Опыт использования PostgreSQL

В проектах UIS, CoMagic

Дмитрий Белобородов

Технический директор

Пару слов о нас

UIS - виртуальная АТС

CoMagic - сервис сквозной аналитики с интегрированными каналами коммуникаций

Немного истории

- ✓ Начали с 7.3
- ✓ Переход на 8
- ✓ Уход от default конфига
- ✓ Перевод CRM из MSSQL (весь мир на postgres)
- ✓ Появился второй разработчик БД
- ✓ RAID и батарейка (производительность дисковой системы)
- ✓ Сформировалась команда базистов (изменились правила работы с БД)
- ✓ Настройка ОС
- ✓ Партиционирование больших таблиц (средствами наследования)
- ✓ Переход на SSD
- ✓ Внедрение прокси уровня запросов (pgbouncer + plexor)

CoMagic

postgresql.conf

Ставьте сразу больше, потом если закончатся на ходу не увеличить!

```
max_connections = 900
max_files_per_process = 4096
```

Оперативка подешевела

```
shared_buffers = 24GB
huge_pages = on
work_mem = 128MB
maintenance_work_mem = 2GB
```

SSD

```
effective_io_concurrency = 4
random_page_cost = 1.0
```

postgresql.conf

В postgresql 2 беды - checkpoint

```
bgwriter_delay = 20ms
bgwriter_lru_maxpages = 800
bgwriter_lru_multiplier = 8.0
checkpoint_timeout = 15 min
max_wal_size = 12GB --потерялся при переезде на 9.6
checkpoint_completion_target = 0.9
checkpoint_warning = 1min
```

… и autovacuum

```
autovacuum_max_workers = 16
autovacuum_naptime = 20s
autovacuum_vacuum_cost_delay = 2ms
```

CoMagic

postgresql.conf

защита от залипаний при использовании распределенных транзакций

```
lock_timeout = 120000
```

защита от неадекватных запросов

```
statement_timeout = 3600 (alter role postgres set statement_timeout = '90s')
```

баланс между быстродействием и надежностью

```
synchronous_commit = off (alter database billing set synchronous_commit = 'on')
```

статистика в оперативке!

```
stats_temp_directory = '/var/lib/pgsql/pg_stat_tmp'
```

CoMagic

CPU

Модели процессоров, которые мы пробовали и используем:

- ✓ Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
- ✓ Intel(R) Xeon(R) CPU E5-2695 v3 @ 2.30GHz
- ✓ **Intel(R) Xeon(R) CPU E5-2667 v4 @ 3.20GHz**

Диски

RAID: 10

disk count: 8

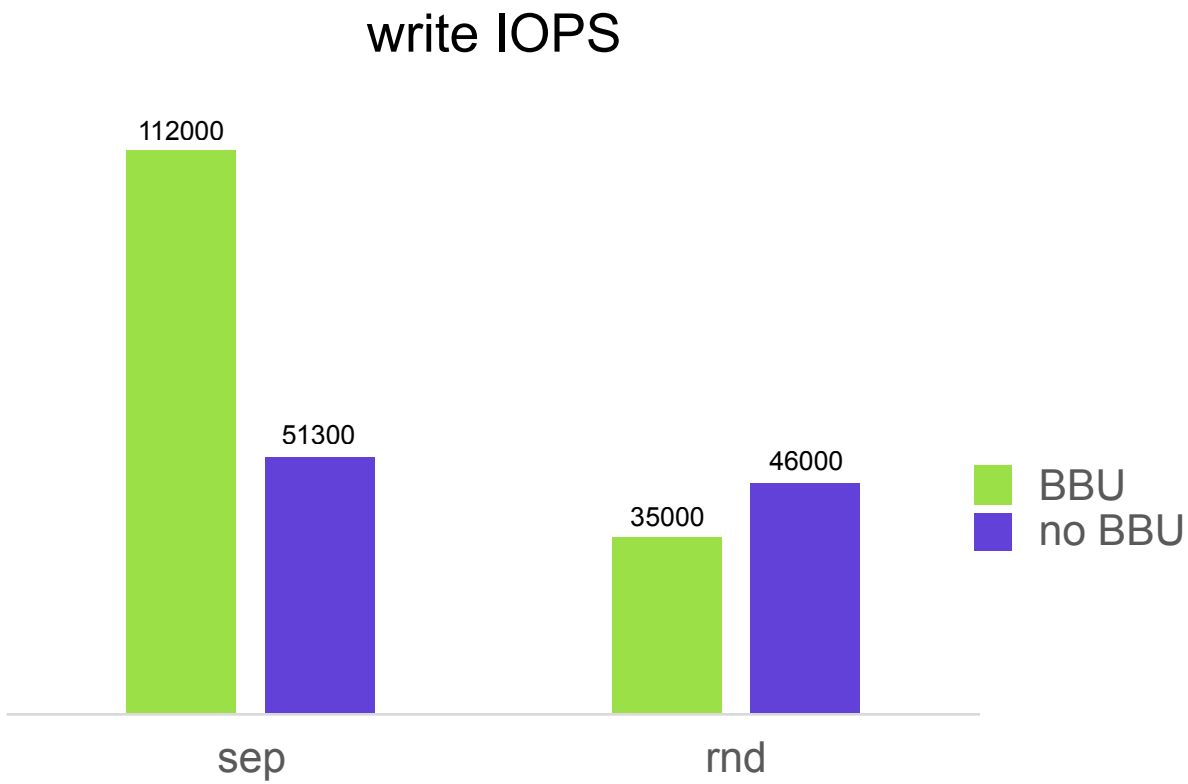
disk model: INTEL SSDSC2BA012T4

controller: RMS3CC080

blocksize: 8192

iodepth: 32

ioengine: libaio, direct, unbuffered



Диски

disk count: 12

disk model: INTEL SSDSC2BA012T4

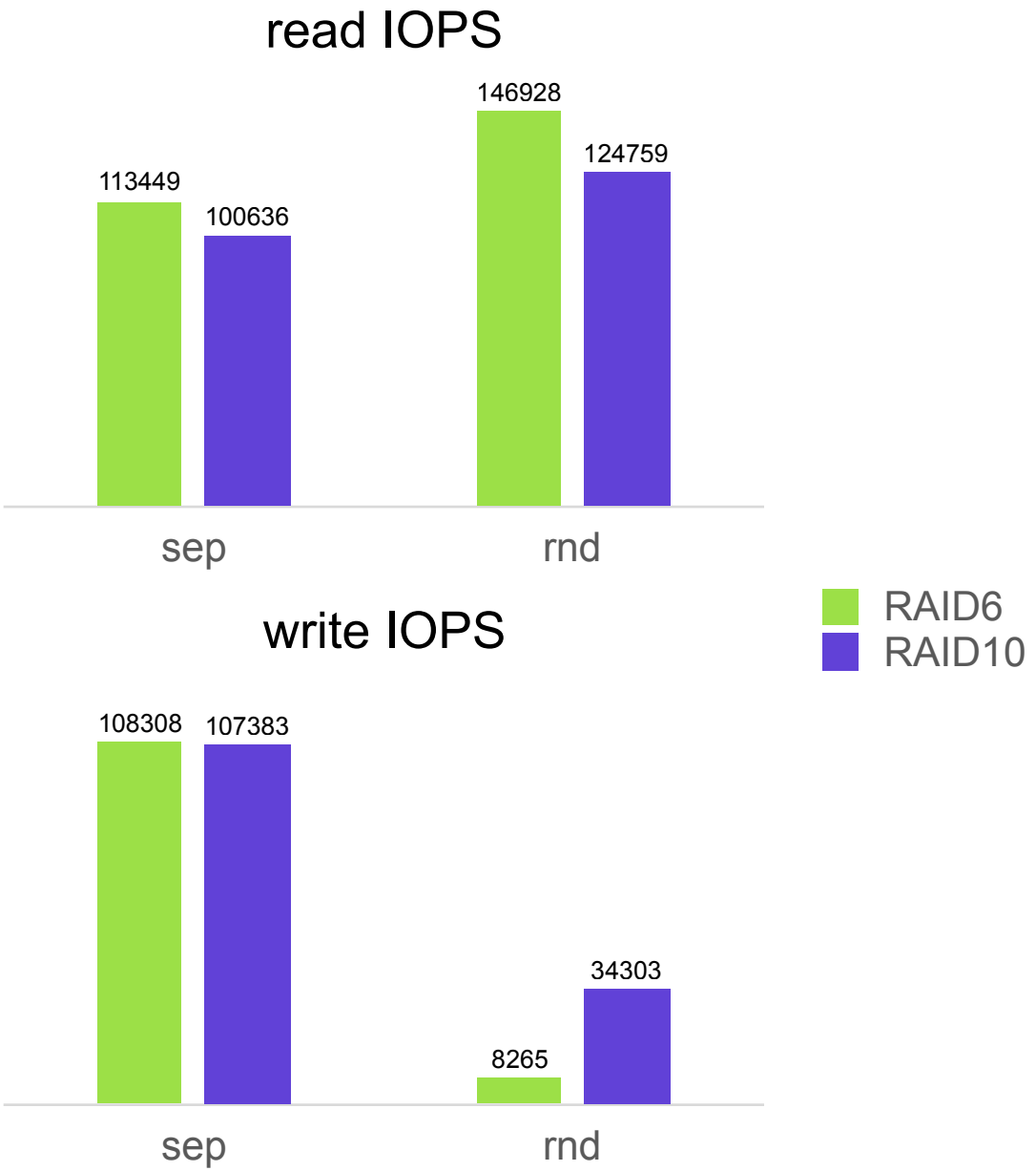
controller: RMS3CC080

blocksize: 8192

iodepth: 32

ioengine: libaio, direct, unbuffered

bbu: yes, cache=WB



ОС - настройка

✓ Настройка BIOS

NUMA Optimization := Disable
CPU FAN profile := Performance

✓ tuned

[cpu]
force_latency=1
governor=performance
energy_perf_bias=performance

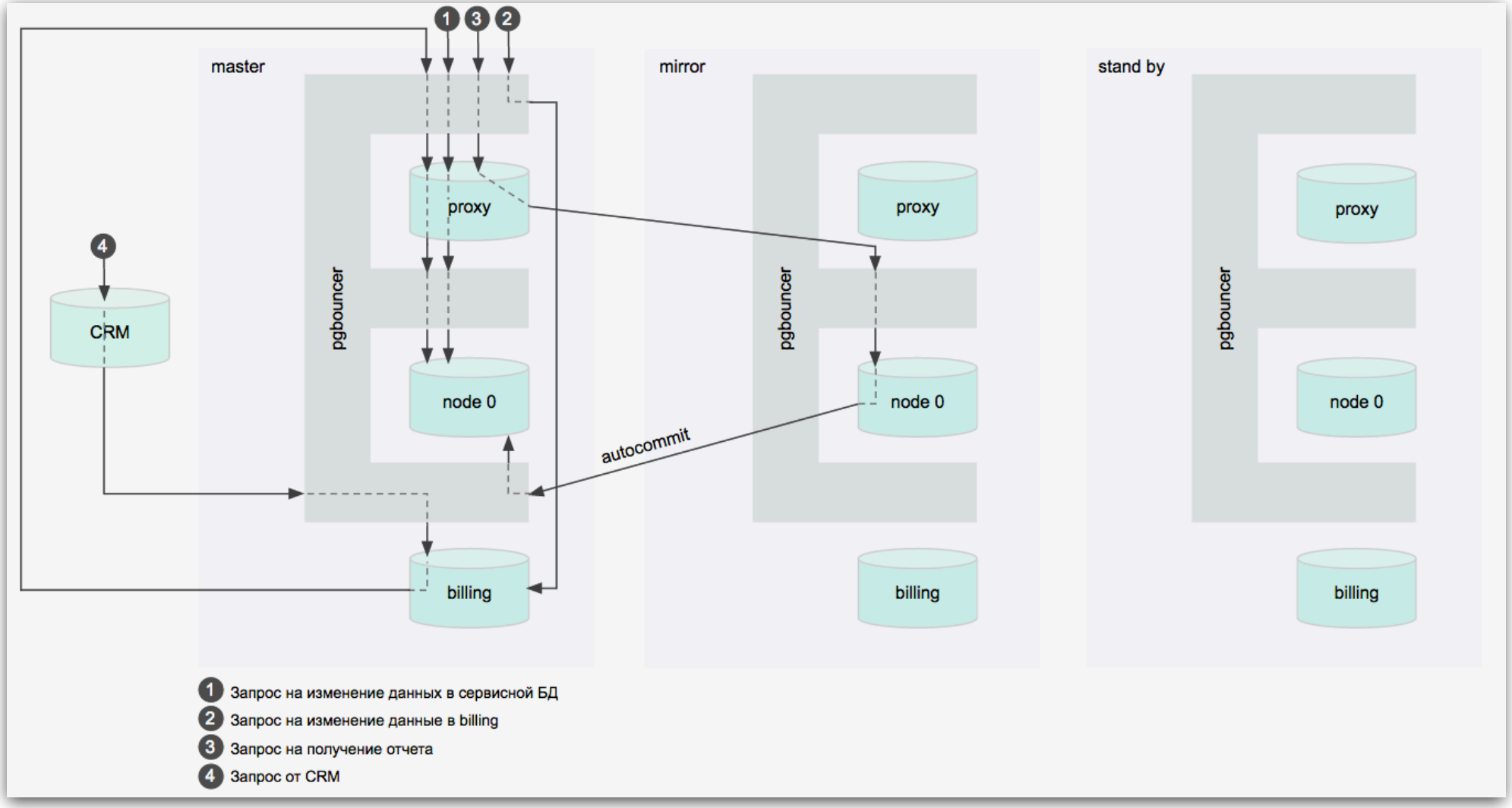
[sysctl]
min_perf_pct=100
vm.swappiness = 1
vm.dirty_background_bytes = 536870912 (половина кэша RAID)
vm.dirty_writeback_centisecs = 250
vm.dirty_expire_centisecs = 1000
vm.nr_hugepages = 12632 (измеряется в кол-ве страниц, по 2 МБ каждая)

CoMagic

Архитектура

- ✓ Выкинули fdw, почему:
  - ✓ deadlock (распределенные запросы)
  - ✓ блокировки по fk (для обеспечения уровня доступа serializable)
  - ✓ не оптимальность (какой будет план запроса, заранее неизвестно)
  - ✓ фильтрация на стороне клиента
  - ✓ нельзя вызывать функции (есть костыль через view, но не будет return)
- ✓ plexor - https://github.com/comagic/plexor
  - ✓ есть поддержка транзакций (plproxy не поддерживает)
  - ✓ поддерживаются настраиваемые уровни изоляции
  - ✓ поддерживается режим autocommit = autonomous transaction
  - ✓ поддерживаются savepoint
  - ✓ proxy - есть возможность на лету менять маршруты вызова функций
  - ✓ sharding - есть распределять запросы по разным нодам

Маршрутизация запросов



Разработка

Мы храним в репозитории как структуру БД, так и скрипты миграции

```
data/
patches/
    release-va_r5.2.2.sql
proxy/
schema/
    analytics/
        analytics.sql
        functions/
        tables/
        triggers/
        types/
tests/
```

https://github.com/comagic/pg_export

Разработка

1. Создание стенда, создание ветки в репозитории
2. Собственно, разработка
    1. изменение структуры
    2. внесение изменения в скрипт миграции везде,
       где можно используем ссылки (\i analytics/functions/get_ac.plpgsql)
3. Подготовка к релизу (вливаем stable в ветку релиза)
4. Релиз
    1. накатывание скрипта миграции
    2. закрытие ветки
    3. вливание ветки в stable
    4. удаление скриптов миграции (в stable только структура)

CoMagic

Спасибо за внимание

Вопросы?

Дмитрий Белобородов

d.beloborodov@uiscom.ru