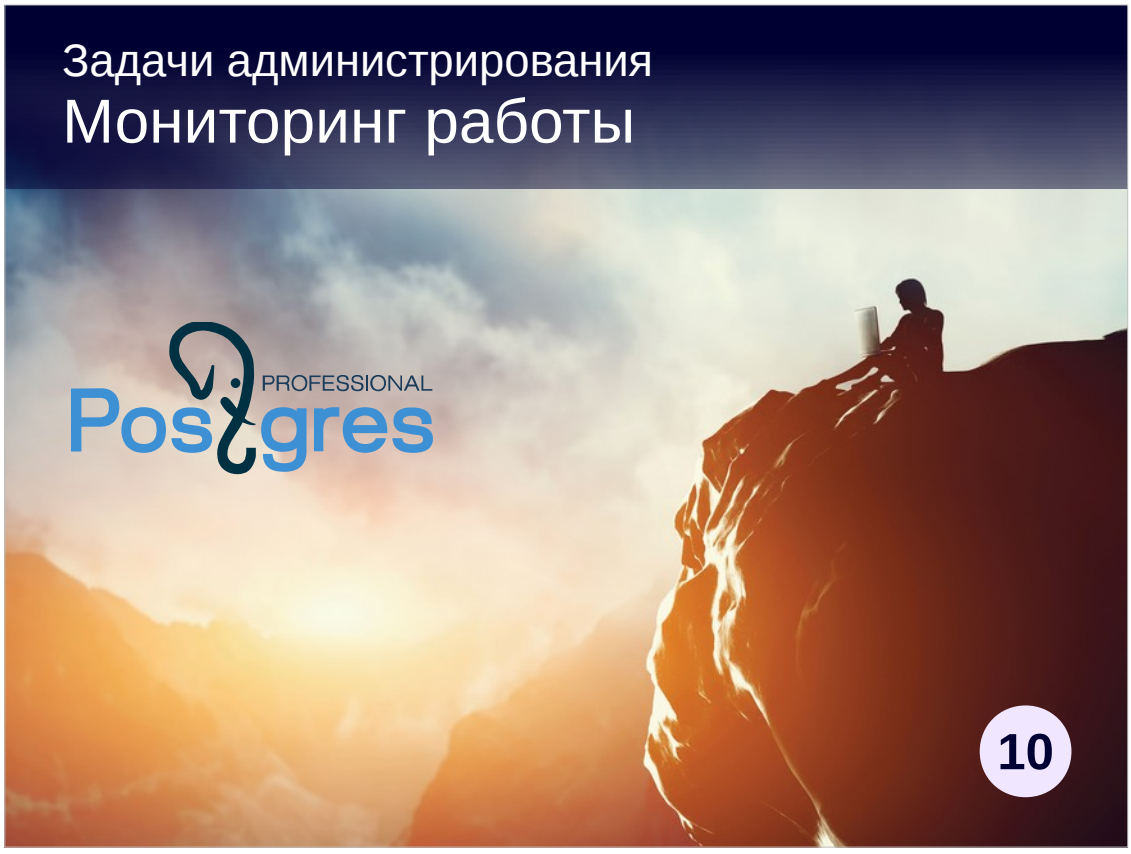


Авторы: Егор Рогов, Павел Лузанов

Обратная связь
Отзывы, замечания и предложения направляйте по адресу:
edu@postgrespro.ru

Темы

Средства операционной системы

Статистика внутри базы

Журнал сообщений сервера

Внешние системы мониторинга

Средства ОС

Процессы

- ps (grep postgres)
- параметр update_process_title для обновления статуса процессов

Использование ресурсов

- iostat, vmstat, sar, top...

Дисковое пространство

- df, du, quota...

PostgreSQL работает под управлением операционной системы и в известной степени зависит от ее настроек.

Unix предоставляет множество инструментов для анализа состояния и производительности.

В частности, можно посмотреть процессы, принадлежащие PostgreSQL. Это особенно полезно при включенном (по умолчанию) параметре сервера update_process_title, когда в имени процесса отображается его текущее состояние.

Для изучения использования системных ресурсов (процессор, память, диски) имеются различные инструменты: iostat, vmstat, sar, top и др.

Необходимо следить и за размером дискового пространства. Место, занимаемое базой данных, можно смотреть как из самой БД (см. модуль «Организация данных»), так из ОС (команда du). Размер доступного дискового пространства надо смотреть в ОС (команда df). Если используются дисковые квоты, надо принимать во внимание и их.

В целом набор инструментов и подходы может сильно различаться в зависимости от используемой ОС и файловой системы, поэтому подробно здесь не рассматриваются.

https://postgrespro.ru/docs/postgresql/10/monitoring-ps

https://postgrespro.ru/docs/postgresql/10/diskusage

Статистика внутри базы

Текущие активности системы

Процесс сбора статистики

Дополнительные расширения

Существует два основных источника информации о происходящем в системе. Первый из них — статистическая информация, которая собирается PostgreSQL и хранится внутри базы данных.

Текущие активности

Настройка

статистика	*параметр*
текущая активность и ожидания обслуживающих и фоновых процессов	track_activities включен по умолчанию

Текущая активность всех обслуживающих процессов и (начиная с версии 10) фоновых процессов отображается в представлении pg_stat_activity. Подробнее на нем мы остановимся в демонстрации.

Кроме этого, есть еще несколько представлений, показывающих текущие активности сервера (работу очистки, репликации и т. п.).

Работу этих представлений можно отключить параметром track_activities, но делать этого не следует.

Сбор статистики

Настройки процесса stats collector

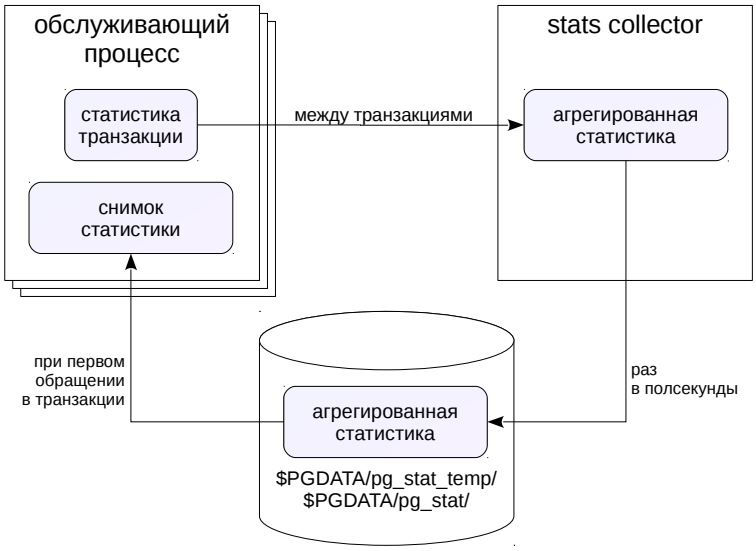
статистика	*параметр*
обращения к таблицам и индексам (доступы, затронутые строки)	track_counts включен по умолчанию и нужен для автоочистки
обращения к страницам	track_io_timing выключен по умолчанию
вызовы пользовательских функций	track_functions выключен по умолчанию

Кроме показа действий, непосредственно происходящих в данный момент, PostgreSQL собирает и некоторую статистику.

Сбором статистики занимается фоновый процесс stats collector. Количеством собираемой информации управляют несколько параметров сервера, так как чем больше информации собирается, тем больше и накладные расходы.

https://postgrespro.ru/docs/postgresql/10/monitoring-stats

Архитектура



Каждый обслуживающий процесс собирает необходимую статистику в рамках каждой выполняемой транзакции. Затем эта статистика передается процессу-коллектору. Коллектор собирает и агрегирует статистику со всех обслуживающих процессов. Раз в полсекунды (время настраивается при компиляции) коллектор сбрасывает статистику во временные файлы в каталог $PGDATA/pg_stat_temp. (Поэтому перенесение этого каталога в файловую систему в памяти может положительно сказаться на производительности.)

Когда обслуживающий процесс запрашивает информацию о статистике (через представления или функции), в его память читается последняя доступная версия статистики — это называется *снимком статистики.* Если не попросить явно, снимок не будет перечитываться до конца транзакции, чтобы обеспечить согласованность.

Таким образом, из-за задержек серверный процесс получает не самую свежую статистику — но обычно это и не требуется.

При останове сервера коллектор сбрасывает статистику в постоянные файлы в каталог $PGDATA/pg_stat. Таким образом, статистика сохраняется при перезапуске сервера. Обнуление счетчиков происходит по команде администратора, а также при восстановлении сервера после сбоя.

Дополнительная статистика

Postgres PROFESSIONAL

Расширения в поставке

pg_stat_statements	статистика по запросам
pgstattuple	статистика по версиям строк
pg_buffercache	состояние буферного кэша

Другие расширения

pg_stat_plans	статистика по планам запросов
pg_stat_kcache	статистика по процессору и вводу-выводу
pg_qualstats	статистика по предикатам
…	

Существуют расширения, позволяющие собирать дополнительную статистику, как входящие в поставку, так и внешние.

Например, расширение pg_stat_statements сохраняет информацию о запросах, выполняемых СУБД; pg_buffercache позволяет заглянуть в содержимое буферного кэша и т. п.

Журнал сообщений

Настройка журнальных записей

Ротация файлов журнала

Анализ журнала

Второй важный источник информации о происходящем на сервере — журнал сообщений.

Журнал сообщений

Приемник сообщений (log_destination = *список*)

stderr	поток ошибок
csvlog	формат CSV (только с коллектором)
syslog	демон syslog
eventlog	журнал событий Windows

Коллектор сообщений (logging_collector = on)

- позволяет собирать дополнительную информацию
- никогда не теряет сообщения (в отличие от syslog)
- записывает stderr и csvlog в log_directory/log_filename

Журнал сообщений сервера можно направлять в разные приемники и выводить в разных форматах. Основной параметр, который определяет приемник и формат — log_destination (можно указать один или несколько приемников через запятую).

Значение stderr (установленное по умолчанию) выводит сообщения в стандартный поток ошибок в текстовом виде. Значение syslog направляет сообщения демону syslog в Unix-системах, а eventlog — в журнал событий Windows.

Обычно дополнительно включают специальный процесс — коллектор сообщений. Он позволяет записать больше информации, поскольку собирает ее со всех процессов, составляющих PostgreSQL. Он спроектирован так, что никогда не теряет сообщения; как следствие при большой нагрузке он может стать узким местом.

Коллектор сообщений включается параметром logging_collector. При значении stderr информация записывается в каталог, определяемый параметром log_directory, в файл, определяемый параметром log_filename.

Включенный коллектор сообщений позволяет также указать приемник csvlog; в этом случае информация будет сбрасываться в формате CSV в файл log_filename с расширением .csv.

Информация в журнале

Настройки

информация	*параметр*
сообщения определенного уровня	log_min_messages
время выполнения длинных команд	log_min_duration_statement
время выполнения команд	log_duration
имя приложения	application_name
контрольные точки	log_checkpoints
подключения и отключения	log_(dis)connections
длинные ожидания	log_lock_waits
текст выполняемых команд	log_statement
использование временных файлов	log_temp_files
...	

В журнал сообщений сервера можно выводить множество полезной информации. По умолчанию почти весь вывод отключен, чтобы не превратить запись журнала в узкое место для дисковой подсистемы. Администратор должен решить, какая информация важна, обеспечить необходимое место на диске для ее хранения и оценить влияние записи журнала на общую производительность системы.

Ротация файлов журнала

С помощью коллектора сообщений

настройка	*параметр*
маска имени файла	log_filename
время ротации, мин	log_rotation_age
размер файла для ротации, КБ	log_rotation_size
перезаписывать ли файлы	log_truncate_on_rotation = on

комбинируя маску файла и время ротации, получаем разные схемы:

`'postgresql-%H.log', '1h'`	24 файла в сутки
`'postgresql-%a.log', '1d'`	7 файлов в неделю

Внешние средства

например, 24 файла в сутки с Apache rotatelogs:

```
pg_ctl start | rotatelogs имя_файла 3600 -n 24
```


Если записывать журнал в один файл, рано или поздно он вырастет до огромных размеров, что крайне неудобно для администрирования и анализа. Поэтому обычно используется та или иная схема ротации журналов.

https://postgrespro.ru/docs/postgresql/10/logfile-maintenance

Коллектор сообщений имеет встроенные средства ротации, которые настраиваются несколькими параметрами, основные из которых приведены на слайде.

Параметр log_filename может задавать не просто имя, а маску имени файла с помощью спецсимволов даты и времени.

Параметр log_rotation_age задает время переключения на следующий файл в минутах (а log_rotation_size — размер файла, при котором надо переключаться на следующий).

Включение log_truncate_on_rotation перезаписывает уже существующие файлы.

Таким образом, комбинируя маску и время переключения, можно получать разные схемы ротации.

https://postgrespro.ru/docs/postgresql/10/runtime-config-logging.html#RUNTIME-CONFIG-LOGGING-WHERE

Альтернативно можно воспользоваться внешними программами ротации, например rotatelogs.

Анализ журнала

Средства операционной системы

- grep, awk...

Специальные средства анализа

- pgBadger — требует определенных настроек журнала

Анализировать журналы можно по-разному.

Можно искать определенную информацию средствами ОС или специально разработанными скриптами.

Стандартом де-факто для анализа является программа pgBadger, но надо иметь в виду, что она накладывает определенные ограничения на содержимое журнала. В частности, допускаются сообщения только на английском языке.

Внешний мониторинг

Универсальные системы мониторинга

- Zabbix, Munin, Cacti...
- в облаке: Okmeter, NewRelic, Datadog...

Системы мониторинга PostgreSQL

- PGObserver
- PostgreSQL Workload Analyzer (PoWA)
- Open PostgreSQL Monitoring (OPM)
- ...

На практике, если подходить к делу серьезно, требуется полноценная система мониторинга, которая собирает различные метрики как с PostgreSQL, так и с операционной системы, хранит историю этих метрик, отображает их в виде понятных графиков, имеет средства оповещения при выходе определенных метрик за установленные границы и т. д.

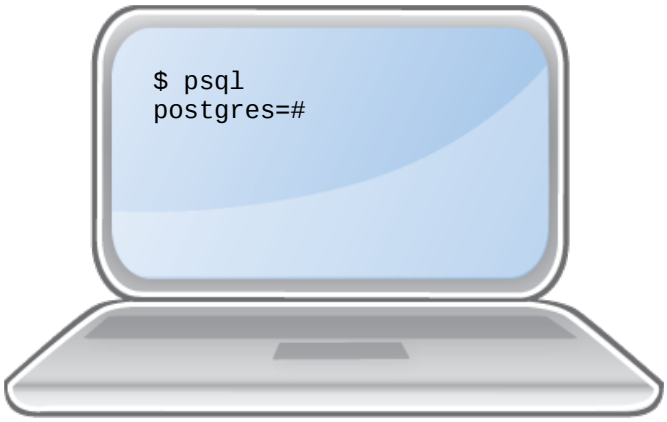
Собственно PostgreSQL не располагает такой системой; он только предоставляет средства для получения информации о себе (которые мы рассмотрели). Поэтому для полноценного мониторинга нужно выбрать внешнюю систему.

Таких систем существует довольно много. Если универсальные системы, имеющие плагины или агенты для PostgreSQL. К ним относятся Zabbix, Munin, Cacti, облачные сервисы Okmeter, NewRelic, Datadog и другие.

Есть и системы, ориентированные специально на PostgreSQL, такие, как PGObserver, PoWA, OPM и т. д.

Неполный, но представительный список систем мониторинга можно посмотреть на странице https://wiki.postgresql.org/wiki/Monitoring

Демонстрация



Итоги

Мониторинг заключается в контроле работы сервера
как со стороны операционной системы,
так и со стороны базы данных

PostgreSQL предоставляет собираемую статистику
и журнал сообщений сервера

Для полноценного мониторинга требуется внешняя система

Практика

1. В новой базе данных создайте таблицу, выполните вставку нескольких строк, а затем удалите все строки.
2. Посмотрите статистику обращений к таблице и сопоставьте цифры (n_tup_ins, n_tup_del, n_live_tup, n_dead_tup) с вашей активностью.
3. Выполните очистку (vacuum), снова проверьте статистику и сравните с предыдущими цифрами.
4. Создайте ситуацию взаимоблокировки двух транзакций.
5. Посмотрите, какая информация записывается при этом в журнал сообщений сервера.

4. Взаимоблокировка (deadlock) — ситуация, в которой две (или больше) транзакций ожидают друг друга. В отличие от обычной блокировки, при взаимоблокировке у транзакций нет возможности выйти из этого «тупика» и СУБД вынуждена принимать меры — одна из транзакций будет принудительно прервана, чтобы остальные могли продолжить выполнение.

Проще всего воспроизвести взаимоблокировку на таблице с двумя строками. Первая транзакция меняет (и, соответственно, блокирует) первую строку, а вторая — вторую. Затем первая транзакция пытается изменить вторую строку и «повисает» на блокировке. А потом вторая транзакция пытается изменить первую строку — и тоже ждет освобождения блокировки.