PostgreSQL-Consulting

data egret

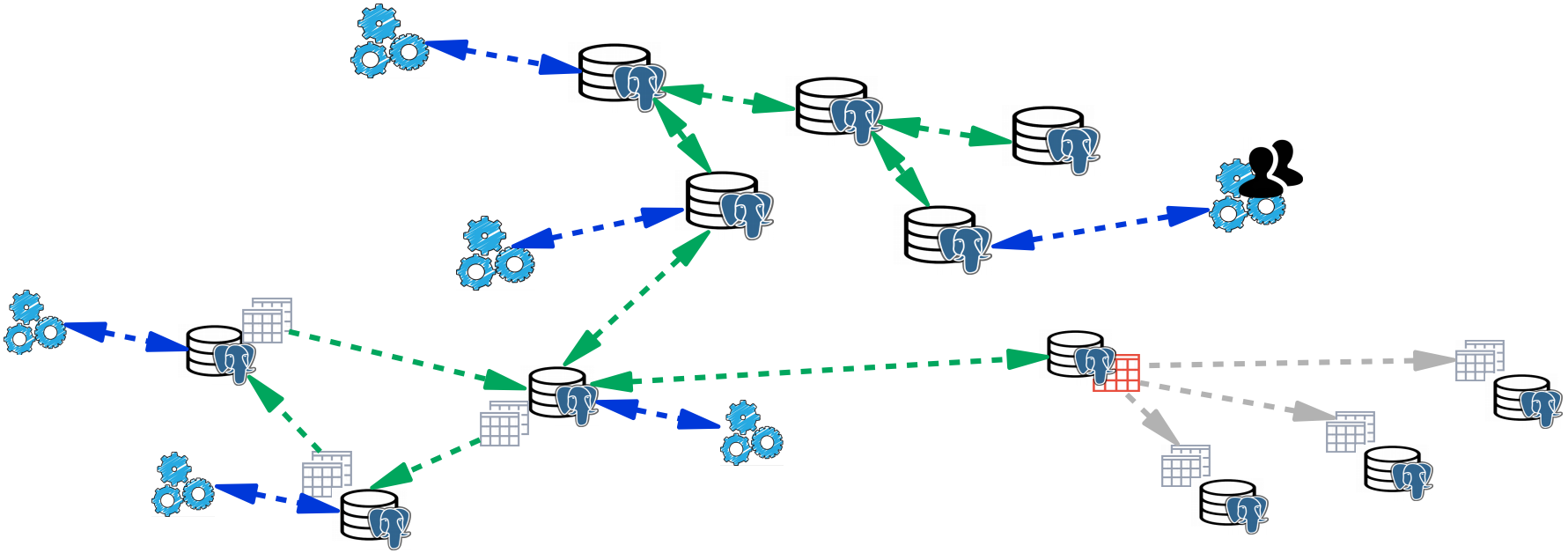
PostgreSQL Scaling Usacases

Alexey Lesovsky
alexey.lesovsky@dataegret.com

Кто докладчик?

PostgreSQL DBA, System administrator

Data Egret — PostgreSQL consulting&support



О чем доклад?

PostgreSQL Scaling Usecases

- Проблемы приводящие к мысли о масштабировании
- Варианты решения
- Суммаризация опыта

О чем доклад?

PostgreSQL Scaling Usecases

- Streaming replication
- Declarative Partitioning
- Built-in «sharding»

Чего НЕ будет в докладе

Как планировать/рефакторить схему под шардинг

Историй успеха о распиле на шарды

Рассказов об использовании Greenplum, Timescale, Citus, etc...

Когда приходят мысли о масштабировании?

Недостаточно текущих ресурсов:

- Вычислительные ресурсы (CPU, RAM)
- Ресурсы хранения (Disk space)
- Отказоустойчивость на уровне датацентров

Когда НЕ нужно думать о масштабировании

Все данные умещаются в RAM

Проект на стадии MVP

Когда НЕ нужно думать о масштабировании

Все данные умещаются в RAM

Проект на стадии MVP

Время на разработку/реализацию

Время на поддерживание

Увеличение общей сложности

Преждевременная оптимизация

Масштабирование нужно

Исчерпали ресурс оптимизации запросов

Исчерпали ресурс схемы данных

Исчерпали инфраструктурные ресурсы (hardware, software)

Самый простой и частый способ масштабирования

Streaming replication

Всегда есть в качестве hot standby

Доступен для read-only трафика

Часто недооценен

Самый простой и частый способ масштабирования

Streaming replication

Всегда есть в качестве hot standby

Доступен для read-only трафика

Часто недооценен

Лаг репликации

Балансировка трафика

Балансировка

Infrastructure-based

Application-based

Балансировка

Infrastructure-based

DNS, Consul, Etcd

Application-based

Haproxy, Confd, Consul-Template

Keepalived, VRRP

...

Балансировка

Infrastructure-based

Application-based

Volatile master

Переустановка соединения

Повтор запроса

Circuit breaker, back-off

Где и как используем

Почти всегда, у всех есть hot standby

Реже отдельный read трафик, аналитические запросы

Резервное копирование

Недостатки и особенности

Лаг репликации (вакуумы, bulk операции, остановка wal replay)

Recovery conflicts — отмена долгих запросов на реплике

Bloat на мастере, см. *hot_standby_feedback*

Что в итоге

Мастер на запись

Реплики на чтение → масштабирование нагрузки

Запись по-прежнему ограничена мощностью одного узла…

Заход с другой стороны — много данных

Declarative partitioning

Оперативные данные

Холодный архив

Заход с другой стороны — много данных

Declarative partitioning

Оперативные данные

Холодный архив

Хорошо для неизменяемых данных

Легко управлять местом

Архив можно унести

Foreign data wrappers

Где и как используем?

Пример из практики:

- Исторические данные — jobs, events, stats…
- Оперативные данные ~7.6TB (таблицы+индексы)
- Данные в Amazon S3 ~1.1TB (zip, csv)
- Архив содержит данные с 2015 года

Какие проблемы?

Cron-скрипты пакуют и сгружают данные в S3 (tar, pbzip2, psql, awscli)

Cron-скрипты удаляют старые партиции (detach, drop table)

Руками подключение архивных партиций при необходимости

Когда партиционирования недостаточно

Declarative Partitioning

Foreign Data Wrappers

Scale-up → в случае Postgres это единственный подходящий вариант

Путь для сильных духом

Несвязанные репликацией узлы

Мастер узел держит partitioned таблицу

Leaf узлы доступны на запись

Каждая таблица это foreign table

Какие проблемы есть на пути

Нет инструментов которые позволяют управлять подобными схемами

По большей части все делается руками (можно и автоматизировать)

Издержки для high availability

Не самая лучшая производительность

3x data nodes, 1x pgbench node — 2GB RAM, 2xCPU

In-memory dataset

```
CREATE TABLE measurements (
    ts timestamptz not null,
    peaktemp int default 1,
    unitsales int
) PARTITION BY RANGE (logdate);
```

Performance

```
SELECT peaktemp FROM measurements WHERE ts = :ts
```

Latency, baseline: 0.57ms

Latency, sharding: 2.91ms ~6x

Performance

```
SELECT peaktemp FROM measurements WHERE ts BETWEEN :ts AND :ts + 1000
```

Latency, baseline: 2.82ms

Latency, sharding: 26.15ms ~9x

Performance

```
SELECT sum(peaktemp) FROM measurements WHERE ts BETWEEN :ts AND :ts + 1000
```

Latency, baseline: 2.25ms

Latency, sharding: 24.40ms ~10x

Performance

```
UPDATE measurements SET peaktemp = :pt WHERE ts = :ts
```

Latency, baseline: 0.53ms

Latency, sharding: 2.42ms ~5x

Какие еще варианты?

Postgres-XC/XL, CitusDB

- Координаторы, transaction менеджеры, 2 phase commit
- Местами сложно

Cassandra, Riak, etc...

- Масштабирование на запись как базовая функция
- Другие недостатки присущие этим системам

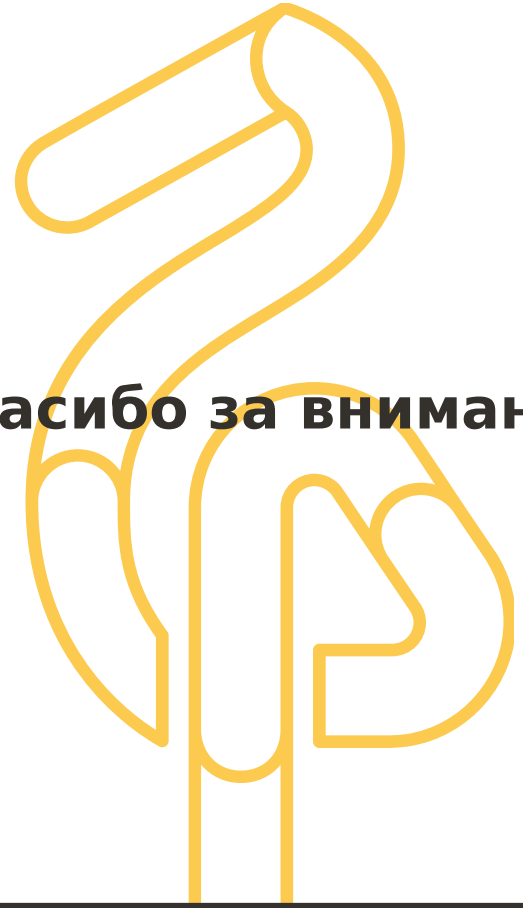
Итого

Postgres хорошо масштабируется на чтение

Нет инструментов упрощающих масштабирование

Streaming replication (physical, logical)

Declarative partitioning, Foreign Data Wrappers



Спасибо за внимание

Алексей Лесовский, alexey.lesovsky@dataegret.com

PGDAY' RUSSIA 20

5-я конференция по базам данных PostgreSQL

10 ИЮЛЯ 2020

CROWNE PLAZA АЭРОПОРТ, САНКТ-ПЕТЕРБУРГ