<https://habr.com/ru/companies/otus/articles/859270/>

# Ускоряем PostgreSQL с помощью Tuned

**6 мин**

**6K**

[Блог компании OTUS](https://habr.com/ru/companies/otus/articles/)[Настройка Linux\*](https://habr.com/ru/hubs/linux/)[PostgreSQL\*](https://habr.com/ru/hubs/postgresql/)[IT-инфраструктура\*](https://habr.com/ru/hubs/it-infrastructure/)

Администраторам баз данных всегда хочется, чтобы их СУБД работали быстрее. Всегда кажется, что можно прооптимизировать определенные настройки, и запросы начнут отрабатывать быстрее.

Есть множество различных параметров и настроек, которые позволяют оптимизировать работу БД PostgreSQL. В этой статье мы не будем пытаться охватить их все и поговорим об оптимизации параметров операционной и файловой систем, а также памяти на самом сервере СУБД.

## Оптимизируем ОС

Далее мы будем предполагать, что наш PostgreSQL развернут под Linux. В моих примерах будет использоваться Ubuntu, но это не принципиально.

Начнем с оптимизации ОС. Конечно, можно вручную вносить изменения /etc/sysctl, но лучше воспользоваться специализированными утилитами, например, tuned. Установим эту утилиту.

$ sudo apt update

$ sudo apt install tuned

И запустим:

$ sudo systemctl enable --now tuned

Утилита tuned поставляется с множеством предопределенных профилей. Вы можете получить их список с помощью команды:

tuned-adm list

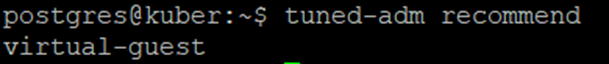


Вы можете запустить следующую команду, чтобы узнать, какой профиль порекомендует настроенный демон tuned после оценки вашей системы:

tuned-adm recommend

Используйте приведенную ниже команду, чтобы увидеть предварительно настроенное значение:

tuned-adm active



В моем примере используется виртуальная машина под Ubuntu, и утилита соответственно предложила профиль virtual-guest. Однако очень часто настройки по умолчанию могут замедлить работу самой СУБД PostgreSQL, например, они могут предпочесть энергосбережение, что, естественно, замедлит работу процессоров. Аналогичные аргументы справедливы и для настройки сети и интерфейсов ввода-вывода.

Поэтому, если мы хотим оптимизировать работу базы данных на нашем сервере, то нам лучше всего не полагаться на уже готовые профили, а создать наш собственный профиль производительности PostgreSQL.

## Создаем новый профиль

Создать новый профиль можно довольно просто. Файлы с профилями tuned хранятся в /etc/tuned, соответственно, нам потребуются права root. Создадим в /etc/tuned каталог pg\_test, а в нем — файл tuned.conf:

sudo mkdir /etc/tuned/pg\_test

cd pg\_test

sudo nano tuned.conf

[main]  
  
summary=Tuned profile for PostgreSQL Instances  
  
[bootloader]  
  
cmdline=transparent\_hugepage=never  
  
[cpu]  
  
governor=performance  
energy\_perf\_bias=performance  
min\_perf\_pct=100  
  
[sysctl]  
  
vm.swappiness = 10  
vm.dirty\_expire\_centisecs = 500  
vm.dirty\_writeback\_centisecs = 250  
vm.dirty\_ratio = 10  
vm.dirty\_background\_ratio = 3  
vm.overcommit\_memory=0  
net.ipv4.tcp\_timestamps=0  
  
[vm]  
  
transparent\_hugepages=never

Строки, содержащие [ и ], называются настроенными плагинами и используются для взаимодействия с соответствующей частью системы. Давайте подробнее рассмотрим эти параметры и значения:

Плагин [main] содержит сводную информацию и может также включать значения из других настроенных профилей с помощью инструкций include, хотя в нашем примере такие конструкции не используются.

Плагин [cpu] включает в себя настройки и мощности процессора.

Плагин [sysctl] включает в себя значения, которые взаимодействуют с procfs.

Плагины [vm] и [bootloader] включают/отключают прозрачные большие страницы памяти (плагин bootloader поможет нам взаимодействовать с параметрами командной строки GRUB). Большие страницы — это группа из следующих друг за другом страниц виртуальной памяти, объединенных в одну большую.

В результате использования нашего профиля мы хотим достичь следующих результатов. Во-первых, процессоры не будут переходить в режим энергосбережения и наша СУБД PostgreSQL не будет страдать от случайного снижения производительности. Во-вторых, операционная система будет меньше использовать файл подкачки. Параметр vm.swappiness влияет на работу системы со swap. Он задает процент свободной оперативной памяти, при котором начинается использование подкачки. Значение vm.swappiness по умолчанию — 60. У нас указано 10.

В-третьих, ядро поможет Postgres обрабатывать измененные (dirty) данные на диск. Используемые vm.dirty\_ratio и vm.dirty\_background\_ratio — это параметры для настройки работы подсистемы виртуальной памяти ядра Linux и записи изменённых (dirty) данных на диск.

vm.dirty\_ratio указывает количество страниц, при достижении которого выполняющий запись процесс сам начнёт записывать изменённые данные из памяти на диск.

vm.dirty\_background\_ratio определяет количество страниц, при достижении которого фоновые потоки для записи данных в ядре начнут записывать изменённые данные из памяти на диск.

Также, мы отключаем временные метки TCP, чтобы избежать или, по крайней мере, уменьшить всплески, вызванные их генерацией.

И наконец, отключение прозрачных огромных страниц transparent\_hugepages=never значительно повышает производительность PostgreSQL в целом.

Разобрав, какие именно изменения мы хотим внести в настройки операционной системы, давайте применим их. Для этого выполним следующую команду:

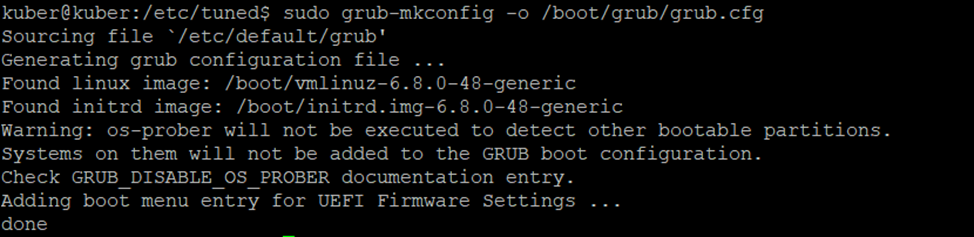
sudo tuned-adm profile pg\_test

Выполнение этой команды может занять некоторое время.

Для того, чтобы завершить нашу борьбу с прозрачными большими страницами, выполним команду и перезагрузимся:

sudo grub-mkconfig -o /boot/grub/grub.cfg

sudo systemctl start reboot.target



Не забудьте убедиться, что система успешно пережила перезагрузку.

## Оптимизация файловой системы

Итак, мы разобрались с операционной системой, но еще надо оптимизировать файловую систему для повышения производительности. PostgreSQL не использует время (временную метку, при которой к файлу осуществлялся последний доступ) для файлов данных, поэтому их отключение сэкономит циклы работы процессора.

Давайте откроем файл /etc/fstab и добавим noatime рядом со значением по умолчанию для диска, на котором хранятся данные PostgreSQL и файлы WAL.

 /dev/mapper/pgdata-01-data /pgdata xfs defaults,noatime 1 1

Активируем изменения с помощью команды:

 mount -o remount,noatime,nodiratime /pgdata

Эти рекомендации хороши для начала, но вам нужно следить как за операционной системой, так и за PostgreSQL, для того, чтобы в дальнейшем выполнять тонкую настройку.

## Оптимизация файловой системы

Ранее мы уже отключили для повышения производительности прозрачные большие страницы. Теперь давайте разберемся с большими страницами в целом.

По умолчанию размер страницы в Linux равен 4 КБ. Обычный экземпляр PostgreSQL может требовать много гигабайт памяти, что приводит к потенциальным проблемам с производительностью из-за небольшого размера страницы. Кроме того, поскольку эти страницы будут фрагментированы, их отображение для больших наборов данных требует дополнительного времени.

Включение больших страниц в Linux повысит производительность PostgreSQL, поскольку в нем будут выделяться большие блоки памяти в целом.

По умолчанию в Linux не включены большие страницы, что также подходит для работы PostgreSQL, так как по умолчанию в конфигурационном файле СУБД используется значение huge\_pages try, что означает «использовать большие страницы, если они доступны в ОС, в противном случае нет».

Настройка больших страниц для PostgreSQL зависит от двух аспектов: настройки операционной системы и настройки PostgreSQL.

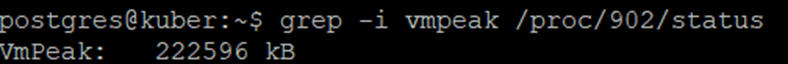
Сначала выясните, сколько больших страниц требуется вашей системе для PostgreSQL. Когда запускается экземпляр PostgreSQL, postmaster создает файл postmaster.pid в $PGDATA. Там вы можете найти pid основного процесса:

$ head -n 1 $PGDATA/postmaster.pid

902

Далее давайте найдем пиковое значение использования виртуальной памяти VmPeak для данного инстанса:

$ grep -i vmpeak /proc/902/status



Обратите внимание, на то, что если у вас на одном сервере запущено более одного экземпляра PostgreSQL, на следующем шаге рассчитайте сумму всех пиковых значений.

Узнаем Hugepagesize:

$ grep -i hugepagesize /proc/meminfo



Наконец, давайте подсчитаем количество больших страниц, которые понадобятся нашим экземпляру(ам):

222596 / 2048 = 108,68

В случае с моей виртуалкой потребовалось относительно немного страниц, но на боевой системе это соотношение может быть гораздо больше. Если вы слишком сильно увеличите это значение, процессы, которым требуется мало страниц и места в операционной системе, не смогут запуститься. Это может даже привести к сбою загрузки операционной системы или сбою запуска других экземпляров PostgreSQL на том же сервере.

Теперь отредактируйте tuned.conf, созданный ранее, и добавьте следующую строку в раздел [sysctl]:

 vm.nr\_hugepages=110

Затем примените изменения:

sudo tuned-adm profile pg\_test

Далее в файле настроек СУБД postgresql.conf можно разрешить использование больших страниц, указав:

 huge\_pages=on

После этого не забудьте перезапустить PostgreSQL.

Для того, чтобы завершить наши настройки, нам необходимо, чтобы демон tuned запускался перед демоном PostgreSQL. Для этого выполним:

 sudo systemctl edit postgresql-13.service

И добавим в файл две строки:

 [Unit]

After=tuned.service

Затем:

sudo systemctl daemon-reload

## Заключение

В этой статье мы рассмотрели основные моменты, связанные с оптимизацией работы операционной и файловой системы на сервере СУБД PostgreSQL. Помимо этого, для оптимизации работы БД можно выполнить также ряд других настроек, о которых пойдет речь в последующих статьях.

Больше актуальных навыков по инфраструктуре приложений вы можете получить в рамках [практических онлайн-курсов](https://otus.pw/nH3o/) от экспертов отрасли. Также приглашаем на ближайшие открытые уроки по PostgreSQL:

* 25 ноября: «Что нового в PostgreSQL 17?». [Узнать подробнее](https://otus.pw/65oV/)
* 28 ноября: «Использование Foreign-Data Wrappers в PostgreSQL». [Узнать подробнее](https://otus.pw/3LCk/)

**Теги:**

* [linux](https://habr.com/ru/search/?target_type=posts&order=relevance&q=%5blinux%5d)
* [postgresql](https://habr.com/ru/search/?target_type=posts&order=relevance&q=%5bpostgresql%5d)
* [ит-инфраструктура](https://habr.com/ru/search/?target_type=posts&order=relevance&q=%5b%D0%B8%D1%82-%D0%B8%D0%BD%D1%84%D1%80%D0%B0%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0%5d)
* [tuned](https://habr.com/ru/search/?target_type=posts&order=relevance&q=%5btuned%5d)

**Хабы:**

* [Блог компании OTUS](https://habr.com/ru/companies/otus/articles/)
* [Настройка Linux](https://habr.com/ru/hubs/linux/)
* [PostgreSQL](https://habr.com/ru/hubs/postgresql/)
* [IT-инфраструктура](https://habr.com/ru/hubs/it-infrastructure/)