediatech

Схема БД



Роли серверов:

I (Input) - Быстрая запись
U (User) - Быстрая запись
B (Banner) - Быстрое чтение
A (Analytic) - Работа с тяжелыми запросами
F (Failover) - Резерв Мастера
M (Master) - Центральный сервер системы

Способы передачи данных:

L - Londiste
F - Postgres_FDW
S - Streaming replication

ADSTERRA NETWORK

PostgreSQL

Mediatech

Пути передачи данных между базами

	+	-
STREAM	-из коробки -простота настройки	-реплицируется вся база -невозможно менять данные на слейве
LONDISTE	-потабличная логическая репликация	-триггеры -не развивается
POSTGRES_FDW	-переспективно для кластеров	-нагрузка на сеть

ADSTERRA NETWORK

PostgreSQL

Mediatech

Шардинг. Input сервера.

1. Шардим по id рекламного места(placement) Input сервера.
2. Ключи распределяются по шардам с помощью карты соответствия.
3. Можем в любой момент без проблем менять шард для любого placement-а.

ADSTERRA NETWORK

PostgreSQL

Mediatech

Шардинг. Banner сервера.

1. Сейчас шардинга нет - нагрузка на чтение распараллеливается по N одинаковым серверам.
2. Планируем использовать схему как и на Input сервере, чтобы уменьшить таблицы.

Шардинг. User сервера

1. Шардим по UUID.
2. Id шарда заложен в самом UUID.

SQL запросы под OLTP нагрузкой

1. Слабоумие и отвага.
2. Зато надежно!
3. И быстро!

Особенности запросов на показ рекламы:

1. Каждый запрос выдает разный (но не случайный) результат, даже если в системе ничего не меняется.
2. В системе постоянно что-то меняется.

Вывод: **кэшировать такие запросы невозможно**.

ADSTERRA NETWORK

PostgreSQL

Mediatech

SQL запросы под OLTP нагрузкой

Цифры из реальной жизни(основной запрос на показ баннера):

join 8 таблиц(самая большая **~2 млн** строк **~9Gb** размер),**>30** условий, сортировки,**500 rps** на сервер

~10 ms время выполнения

PS: Кэш запросов мы тоже используем для редко изменяемых данных вида key->value.

PL/pgsql

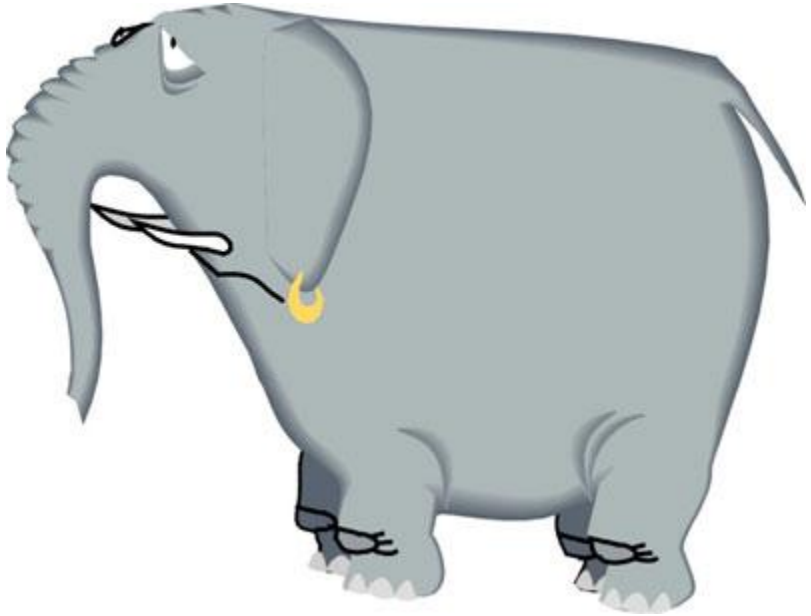
1. Изначально все стали писать на PL/pgSQL.
2. Но оверхед очень большой, под OLTP никак не годится.
3. Под OLTP даже SQL функции не годятся.
4. Используем под OLTP просто запросы, для скриптов обслуживания в основном функции SQL.
5. PL/pgSQL там, где без него никак: Динамические запросы и использование временных таблиц.

Партиционирование

1. Разбиваем статистику по дням и месяцам.
2. Старую храним на SATA, свежую на SSD.
3. Для старой индексов меньше, чем для свежей.
4. Используем как представления с UNION ALL, так и inherritance.
5. Но с ростом количества таблиц, скорость запросов падает:(

Внимание: при UNION ALL не забывать set constraint_exclusion = on

Иногда бывают проблемы...



ADSTERRA NETWORK

PostgreSQL

Mediatech

Materialized Views

Применение 1: На мастере генерить материализованные представления и реплицировать Londist-ом на Input и Banner сервера.

Materialized Views

Применение 1: На мастере генерить материализованные представления и реплицировать Londist-ом на Input и Banner сервера.
Не получилось: Londist так не умеет.

Materialized Views

Применение 1: На мастере генерить материализованные представления и реплицировать Londist-ом на Input и Banner сервера.
Не получилось: Londist так не умеет.

Вариант: Londist-ом реплицировать исходные таблицы, а вьюхи генерить из них на конечных серверах. Обновлять по триггерам.

Materialized Views

Применение 1: На мастере генерить материализованные представления и реплицировать Londist-ом на Input и Banner сервера.
Не получилось: Londist так не умеет.

Вариант: Londist-ом реплицировать исходные таблицы, а вьюхи генерить из них на конечных серверах. Обновлять по триггерам.
Не получилось 2: На слейв таблицы триггера не получилось поставить.

Materialized Views

Применение 1: На мастере генерить материализованные представления и реплицировать Londist-ом на Input и Banner сервера.
Не получилось: Londist так не умеет.

Вариант: Londist-ом реплицировать исходные таблицы, а вьюхи генерить из них на конечных серверах. Обновлять по триггерам.
Не получилось 2: На слейв таблицы триггера не получилось поставить.

Выход: Реплицировали таблицы, а представления обновляли просто раз в минуту.

Materialized Views. Проблемы обновления.

Планировалось, что конкурентный апдейт будет довольно быстрым:

"This option may be faster in cases where a small number of rows are affected."

Materialized Views. Проблемы обновления.

Планировалось, что конкурентный апдейт будет довольно быстрым:

"This option may be faster in cases where a small number of rows are affected."

В реальности, конкурентный апдейт был всегда дольше обычного и даже дольше создания новой таблицы из запроса со всеми индексами.

Materialized Views. Проблемы обновления.

Планировалось, что конкурентный апдейт будет довольно быстрым:

"This option may be faster in cases where a small number of rows are affected."

В реальности, конкурентный апдейт был всегда дольше обычного и даже дольше создания новой таблицы из запроса со всеми индексами.

В итоге стали использовать собственные “Materialized Views” на базе обычных таблиц, обновляемых по триггерам.

Materialized Views. Неожиданное применение.

Оказались удобны для работы с аналитикой:

1. Отложенные долгие запросы по статистике
2. Используются для реализации параллельных запросов к статистике

Londiste

Самая адекватная логическая репликация(как нам казалось), что была доступна.

Но давно не развивается.

И имеет кучу проблем...

Londiste. Проблемы: Триггера.

Тормозят обновления на мастере.

Londiste. Проблемы: Триггера.

Тормозят обновления на мастере.

Частичный выход: Использовать аггрегирование, уменьшать количество обновляемых строк.

Londiste. Проблемы: Отставание Слейва.

- Update идет по всем полям таблицы.
- Мешают индексы на слейве.
- При большом потоке изменений лаг необратимо увеличивается.

Londiste. Проблемы: Отставание Слейва.

- Update идет по всем полям таблицы.
- Мешают индексы на слейве.
- При большом потоке изменений лаг необратимо увеличивается.

В итоге где-то даже пробовали отказаться от Londiste и сделали свой скрипт периодического дампа таблицы на слейв.

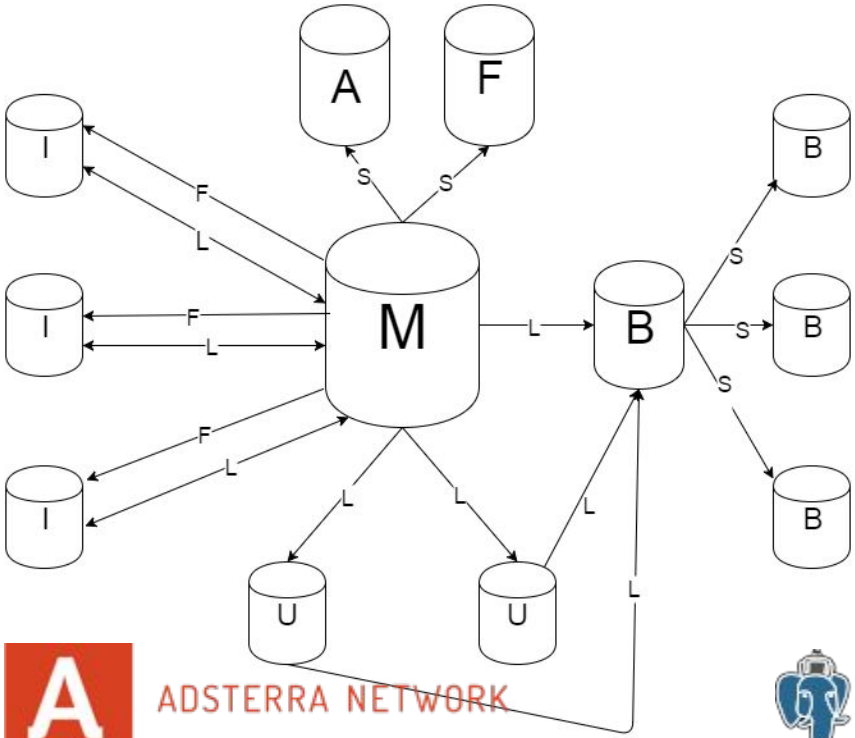
Londiste. Проблемы: Отставание Слейва.

- Update идет по всем полям таблицы.
- Мешают индексы на слейве.
- При большом потоке изменений лаг необратимо увеличивается.

В итоге где-то даже пробовали отказаться от Londiste и сделали свой скрипт периодического дампа таблицы на слейв.

Такая схема создает большие проблемы со стримовой репликой.

Схема БД



Роли серверов:

I (Input) - Быстрая запись
U (User) - Быстрая запись
B (Banner) - Быстрое чтение
A (Analytic) - Работа с тяжелыми запросами
F (Failover) - Резерв Мастера
M (Master) - Центральный сервер системы

Способы передачи данных:

L - Londiste
F - Postgres_FDW
S - Streaming replication

ADSTERRA NETWORK

PostgreSQL

Mediatech

Londiste. Проблемы: Отставание Слейва.

Решения:

- Londiste можно параллелить по таблицам.
- Следить за необходимостью индексов на слейве.
- Увеличить буфер в skytools/sqltools.py по совету Avito.

original: def __init__(self, dstcurs, tablename = None, limit = 512*1024,

hack:def __init__(self, dstcurs, tablename = None, limit = 512*1024*1024*2,

Londiste. Проблемы: Мистические.

Иногда он просто выкидывает таблицы из реплики...

Londiste. Проблемы: Мистические.

Иногда он просто выкидывает таблицы из реплики...

Решение: Мониторинг.

Stream Replication

Плюсы:

1. Встроенное средство.
2. Самый простой способ делать полную копию БД.
3. Надежно.

Stream Replication

Минусы:

1. Растет лаг при долгих транзакциях на слейве и конфликтах с мастером.
2. Медленные винты на слейве создают проблемы при большом потоке данных.

Как мы не понимаем планировщик

- Иногда костыли для планера действительно помогают

Например, использовали enable_nestloop =FALSE и join_collapse_limit (1)

- Но со временем могут начать мешать.

Конкурентные транзакции. Upserta нет...

- Делали сами через PL/pgsql - долго.
- Для OLTP нагрузки - избегаем ситуаций, когда он нужен.
- Для обычной работы - используем advisory locks.

Конкурентные транзакции. Sequence.

1. Выделяется в момент обращения, а не в начале транзакции.
2. При построении собственной системы очередей могут возникать проблемы, если это не учитывать.

Конкурентные транзакции. Sequence.

Time	Function 1	Function 2
00:00:01	запуск	
00:00:02	что-то считает	запуск
00:00:03	что-то считает	что-то считает
00:00:04	что-то считает	вставляет в очередь
00:00:05	вставляет в очередь	

ADSTERRA NETWORK

PostgreSQL

Mediatech

intarray

Плюсы

- Удобно для фильтрации
- Работают быстро с GIN/Gist индексами
- Позволяют **сильно** уменьшать таблицы

intarray

Позволяют сильно уменьшать таблицы:

1) relations_map

entity1 integer NOT NULL, entity2 **integer[]** NOT NULL +
PRIMARY KEY (entity1) +
INDEX relations_map_idx ON relations_map USING gist (entity2 gist__intbig_ops)

rows:**179181**, size(with indexes): **1785 MB**, size(w/o indexes): **58 MB**

2) relations_map_unnest

entity1 integer NOT NULL, entity2 **integer** NOT NULL +
INDEX relations_map_unnest_idx ON relations_map_unnest (entity1 ,entity2);

rows: **408712632** (408 mln, Karl!), size(with indexes):**22 GB**, size(w/o indexes):**14 GB**

ADSTERRA NETWORK

PostgreSQL

Mediatech

intarray

Минусы:

- "The potential for bloat in non-B-tree indexes has not been well researched."
- При частых апдейтах лучше Gist индексы
- Bloat самой таблицы пока непонятно как решить

Автовакум и Вакуум

- Даже при самых жестких настройках автовакуума, он не работает как надо на таблицах с интенсивными обновлениями. Особенно при использовании массивов.
- Приходится делать vacuum full: exclusive lock, работает долго.

Автовакуум и Вакуум

- Даже при самых жестких настройках автовакуума, он не работает как надо на таблицах с интенсивными обновлениями. Особенно при использовании массивов.
- Приходится делать vacuum full: exclusive lock, работает долго.

Костыльное решение: “ручной вакуум” - пересоздаем таблицы.

Не подходит для таблиц в реплике...

Немного экспериментов

ADSTERRA NETWORK

PostgreSQL

Mediatech

PL/Python

1. Раньше думал: “зачем весь этот зоопарк?”
2. Оказалось - есть зачем!
3. PL/Python позволяет осуществлять “параллельные” запросы к статистике!

Почти MPP

Аггрегированная статистика начала плохо справляться:

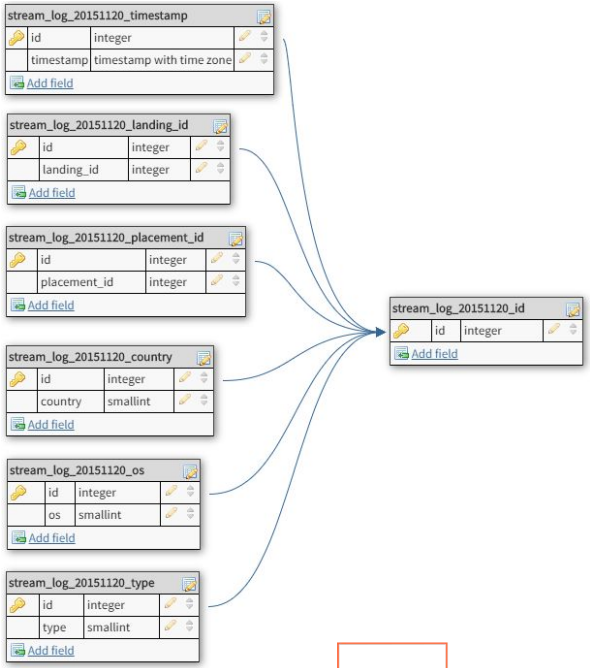
1. Менеджерам надо добавить новые столбцы для статистики. Добавить их в старые таблицы - это боль.
2. С каждым новым столбцом аггрегирование стремится к нулю.
3. Надо придумать что-то, чтобы расширяться без границ...

Почти MPP

1. Спасибо Николаю Голову(Avito) и его докладам про HP Vertica
2. Но мы не хотим HP Vertica…
3. Сделаем что-то подобное в Постгресе!

Почти MPP: Anchor Modeling

aggregated_stats		
a_date	date	
a_hour		
landing_id	integer	
placement_id	integer	
country		
os		
requests	integer	
impressions	integer	
Add field		



ADSTERRA NETWORK

PostgreSQL

Mediatech

Почти MPP

1. Шардим данные на несколько серверов.
2. С помощью PL/Python асинхронно создаем materialized view на каждом шарде в N потоков. Получаем параллельное создание агрегированных данных.
3. Собираем все данные на мастер через FDW и делаем окончательную аггрегацию.
4. Радуемся пол часа.
5. Но, конечно, это не идеал...

Почти MPP

1. Столбцы не сжимаются, как в колоночных базах.
2. Проблемы с использованием больших объемов shared memory.
3. Данные с шардов выгружаются последовательно.
4. Агрегированные таблицы все равно надо делать.
5. Продолжаем думать..

Наши мечты

ADSTERRA NETWORK

PostgreSQL

Mediatech

Что нам надо для счастья

Я не хочу ни о чем думать, я хочу платье...и нормальную логическую реплику:

1. Без триггеров.
2. Без лагов и зависаний на любых объемах.
3. Простую в настройке и поддержке.

pglogical(http://2ndquadrant.com/en/resources/pglogical/)...?

ADSTERRA NETWORK

PostgreSQL

Mediatech

Что нам надо для счастья

Я не хочу ни о чем думать, я хочу платье… и способ шардить мастер!

1. Менеджеров и клиентов все больше.
2. 1го сервера все меньше хватает.
3. Надо много думать над шардингом мастер-сервера...но не хочется. Но придется.

???

Всем спасибо!

Особенно:

- Жене и детям за то, что помнят
- Коллегам за то, что терпят

Ответы? Предложения?


email: yury@adsterra.com


skype: yury.adsterra


Telegram: https://telegram.me/YuryAdsterra


https://github.com/newmediatech

ADSTERRA NETWORK

PostgreSQL

Mediatech