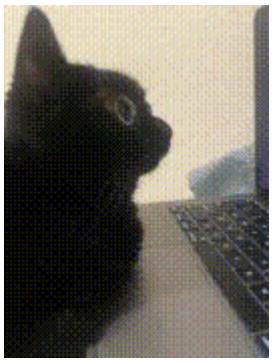
Производительность

Статья с уклоном в оптимизацию для 1С, но скорее всего и для других случаев подойдет.



https://sematext.com/blog/postgresql-performance-tuning/

Если база постоянно используется, для диагностики будет проще поднять еще одну СУБД на каком-нибудь отдельном хосте, хоть на виртуалке, и крутить там параметры как угодно. Замечая разницу между изменением параметров можно соотнести результаты с боевым хостом.

Как вообще работает постгря?

> TODO

Куда смотреть на мониторинге? Как мониторить? Что

мониторить?

Понятное дело, надо мониторить саму железку. Сеть, проц, озу, нагрузку на диски. Также надо мониторить параметры постгреса. Советую настроить мониторинг тем же заббиксом, там из коробки мониторится много полезных параметров.

Мониторим запросы, таблицы, индексы

Запросы:

```
\l+ -- Список баз
\c test -- Переключиться на базу
\dt+ -- Список таблиц и их размеры
\d table -- Посмотреть структуру таблицы и ее индексы
\di+ -- Список индексов и их размеры
```

Синтаксис SQL запросов (`select ... from ... where`) у постгри практически такой же, как у мускуля.

Смотрим на статистику в моменте

https://www.craigkerstiens.com/2012/10/01/understanding-postgres-performance/

Посмотрим размер таблиц:

```
SELECT
  table_name,
  pg_size_pretty( pg_total_relation_size(quote_ident(table_name))),
  pg_total_relation_size(quote_ident(table_name))
FROM
  information_schema.tables
WHERE
  table_schema = 'public' and table_catalog = 'URDATABASE'
ORDER BY
  pg_total_relation_size(quote_ident(table_name)) DESC;
```

Если есть что-то сильно жирное - надо партиционировать. Я считаю, что стоит разбивать таблицы больше 10 Гб. В случае с 1С не пугайтесь большого (около 9-10к) количества таблиц - это норма.

- Cache Hit Rate

Будет прекрасно, если hit rate будет около 99%. Смотреть вот так (предварительно зайти в нужную базу):

```
SELECT
  sum(heap_blks_read) as heap_read,
  sum(heap_blks_hit)  as heap_hit,
  sum(heap_blks_hit) / (sum(heap_blks_hit) + sum(heap_blks_read)) as ratio
FROM
  pg_statio_user_tables;
```

- Index Usage

Статистика использования индексов. Заходим в базу и выполняем:

```
SELECT
  relname,
  100 * idx_scan / (seq_scan + idx_scan) percent_of_times_index_used,
  n_live_tup rows_in_table
FROM
  pg_stat_user_tables
WHERE
    seq_scan + idx_scan > 0 and n_live_tup > 10000
ORDER BY
  n_live_tup DESC;
```

While there is no perfect answer, if you’re not somewhere around 99% on any table over 10,000 rows you may want to consider adding an index.

- Index cache hit rate

Должно быть близко к 99%. Заходим в базу и выполняем:

```
SELECT
  sum(idx_blks_read) as idx_read,
  sum(idx_blks_hit)  as idx_hit,
```

```
  (sum(idx_blks_hit) - sum(idx_blks_read)) / sum(idx_blks_hit) as ratio
FROM
  pg_statio_user_indexes;
```

Если с индексами беда - надо их создавать

> Создание индексов нарушает лицензионное соглашение 1С (WTF?)

> TODO: Как это делать?

Читаем графики

Советую настроить мониторинг в заббиксе: https://www.zabbix.com/integrations/postgresql.
Или хотя бы сделать аналогичный, хотя бы по количеству висящих запросов в транзакции и кол-ву соединений.

Разное

- Connections sum: Idle in transaction

Кол-во соединений, которые висят в транзакции. Должно быть практически всегда на 0. Если есть долгие соединения, значит что-то долго выполняется (спасибо кэп).

- Connections sum (Active/Idle/Total)

Думаю тут и так понятно все.

bgwriter

- Checkpoint write/sync time

При бездействии должно быть в районе 5 сек. Когда идет активная работа с базой - значение уменьшается, чекпоинт создается чаще.

Анализ запросов

https://its.1c.ru/db/metod8dev/content/6011/hdoc

Если база боевая и перезапускать ее нельзя, то можно накостылять такой скриптец и поместить его в крон, однако далеко не факт, что он что-то выявит:

```
#!/bin/bash

result=$(sudo -u postgres psql  -c "select query_start, state, query from pg_stat_activity where (state = 'idle in transaction') and xact_start is not null and (now() - pg_stat_activity.query_start) > interval '1 seconds';")
echo $result | grep '0 rows' && exit
echo ok
echo $result >> /root/log
```

Для начала будет достаточно включить логгирование долгих запросов, чтобы увидеть, что вообще подвисает

```
ALTER SYSTEM SET lc_messages = 'C'; -- (необяз) все сообщения на английском языке
ALTER SYSTEM SET logging_collector = on; -- включаем логгирование, потребуется перезапуск сервера
ALTER SYSTEM SET log_directory = 'log'; -- указываем директорию для логов
ALTER SYSTEM SET log_min_duration_statement = 1s; -- 0=все запросы. ВНИМАНИЕ. Сбор всех запросов может ухудшить производительность при высокой нагрузке. Поэтому тут укажем 1s
ALTER SYSTEM SET log_duration = on; -- Записывает продолжительность каждой завершённой команды
```

Логи будут в `/var/lib/postgresql/15/main/log/`. Если вылезают какие-то долгие селекты - скорее всего 1С наткнулся на большую таблицу и ее надо разбить. Если вылезают изменения данных - надо проводить оптимизацию.

Чтобы получить детальную статистику по запросам необходимо подключить соотв-ие модули:

> Вообще проводить детальный разбор запросов надо с разработчиком 1С, так как системный администратор мало чем может тут помочь.

```
SELECT current_setting('shared_preload_libraries');
-- Смотрим какие либы подключены, дописываем их в следующем запросе вместо ...,
ALTER SYSTEM SET shared_preload_libraries = ..., 'auto_explain', 'pg_stat_statements';
```

После этого необходимо перезапустить базу. Релоад не работает.

Настраиваем подключенные модули

```
-- auto_explain
ALTER SYSTEM SET auto_explain.log_min_duration = '10s'; -- записывать в лог план для запросов >= 10 сек.
ALTER SYSTEM SET auto_explain.log_analyze = true;
SELECT pg_reload_conf(); -- применение настроек


-- pg_statements
ALTER SYSTEM SET pg_stat_statements.max = 10000;
ALTER SYSTEM SET pg_stat_statements.track = 'all';
SELECT pg_reload_conf(); -- применение настроек
\c ИМЯ_БАЗЫ
CREATE EXTENSION pg_stat_statements;
SELECT pg_stat_statements_reset(); -- для сброса накопленной статистики..
```

Логи будут в `/var/lib/postgresql/15/main/log/`
Теперь надо создать нагрузку. Выполнить типовые операции, прогнать тест Гилева и т.д..

Впринципе можно обойтись одним лишь slow_log'ом. Если таблица, на которой возник затуп, большая, то впринципе и так понятно, что проблема в ее размере и ее надо разбивать.

Настраиваем/тюним

Важно **партиционирование** и **индексы**. Надо нарезать большие таблицы на более мелкие (партиционировать их) и создавать индексы для столбцов, которые чаще всего используются. Но не стоит переусердствовать с индексами.

> Создание индексов нарушает лицензионное соглашение 1С (WTF?)

Базовые настройки ОЗУ и ядер можно взять отсюда: https://pgtune.leopard.in.ua/

Статьи по **оптимизации** под 1С:

- https://habr.com/ru/articles/727250/
- https://infostart.ru/1c/articles/1180438/
- https://sysadminium.ru/postgresql-optimization-for-1c/

Коннектов указывать лучше в районе 100. Где-то говорят что хватит вообще 10 штук, где-то говорят от 500 до 1000. Тут все индивидуально, надо подключать мониторинг и смотреть, сколько реально висит коннектов в среднем.

(Для 1С) Подключим либы:

```
shared_preload_libraries = 'online_analyze, plantuner'
online_analyze.threshold = 50
online_analyze.scale_factor = 0.1
online_analyze.enable = on
online_analyze.verbose = off
online_analyze.min_interval = 10000
online_analyze.table_type = 'temporary'
plantuner.fix_empty_table = on
```

> Модуль plantuner добавляет поддержку указаний для планировщика, позволяющих отключать или подключать определённые индексы при выполнении запроса.
> F.46.1. Объяснение
>
> В некоторых случаях может потребоваться управлять планировщиком, давая ему указания, чтобы оптимизатор не выполнял некоторые части своего алгоритма. В частности, нередко возникают ситуации, когда разработчик хочет временно отключить некоторые индексы, не удаляя их, либо наоборот, использовать определённый индекс принудительно.
>
> Эта версия plantuner даёт возможность скрыть определённые индексы от планировщика Postgres Pro, чтобы он их не использовал. С некоторой нагрузкой Postgres Pro бывает слишком пессимистичным в отношении только что созданных таблиц и считает, что в них содержится гораздо больше строк, чем есть на самом деле. Если же переменная GUC plantuner.fix_empty_table имеет значение true, plantuner будет обнулять число страниц/кортежей в таблице, которая не содержит никаких блоков в файле.

> Модуль `online_analyze` предоставляет набор функций, которые немедленно обновляют статистику после операций `INSERT`, `UPDATE`, `DELETE` или `SELECT INTO` в целевых таблицах.

`online_analyze` на самом деле **включать не стоит**, если 1С достаточно свежий. Подумай. https://it-expertise.ru/blog/records/parametr-online-analyze-postgresql-vs-1c-predpriyatie-8/ :

> Дело в том, что, начиная с версии 8.3.13, платформа 1С самостоятельно включает использование analyze явным образом после вставки во временную таблицу.
>
> Так что при использовании актуальной версии платформы 1С на своих проектах с использованием PostgreSQL не нужно включать online_analyze.
>
> Свои рекомендации, к слову, команда Постгрес актуализировала – сейчас в документации уже отсутствует устаревшая рекомендация.

Проверяем, например, Поиск объектов помеченных на удаление до изменения параметров и после. Не помогло? Смотрим таблицы

```
postgres=# \c test
You are now connected to database "test" as user "postgres".
test=# \dt
test=# \dt+
```

В логе медленных запросов проскакивают несколько запросов по 2м таблицам, криминала не видно. Большое количество таблиц это норма для 1С.

Вакуум

Если нет возможности выполнить вакуум или перезалить базу, можно попробовать эту утилиту: https://github.com/dataegret/pgcompacttable

Если простой в работе допустим - попробуем `VACUUM FULL VERBOSE` . Внимание! Будет создана полная копия базы, потребуется X2 места, которое занимает база, если места мало - не выполняем. Лучше попробовать pgcompacttable

> Если вы недавно перезаливали базу, то вакуум бесполезен, так как при дампе отсекаются все лишние данные, вакуумировать просто нечего.

Цитаты из интернета https://is1c.ru/career/blog/derzhi-dannye-v-teple-tranzaktsii-v-kholode-a-vacuum-v-golode/ :

>

> Но опять же, вспоминаем – у нас с вами на элементарной базе около 25 тысяч индексов. Успеешь состариться, пока проанализируешь их хотя бы один раз – что там выросло, а что нет. Есть специальная утилита – pg_repack.
>
> > NB: Утилила древняя, советую pgcompacttable
>
> Она за вас все пройдет, найдет, проанализирует и сделает в том числе и эту операцию. Утилита pg_repack – это замена операции VACUUM FULL, которая является блокирующей. Для 1С-ников - VACUUM FULL – то же самое, что реструктуризация таблиц. Наводит полный порядок, но работать в базе нельзя. А pg_repack – это почти реструктуризация версии 2.0. Рядом создастся табличка, туда аккуратно переедут данные, потом ваша старая таблица заблокируется, в новую таблицу переедет инкремент (то, что успело создастся с момента перепаковки), потом удалится старая таблица и будет переименована новая, чтобы уже с ней дальше работать. Офлайн все равно случится, но буквально на секунду, на две. Не больше.

> Едем дальше. Мы все настроили – автовакуумы, репаки работают, все классно. Но, когда **база большая (например, несколько ТБ), они не успеют, никто не успеет. Это нереально. Автовакуум будет там полгода ходить, наверное**. Что делать? Надо замораживать (использовать параметр FREEZE)

Про checkpoint timeout:

> Что касается PostgreSQL – там все чуть полегче, не нужно ничего высвобождать. Он работает с кэшем по-другому. Он высвободит все за вас, у него есть счетчик «сколько транзакций сейчас требуют данные, которые я в кэш поднял?». И если этот счетчик равен нулю, он скинет этот кэш, освободит за вас, вам делать ничего не надо. Этим управляет настройка checkpoint timeout. Она по умолчанию раз в пять минут делается. В PostgreSQL с этим ничего специально делать не надо.

Партиционирование

> TODO

https://infostart.ru/1c/articles/79517/

Если мы хотим порезать таблицу, то надо понять, по какому признаку ее резать. По месяцам, по годам, по айдишникам. Партиция может создаваться по диапазону значений или по конкретно заданному списку.

Приступим:

Предположим у нас есть таблица users

Создаем партиции, что означает – наследованные таблицы:

```
$ create table users_1 () inherits (users);


$ \d users_1
                                Table "public.users_1"
    Column     |          Type          |                    Modifiers
---------------+------------------------+-------------------------------------------------
 id            | integer                | not null default nextval('users_id_seq'::regclass)
 username      | text                   | not null
 password      | text                   |
 created_on    | timestamp with time zone | not null
 last_logged_on | timestamp with time zone | not null
Inherits: users
```

Полезные статьи

Сюрпризы планировщика запросов в БД PostgreSQL -

https://habr.com/ru/companies/okko/articles/443276/

Всякое полезное - https://rocket-koala.ru/nastrojka-postgresql-dlya-1s/

https://sysadminium.ru/postgresql-optimization-for-1c/

https://its.1c.ru/db/metod8dev#content:6011:hdoc

https://habr.com/ru/articles/727250/

https://dataegret.com/2017/03/deep-dive-into-postgres-stats-pg_stat_bgwriter-reports/

https://wiki.postgresql.org/wiki/Performance_Optimization

https://www.craigkerstiens.com/2013/01/10/more-on-postgres-performance/

http://www.gilev.ru/category/postgresql-2/

fin

Если никакие оптимизации и советы не помогли - стоит провести аудит самого 1С. Также стоит попробовать кластеризировать/обновить железо 1С (сам кластер серверов 1С:Предприятие, не клиентов), на них лежит логика и это может сильно помочь.

Revision #40
Created 2 May 2023 12:47:09 by Ivan
Updated 14 May 2023 08:47:03 by Ivan