Настройка PostgreSQL для использования с 1C

Экспорт в PDF 

Дата создания: 2022/07/07 03:53 (C) mihanik

PostgreSQL с настройками по-умолчанию готов работать практически на любом железе с любыми базами данных. Однако, настройки по-умолчанию хороши для баз, размер которых не превышает 4-5 Гигабайтов. Если у вас базы 1C больше 4-5 Гигабайтов, настройки PostgreSQL лучше «подрихтовать».

В Интернете есть много рекомендаций по настройке PostgreSQL для нужд 1C. Решил добавить свои 5 копеек.

Опирался на эту рекомендацию.

Для вычислений используется 2 параметра:

- количество ОЗУ, которое вы готовы отдать PostgreSQL;
- количество ядер процессора, которые вы готовы отдать PostgreSQL;

Ссылка на файл с расчётами ниже.

postgresql.ods

Замечания.

Предположим, что у вас есть сервер, на котором «крутятся» 1C и PostgreSQL.

Рассуждаем так.

- одно ядро процессора и 4 Гб ОЗУ отдаём под нужды ОС.
- Половину оставшихся ядер и половину оставшейся оперативки отдаём под нужды PostgreSQL.

UPD

26.06.2025

Для автоматизирования настройки набросал небольшой скрипт для Linux

PG-Tune.sh

```
#!/bin/bash

# Задаём путь к файлу с настройками
PGConf=~/postgresql.conf
#PGConf=/var/lib/pgpro/1c-14/data/postgresql.conf

# Задаём размер ОЗУ в мегабайтах, который отводится для PostgreSQL.

MyMem=32768

# Задаём количество ядер процессора, которые отводятся для PostgreSQL.

MyCPU=6

(
echo '#------------------------------------------------------------------
--------------'
echo '# CUSTOMIZED BY MIHANIK OPTIONS'
echo '#------------------------------------------------------------------
--------------'

echo "listen_addresses ='*'"

#Отключение контроля на уровне записей.
echo "row_security = off"

# Выключение шифрования, которое может приводить к увеличению загрузки CPU.
echo "ssl = off"

# Количество памяти, выделенной PostgreSQL для совместного кеша страниц.
# Эта память разделяется между всеми процессами PostgreSQL.
echo "shared_buffers = $(($MyMem / 4))MB"

# Количество памяти, выделенной PostgreSQL для совместного кеша страниц.
# Эта память разделяется между всеми процессами PostgreSQL.
echo "temp_buffers = 256MB"

# Лимит памяти для обработки одного запроса.
# work_mem = RAM/32..64 или 32MB..128MB
echo "work_mem = $(($MyMem / 24))MB"

# maintenance_work_mem = RAM/16..32 или work_mem * 4 или 256MB..4GB
echo "maintenance_work_mem = $(($MyMem / 6))MB"

# Сброс буферов на диск (выполнение PostgerSQL системных вызовов fsync()).
#  Выключение параметра приводит к росту производительности, но появляется
# значительный риск потери всех данных при внезапном выключении питания.
echo "fsync = on"
```

```
# Выключение синхронной записи в WAL момент коммита транзакции.
# Создает риск потери последних нескольких транзакций (в течении 0.5-1" секунды),
# но гарантирует целостность базы данных. Может значительно увеличить производительность.
echo "synchronous_commit = off"

# Минимальное и максимальный объем WAL файлов.
# Аналогично checkpoint_segments.
echo "min_wal_size = $(($MyMem / 8))MB"
echo "max_wal_size = $(($MyMem / 4))MB"

# Групповой коммит нескольких транзакций.
# Имеет смысл включать, если интенсивность транзакций превосходит 1000 TPS.
echo "commit_delay = 1000"
echo "commit_siblings = 5"

# Время сна между циклами записи на диск фонового процесса записи.
echo "bgwriter_delay = 20ms"

# Параметры, управляющие интенсивностью записи фонового процесса записи.
# За один цикл bgwriter записывает не больше, чем было записано в прошлый цикл, умноженное на bgwriter_lru_multiplier, но не больше чем bgwriter_lru_maxpages.
echo "bgwriter_lru_multiplier = 4.0"
echo "bgwriter_lru_maxpages = 400"

# Включение автовакуума.
echo "autovacuum = on"

# Количество процессов автовакуума.
# Общее правило - чем больше запросов на запись выполняется в системе
# (такие системы называются OLTP), тем больше процессов.
# autovacuum_max_workers =" CPU "cores/4..2 но не меньше 4
if [ $((MyCPU /4 )) -lt 4 ];
then
echo "autovacuum_max_workers = 4"
else
echo "autovacuum_max_workers = $(($MyCPU / 4))"
fi

# Время сна процесса автовакуума. Слишком большая величина будет приводить к тому, что таблицы не будут успевать «чиститься»,
# что приведет у роста размера и снижению производительности работы. Малая величина приведет к бесполезной нагрузке.
echo "autovacuum_naptime = 20s"

# Значение по умолчанию – 8000, его не нужно уменьшать.
echo "max_files_per_process = 8000"

# effective_cache_size =" RAM - "shared_buffers
# Оценка планировщика запроса о размере дискового кеша, доступного для одного
```

```
запроса.
echo "effective_cache_size = $(($MyMem - $(($MyMem / 4)) ))MB"

# random_page_cost = 1.5-2.0 для RAID, 1.1-1.3 для SSD
# Стоимость чтения рандомной страницы, на которую будет опираться оптимизатор (по-умолчанию 4).
echo "random_page_cost = 1.2"

# Задаёт максимальное число элементов в списке FROM, до которого планировщик будет объединять вложенные запросы с внешним запросом.
# При меньших значениях сокращается время планирования, но план запроса может стать менее эффективным.
echo "from_collapse_limit= 20"

# Задаёт максимальное количество элементов в списке FROM, до достижения которого планировщик будет сносить в него явные конструкции JOIN (за исключением FULL JOIN).
# При меньших значениях сокращается время планирования, но план запроса может стать менее эффективным.
echo "join_collapse_limit = 20"

# GEQO - генетический оптимизатор запросов PostgreSQL, который осуществляет планирование запросов, применяя эвристический поиск вместо полного перебора отношений.
echo "geqo = on"

# Задаёт минимальное число элементов во FROM, при котором для планирования запроса будет привлечён генетический оптимизатор.
echo "geqo_threshold = 12"

# Оценочное значение одновременных запросов к дисковой системе, которые она может обслужить единовременно.
echo "effective_io_concurrency = 2"

# Разрешить использовать символ \ для экранирования.
echo "standard_conforming_strings = off"

#Не выдавать предупреждение о использовании символа \ для экранирования.
echo "escape_string_warning = off"

# Максимальное число блокировок индексов/таблиц в одной транзакции.
# Для высоконагруженных систем с большими конфигурациями с большим числом таблиц рекомендуется устанавливать значение больше, например, 1000.
echo "max_locks_per_transaction = 1000"

# Количество одновременных соединений.
# Для высоконагруженных систем с большим числом одновременных пользователей рекомендуется устанавливать значение больше, например, 2000.
echo "max_connections = 1000"

# В общем случае мы не рекомендуем использовать синхронное автообновление
```

```
статистики, однако его можно включить, если есть основания полагать,
# что фоновое обновление не дает нужного результата / оптимизатор часто ошибается в оценке количества строк.
echo "online_analyze.enable = off"

) >>  $PGConf
```

Наверх

В моей WIKI постоянно ведётся какая-то работа со статьями.
Если у вас возникли вопросы или замечания,
можете их отправлять на почту **support@mihanik.net**