Настройка PostgreSQL для высокой производительности

Илья Космодемьянский
ik@postgresql-consulting.com

PostgreSQL-Consulting.com

План

- Производительность PostgreSQL

План

- Производительность PostgreSQL
- С чего начать?

План

- Производительность PostgreSQL
- С чего начать?
- Базовые настройки

План

- Производительность PostgreSQL
- С чего начать?
- Базовые настройки
- Наблюдение за базой

План

- Производительность PostgreSQL
- С чего начать?
- Базовые настройки
- Наблюдение за базой
- Тонкая настройка

План

- Производительность PostgreSQL
- С чего начать?
- Базовые настройки
- Наблюдение за базой
- Тонкая настройка
- Анализ запросов (Брюс и Максим расскажут подробно)

Что такое производительность базы данных

- Скорость ответа клиентам
- Пропуская способность
- Минимизация даунтайма

Что такое производительность базы данных

- Скорость ответа клиентам
- Пропуская способность
- Минимизация даунтайма

Обеспечение высокой доступности vs highload

- Если есть возможность избегнуть highload - лучше избегнуть
- Тысяч серверов и Big Data - тоже
- Know your data

С чего начать?

Возможны варианты

- Новый проект, вы отвечаете за базу с нуля
- Вы пришли в проект, где есть или возможны проблемы с PostgreSQL

Вы готовите базу данных с нуля

Последовательность действий

- Собрать бизнес-требования
- Исходя из этого рассчитать объем данных и предполагаемую нагрузку
- Выбрать железо, которое "потянет"эту нагрузку и вместить эти объемы
- Все настроить
- Развернуть проект в продакшн

Вы готовите базу данных с нуля

Последовательность действий

- Собрать бизнес-требования
- Исходя из этого рассчитать объем данных и предполагаемую нагрузку
- Выбрать железо, которое "потянет"эту нагрузку и вместить эти объемы
- Все настроить
- Развернуть проект в продакшн
- Откинуться в кресле и ждать наступления коммунизма через 20 лет

Вы готовите базу данных с нуля

В жизни все сложнее

- Бизнес-требования могут быть (мягко говоря) противоречивыми и изменчивыми
- Предугадать нагрузку заранее возможно только приблизительно
- Требуемое железо могут и не купить
- Программисты могу оказаться "с идеями"
- После запуска проекта нагрузка может меняться непредсказуемо

Вы готовите базу данных с нуля

Checklist

- Оценить бизнес-требования на предмет адекватности
- Оценить объемы и нагрузку на предмет влезет-не влезет база в память одного сервера
- Оценить критичность потери данных
- Выбрать максимально мощное железо из возможных вариантов
- Произвести базовую настройку ОС и PostgreSQL
- В ходе тестирования, запуска проекта и после его запуска - наблюдать за нагрузкой
- **Избегайте преждевременного масштабирования!**

Вы пришли в проект, где есть или возможны проблемы с PostgreSQL

Все примерно так же, но может быть и хуже

- При изначальном планировании допущены ошибки
- Проблемы нужно решить вчера
- Железо не оптимальное, о покупке нового можно говорить только в конце третьего квартала
- Документация неполная, приложение написано давно и плохо

Вы пришли в проект, где есть или возможны проблемы с PostgreSQL

Checklist

- **Начните с проверки резервного копирования!**
- Всё-ли в порядке с базовыми настройками ОС и PostgreSQL?
- Что показывает мониторинг?
- Локализация проблем
- Тонкая настройка $\rightarrow$ наблюдение за результатами $\rightarrow$ тонкая настройка
- Анализ top медленных запросов

Выбор сервера для PostgreSQL

Железо

- Дешевый RAID контроллер - зло, уж лучше software RAID
- RAID контроллер должен быть с "батарейкой"
- Производители: megaraid или perc - OK, а например HP или ARECA - не всегда
- За батарейкой надо следить

Выбор сервера для PostgreSQL

Железо - Диски

- 2,5"SAS (вполне бывает 15K) - seek в 2-3 раза быстрее чем 3,5"
- Не все SSD одинаково полезны: enterprise level Intel p3700 vs desktop-level Samsung
- Использовать только SSD для PostgreSQL означает быть готовыми к ряду проблем
- RAID 1+0
- если нет хороших дисков - $synchronous_commit \rightarrow off$

Выбор сервера для PostgreSQL

Операционная система

- Linux, если нет специальных соображений
- Новые ядра (3.3.* а то и 3.16.*) - хорошо изучить, нет-ли в конкретном ядре PostgreSQL-специфических багов
- Дистрибутивы: Debian, Ubuntu, Centos, SuSe, RHEL(версия ядра требует проверки)

Выбор сервера для PostgreSQL - настройка железа

Настройка RAID

- Батарейка - присутствует и исправна
- cache mode → write back
- io mode → direct
- Disk Write Cache Mode → disabled

Выбор сервера для PostgreSQL - настройка железа

BIOS

- numa → off или $vm.zone\_reclaim\_mode = 0$
- То есть по уму лучше $numactl -- interleave = all\ /etc/init.d/postgresql\ start$
- Ещё лучше чтобы база разруливала сама, но это светлое будущее

Выбор сервера для PostgreSQL - настройка OS

Huge pages - что это такое?

- Сервера с большим количеством памяти (128-256Gb) теперь стоят разумных денег
- Держать базу в памяти хорошо (любую хорошо, PostgreSQL совсем хорошо)
- По умолчанию память выделяется по 4kB - так быстрее
- ОС транслирует виртуальные адреса в физические, результат кэшируется в Translation Lookaside Buffer (TLB)
- $\frac{1Gb}{4kB} = 262144$ и 64-битная адресация - на больших объемах памяти оверхэд и низкая эффективность TLB
- Выделяем память большими порциями

Выбор сервера для PostgreSQL - настройка OS

Huge pages в PostgreSQL

- *vm.nr_hugepages* = 3170 (sysctl, ядро должно поддерживать)
- До 9.2 включительно - через библиотеку hugetlb
- 9.3 улучшена работа с shared memory с помощью mmap - huge pages не работают
- 9.4+ *huge_pages* = *try*|*on*|*off* (postgresql.conf)
- *try* - дефолт, не получилось, не используем
- Сейчас работат только на linux, *try* на FreeBSD - OK

Выбор сервера для PostgreSQL - настройка OS

Filesystem

- barrier=0, noatime
- xfs или ext4

Выбор сервера для PostgreSQL - настройка OS

dirty

- По умолчанию $vm.dirty\_ratio = 20$ $vm.dirty\_background\_ratio = 10$
- Более разумные значения $vm.dirty\_background\_bytes = 67108864$ $vm.dirty\_bytes = 536870912$
- Если нет кэша на RAID - разделить на 4

Выбор сервера для PostgreSQL - настройка OS

Swap

- *vm.swappiness* = 0
- На больших объемах памяти swap все равно почти бесполезен

Как будет выглядеть инсталляция?

Необходимый минимум

- Основной сервер
- Hot-standby
- pgbouncer $*$ 2

Как будет выглядеть инсталляция?

Необходимый минимум

- Основной сервер
- Hot-standby
- pgbouncer $*$ 2

Желательно

- Stage
- Сервер для валидации бэкапов
- Отдельные сервера для мониторинга и средств автоматизации

Базовые настройки postgresql.conf

- *shared_buffers*
- WAL
- autovacuum

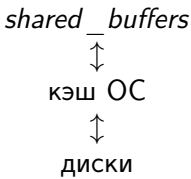
Базовые настройки postgresql.conf

shared_buffers

- 25% RAM - хорошая отправная точка
- 75% - может быть хорошо если база помезается в память
- большие объемы *shared_buffers* - только при хорошем RAID

Базовые настройки postgresql.conf

WAL



Базовые настройки postgresql.conf

WAL

- *wal_buffers* (768kB → 16Mb)
- *checkpoint_segments* (3 - чекпойнт каждые 48Mb → 256 - 4Gb)
- *checkpoint_timeout* (что первое наступит)
- *checkpoint_completion_target* (для распределения чекпойнтов)
- *pg_stat_bgwriter* system view

Пусть bgwriter работает (это уже продвинутая настройка)

```
postgres=# select name, setting, context, max_val, min_val  from pg_settings where name ~ 'bgwr';
  name            | setting | context | max_val | min_val
--------------------------+---------+---------+---------+---------
 bgwriter_delay           | 200     | sighup  | 10000   | 10
 bgwriter_lru_maxpages    | 100     | sighup  | 1000    | 0
 bgwriter_lru_multiplier | 2       | sighup  | 10      | 0
(3 rows)
```

Autovacuum - настройка

```
postgres=# select name, setting, context  from pg_settings where category ~ 'Autovacuum';
                 name                  |  setting  |  context
---------------------------------------+-----------+------------
 autovacuum                            | on        | sighup
 autovacuum_analyze_scale_factor       | 0.1       | sighup
 autovacuum_analyze_threshold          | 50        | sighup
 autovacuum_freeze_max_age             | 200000000 | postmaster
 autovacuum_max_workers                | 3         | postmaster
 autovacuum_multixact_freeze_max_age   | 400000000 | postmaster
 autovacuum_naptime                    | 60        | sighup
 autovacuum_vacuum_cost_delay          | 20        | sighup
 autovacuum_vacuum_cost_limit          | -1        | sighup
 autovacuum_vacuum_scale_factor        | 0.2       | sighup
 autovacuum_vacuum_threshold           | 50        | sighup
(11 rows)
```

Базовые настройки *pg_hba.conf*

- Из коробки - хороший уровень безопасности
- Не надо открывать широко
- Standby может внезапно стать мастером!

Базовые настройки pgbouncer

- $pool\_mode = transaction$ если возможно
- Помним о сессионных переменных
- $pool\_size = (10|20|30)$
- $max\_client\_connections = (1000|10000)$

Мониторинг

- must have, какой больше нравится, мы рекомендуем обычно zabbix
- загрузка диска - утилизация (iostat -d -x 1, последняя колонка, %*util*)
- состояние батарейки

Еще немного продвинутых настроек

- если есть репликация - *hot_standby_feedback = on*
- избегайте настроек оптимизатора, но иногда нужно, например *join_collapse_limit*

Вопросы?

ik@postgresql-consulting.com