



# Мониторинг работы



Средства операционной системы

Сбор и просмотр статистики

Дополнительные расширения

Просмотр и анализ журнала сервера

## Процессы

ps (grep postgres)

параметр `update_process_title` для обновления статуса процессов

## Использование ресурсов

iostat, vmstat, sar, top...

## Дисковое пространство

df, du, quota...

## Настройки процесса stats collector

### *статистика*

текущие активности сервера  
и фоновых процессов

обращения к таблицам и индексам  
(доступы, затронутые строки)

обращения к страницам

вызовы пользовательских функций

### *параметр*

track\_activities

включен

track\_counts

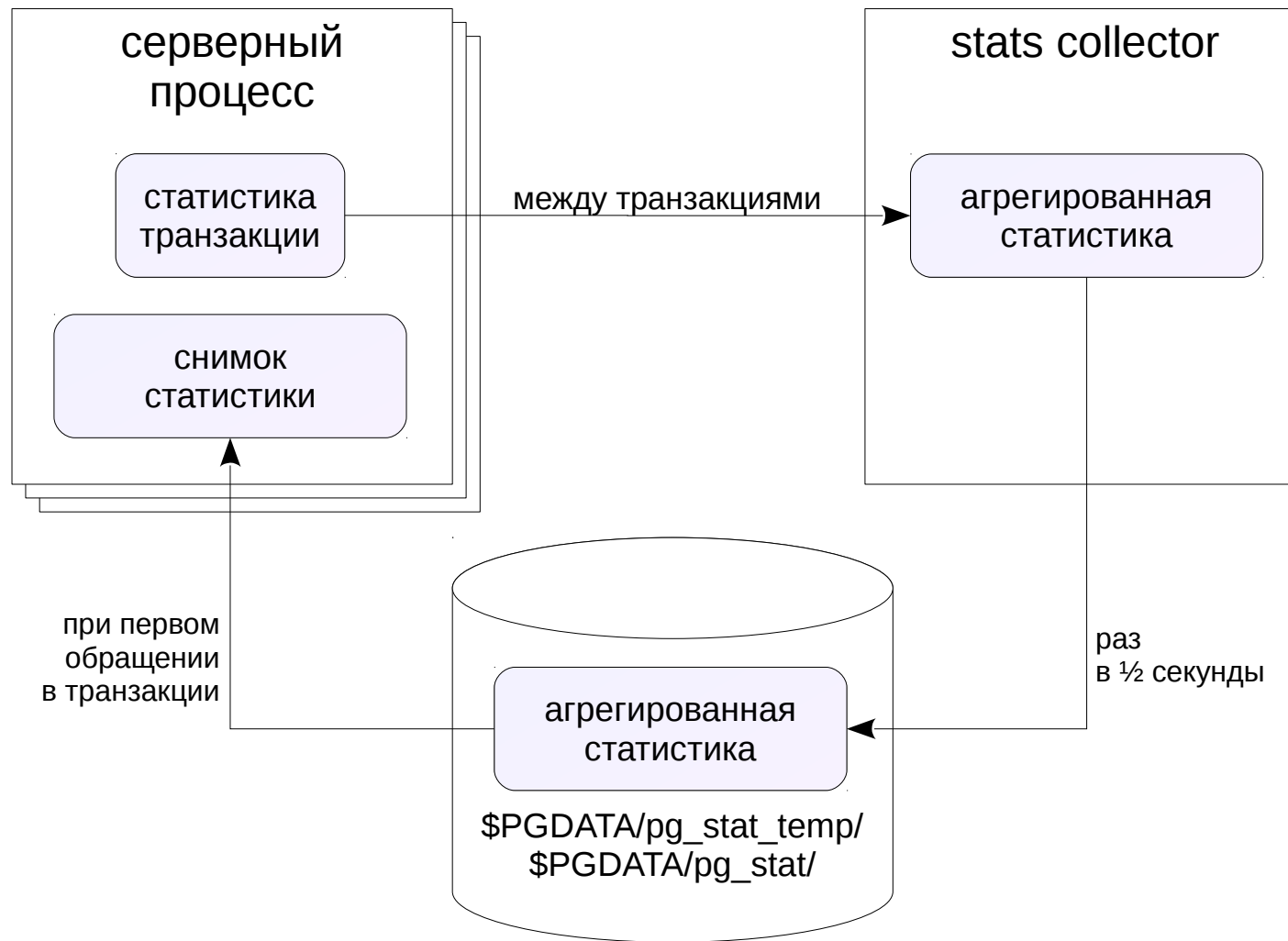
включен, нужен для автоочистки

track\_io\_timing

выключен

track\_functions

выключен



## 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

...

## Настройки

### *информация*

сообщения определенного уровня  
время выполнения длинных команд  
время выполнения команд  
имя приложения  
контрольные точки  
подключения и отключения  
длинные ожидания  
текст выполняемых команд  
использование временных файлов  
...

### *параметр*

log\_min\_messages  
log\_min\_duration\_statement  
log\_duration  
application\_name  
log\_checkpoints  
log\_(dis)connections  
log\_lock\_waits  
log\_statement  
log\_temp\_files

## Встроенные средства

### *настройка*

включение сбора сообщений  
формат журнала  
перезаписывать ли файлы  
маска имени файла  
время ротации, мин

### *параметр*

logging\_collector = on  
log\_destination = stderr, csvlog, syslog...  
log\_truncate\_on\_rotation = on  
log\_filename  
log\_rotation\_age

комбинируя маску файла и время ротации, получаем разные схемы:

postgresql-%H.log, 60

24 файла в сутки

postgresql-%a.log, 1440

7 файлов в неделю

## Внешние средства

например, Apache rotatelog

```
pg_ctl start | rotatelog имя_файла 86400 -n 24
```



Средства операционной системы

grep, awk...

Специальные средства анализа

pgBadger

Системы мониторинга PostgreSQL

PostgreSQL Workload Analyzer (PoWA)

Open PostgreSQL Monitoring (OPM)

Универсальные системы мониторинга

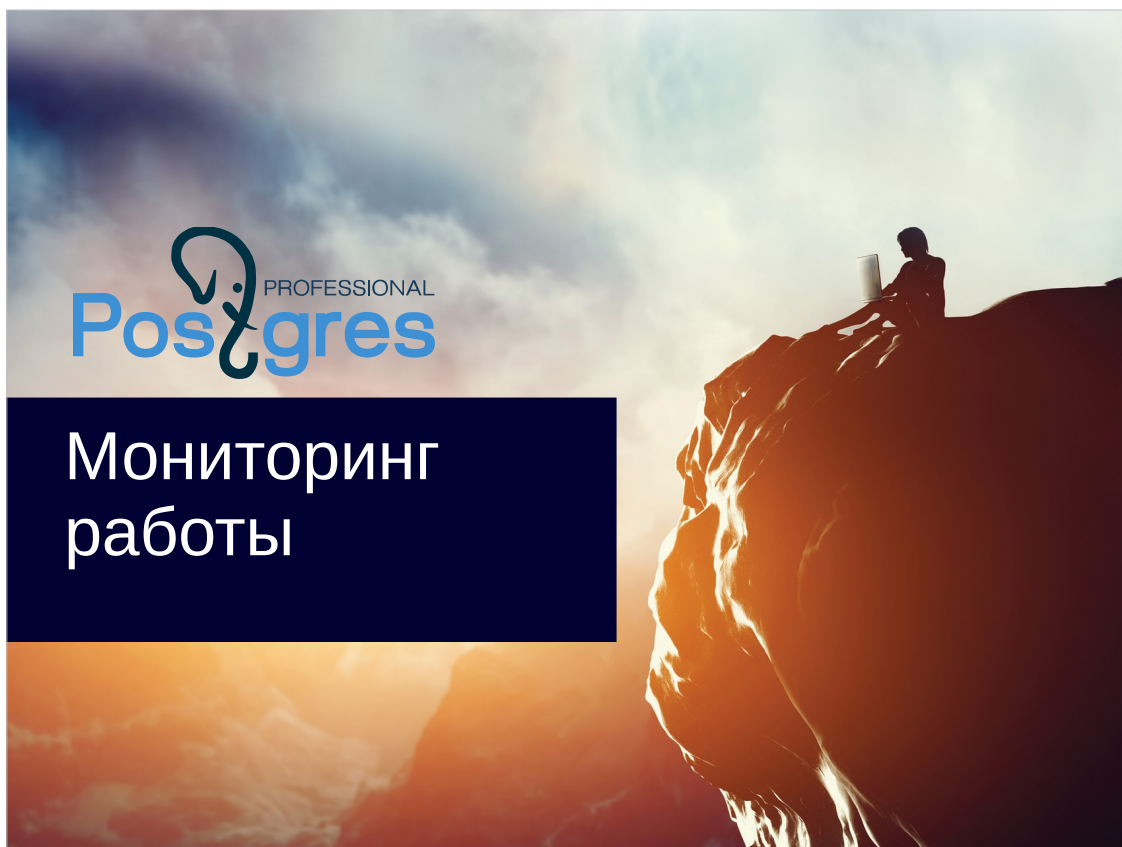
Nagios, Munin, Zabbix, Cacti...



Мониторинг заключается в контроле работы сервера как со стороны операционной системы, так и со стороны PostgreSQL.

Для этого используется собранная статистика и анализ журнала сервера.

1. В базе данных DB14 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 сервера.



### **Авторские права**

Курс «Администрирование PostgreSQL 9.4. Базовый курс» разработан в компании Postgres Professional (2015 год).

Авторы: Егор Рогов, Павел Лузанов

### **Использование материалов курса**

Некоммерческое использование материалов курса (презентации, демонстрации) разрешается без ограничений. Коммерческое использование возможно только с письменного разрешения компании Postgres Professional. Запрещается внесение изменений в материалы курса.

### **Обратная связь**

Отзывы, замечания и предложения направляйте по адресу:  
[edu@postgrespro.ru](mailto:edu@postgrespro.ru)

### **Отказ от ответственности**

Компания Postgres Professional не несет никакой ответственности за любые повреждения и убытки, включая потерю дохода, нанесенные прямым или косвенным, специальным или случайным использованием материалов курса. Компания Postgres Professional не предоставляет каких-либо гарантий на материалы курса. Материалы курса предоставляются на основе принципа «как есть» и компания Postgres Professional не обязана предоставлять сопровождение, поддержку, обновления, расширения и изменения.

Средства операционной системы

Сбор и просмотр статистики

Дополнительные расширения

Просмотр и анализ журнала сервера

## 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 посмотреть как из самой базы (см. темы «Базы данных» и «Табличные пространства»), так из ОС (`du`). Размер доступного дискового пространства надо смотреть в ОС (`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.

<http://www.postgresql.org/docs/current/static/monitoring-ps.html>

<http://www.postgresql.org/docs/current/static/diskusage.html>

## Настройки процесса stats collector

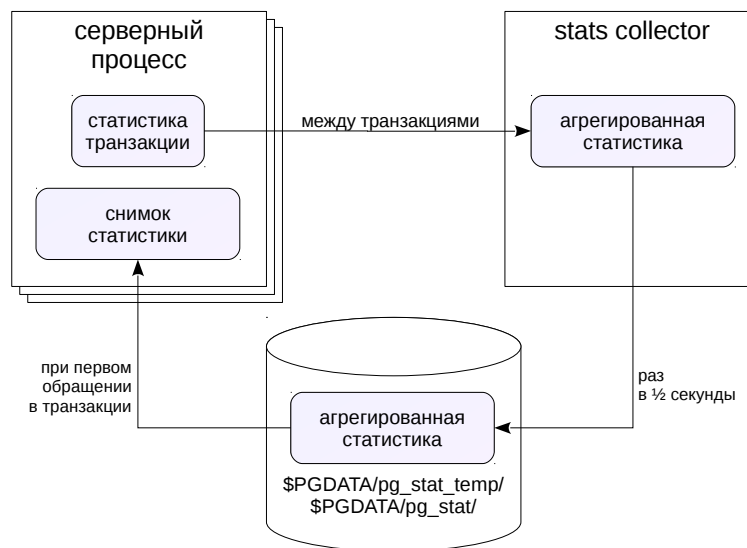
<i>статистика</i>	<i>параметр</i>
текущие активности сервера и фоновых процессов	track_activities включен
обращения к таблицам и индексам (доступы, затронутые строки)	track_counts включен, нужен для автоочистки
обращения к страницам	track_io_timing выключен
вызовы пользовательских функций	track_functions выключен

Внутренние средства мониторинга PostgreSQL включают статистику и журнал сервера.

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.

<http://www.postgresql.org/docs/current/static/monitoring-stats.html>





Каждый серверный процесс собирает необходимую статистику в рамках каждой выполняемой транзакции. Затем эта статистика передается процессу-коллектору. Коллектор собирает и агрегирует статистику со всех серверных 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 серверный 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.

Сказанное выше не относится к информации о текущих активностях сервера и фоновых процессов — она всегда актуальна.

При останове сервера коллектор сбрасывает статистику в постоянные файлы в каталог `$PGDATA/pg_stat`.

## Расширения в поставке

<code>pg_stat_statements</code>	статистика по запросам
<code>pgstattuple</code>	статистика по версиям строк
<code>pg_buffercache</code>	состояние буферного кэша

## Другие расширения

<code>pg_stat_plans</code>	статистика по планам запросов
<code>pg_stat_kcache</code>	статистика по процессору и вводу-выводу
<code>pg_qualstats</code>	статистика по предикатам
...	

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

## Настройки

### *информация*

сообщения определенного уровня  
время выполнения длинных команд  
время выполнения команд  
имя приложения  
контрольные точки  
подключения и отключения  
длинные ожидания  
текст выполняемых команд  
использование временных файлов  
...

### *параметр*

log\_min\_messages  
log\_min\_duration\_statement  
log\_duration  
application\_name  
log\_checkpoints  
log\_(dis)connections  
log\_lock\_waits  
log\_statement  
log\_temp\_files

В журнал 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.

## Встроенные средства

<i>настройка</i>	<i>параметр</i>
включение сбора сообщений	logging_collector = on
формат журнала	log_destination = stderr, csvlog, syslog...
перезаписывать ли файлы	log_truncate_on_rotation = on
маска имени файла	log_filename
время ротации, мин	log_rotation_age
комбинируя маску файла и время ротации, получаем разные схемы:	
postgresql-%H.log, 60	24 файла в сутки
postgresql-%a.log, 1440	7 файлов в неделю

## Внешние средства

например, Apache rotatelog  
pg\_ctl start | rotatelog имя\_файла 86400 -n 24

8

Если запускать сервер так, как это демонстрировалось в теме «Установка PostgreSQL», то вывод будет собираться в одном файле, который может вырасти до огромных размеров. Поэтому обычно используется та или иная схема ротации журналов.

<http://www.postgresql.org/docs/current/static/logfile-maintenance.html>

Можно воспользоваться встроенными средствами, которые настраиваются несколькими параметрами, часть из которых приведена на слайде.

logging\_collector включает фоновый процесс, собирающий журнальные сообщения и перенаправляющий их в файлы.

log\_destination определяет формат сообщений.

log\_filename задает маску имени файла, в которой могут использоваться спецсимволы даты и времени.

log\_rotation\_age задает время переключения на следующий файл в минутах.

log\_truncate\_on\_rotation перезаписывает уже существующие файлы.

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

<http://www.postgresql.org/docs/current/static/runtime-config-logging.html>

Альтернативно можно воспользоваться внешними программами ротации, например rotatelog.

## Средства операционной системы

grep, awk...

## Специальные средства анализа

pgBadger

## Системы мониторинга PostgreSQL

PostgreSQL Workload Analyzer (PoWA)

Open PostgreSQL Monitoring (OPM)

## Универсальные системы мониторинга

Nagios, Munin, Zabbix, Cacti...

Анализировать журналы можно по-разному.

Можно искать определенную информацию средствами ОС, скриптами или специальными средствами анализа.

Можно воспользоваться системами мониторинга, которые позволяют собирать информацию, рисовать графики, присылать уведомления при определенных событиях и т. п. Такие системы есть как непосредственно для PostgreSQL, так и универсальные (в которые PostgreSQL подключается как один из возможных источников информации).

<http://www.postgresql.org/docs/current/static/logfile-maintenance.html>



Мониторинг заключается в контроле работы сервера как со стороны операционной системы, так и со стороны PostgreSQL.

Для этого используется собранная статистика и анализ журнала сервера.

1. В базе данных DB14 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 сервера.

```
# create database db14;
# \c db14
# create table t(n numeric);
# insert into t select 1 from generate_series(1,1000);
# delete from t;

# select * from pg_stat_all_tables where relid='t'::regclass;
-- n_tup_ins = 1000, n_tup_del = 1000
-- n_live_tup = 0, n_dead_tup = 1000

# vacuum;
# select * from pg_stat_all_tables where relid='t'::regclass;
-- n_dead_tup = 0 (убраны при очистке)
-- vacuum_count = 1

# insert into t values (1), (2);
2# begin;
2# update t set n=10 where n=1;
3# begin;
3# update t set n=200 where n=2;
2# update t set n=20 where n=2;
-- второй заблокирован третьим
3# update t set n=100 where n=1;
-- третий заблокирован вторым - взаимоблокировка

$ tail ~postgres/logfile
```