Avito

Мониторинг PostgreSQL в Авито.

Дмитрий Вагин, 2017

О чём это всё?!

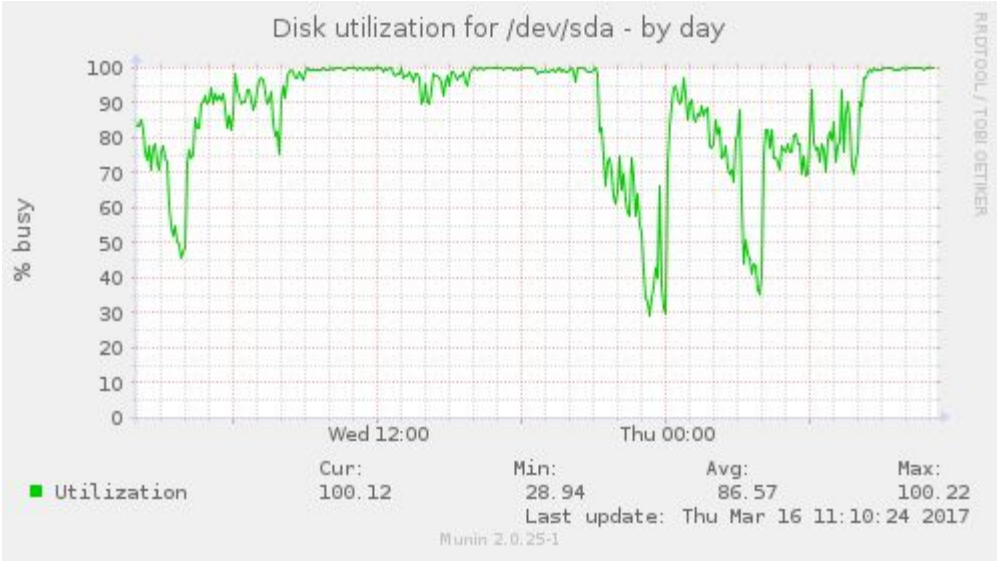
Я Дмитрий - разработчик баз данных в Авито.
Я расскажу про свой опыт сбора статистики баз данных PostgreSQL.

Зачем всё это?!

Очень хочется:

- знать что сломалось
- знать почему сломалось
- знать когда сломается в следующий раз

Как проявляются проблемы

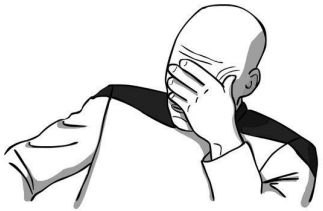


Куда смотреть?!

> *«PostgreSQL's statistics collector is a subsystem that supports collection and reporting of information about server activity.»*

Из pg_stat_database мы узнали что с момента последнего сброса статистики

- было совершено 14 миллиардов 066 миллионов 649 тысячи 489 транзакции
- 128 миллиардов 702 миллиона 052 тысячи 186 блоков было прочитано



Что делать?!

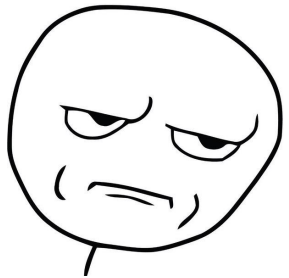
*pg_stat_** и *pg_statio_** практически бесполезны без исторических данных!

```
insert into maintenance.pg_stat_all_tables
select * from pg_stat_all_tables;
```

680kb - один срез (≈2k таблиц)

680kb * 60min * 24h ≈ 1GB per day

1 гигабайт чтобы иметь статистику
по таблицам в одной базе за один день?!



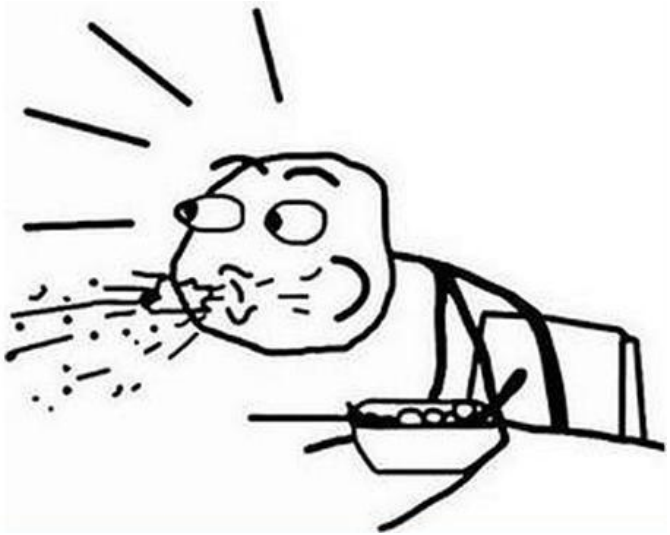
Что у нас есть...

- есть один большой инстанс (кластер), и 5 стендбаев, один для работы, остальные для резерва.
- есть ещё 4 инстанса поменьше, с кучкой баз разной степени нагруженности и важности + по standby на каждый инстанс
- 3 инстанса для логических реплик разных таблиц и материализованных представлений + по резервному инстансу для каждой из них

Что у нас есть...

Итого:

- ≈15 инстансов
- ≈30 баз
- ≈30000 таблиц
- ≈60000 индексов
- ≈1000000 различных метрик



Что делать!?!?!?!?!?!??!!111111

Grafana

graphite

StatsD

Магия


pg_statio_user_tables
Zoom Out
Last 6 hours
instance
db
database
avito_new
schema
All
table
All
limit
10
pg_statio_user_tables by schema
300 MBps
200 MBps
100 MBps
0 Bps
pg_statio_user_tables - hit/read
200 MBps
150 MBps
100 MBps
50 MBps
0 Bps
pg_statio_user_tables table/index
200 MBps
150 MBps
100 MBps
50 MBps
0 Bps
Tuples fetched per table
750 K
500 K
250 K
0
08:30 08:40 08:50 09:00 09:10 09:20 09:30 09:40 09:50 10:00 10:10 10:20 10:30 10:40 10:50 11:00 11:10 11:20 11:30 11:40 11:50 12:00 12:10 12:20 12:30 12:40 12:50 13:00 13:10 13:20 13:30 13:40 13:50 14:00 14:10 14:20
+ADD ROW

Как такое сделать?

1. настраиваем brubeck + graphite + grafana
2. пишем крон или демон который раз в минуту или n-секунд забирает данные из pg_stat_* и pg_statio_* и скидывает в brubeck
3. настраиваем dashboards в графане
4. PROFIT

Про крон или демон

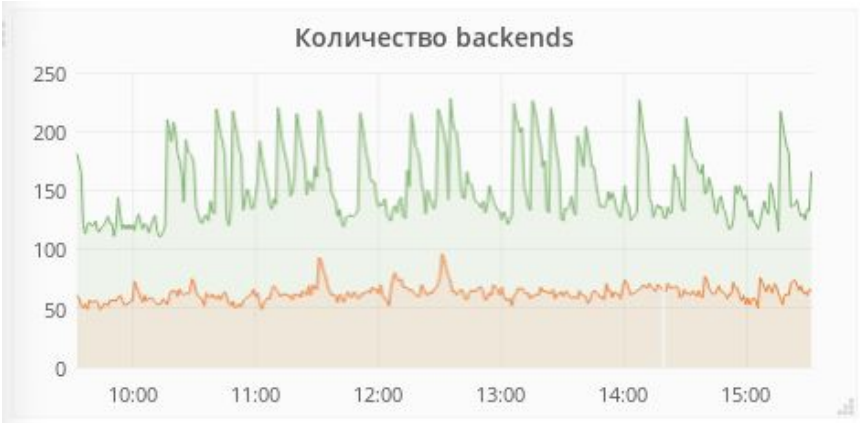
- можно попробовать найти готовое
- можно попробовать plugin collectd-postgresql
- можно сделать своё :)

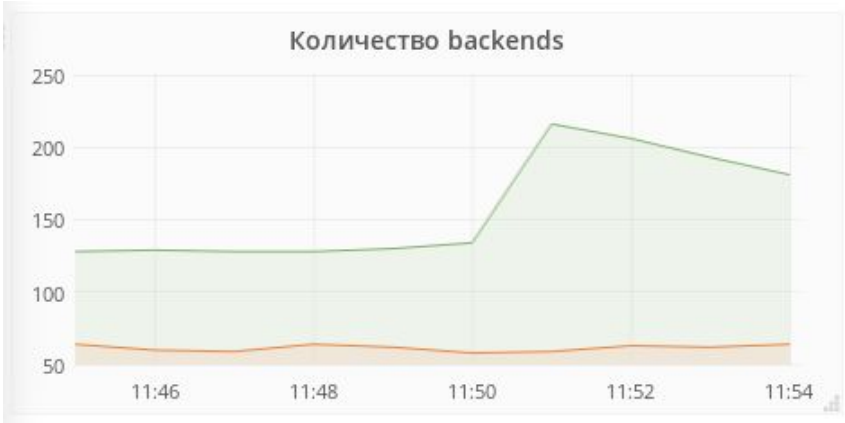
Важно разделять типы метрик:

- метрика отображающая текущее значение — например pg_stat_database.num_backends
- увеличивающийся счётчик — практически все остальные метрики

Про крон или демон

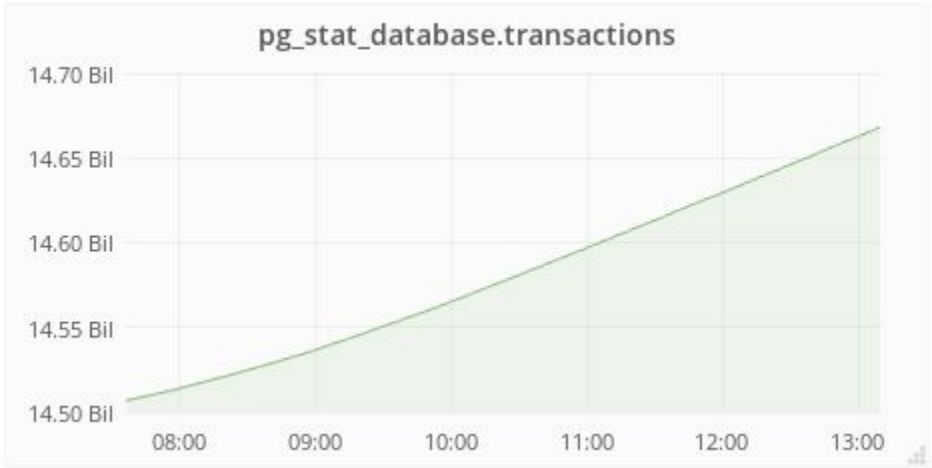
pg_stat_database.num_backends

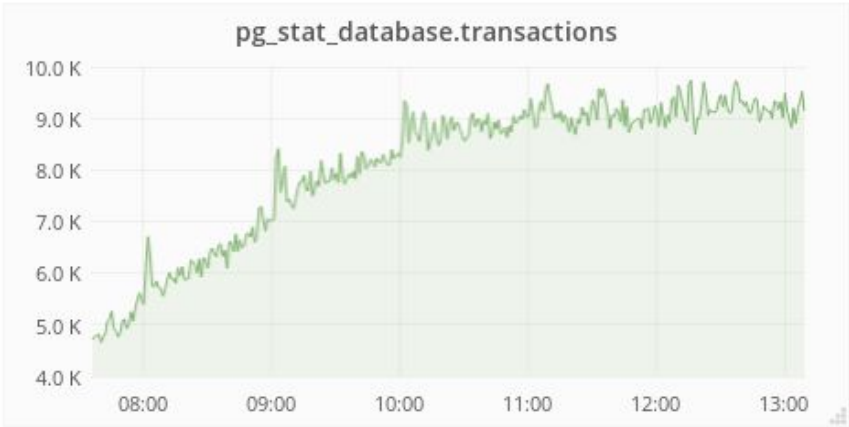




Про крон или демон

pg_stat_database.xact_commit





perSecond()

Про крон или демон

- g - Gauges
- c - Meters
- C - Counters
- h - Histograms
- ms - Timers (in milliseconds)

https://github.com/github/brubeck
https://github.com/etsy/statsd/blob/master/docs/metric_types.md

Документация противоречивая, поэтому смотрим в исходники
https://github.com/github/brubeck/blob/master/src/metric.c

pg_stat_statements

pg_stat_statements

В имя метрики невозможно засунуть весь запрос. Поэтому запрос храним в redis’e, а в имени метрики идёт хэш от запроса.

Для получения текста запроса используем самописную функцию в графите, запрос к графиту выглядит так:

aliasByHash*(aliasByNode(keepLastValue(limit(sortByTotal(metric.namespace.*.$column), $limit), 1), 5), 'sql_query')*

pg_stat_statements

Из хорошего:

- мы видим самые часто выполняемые запросы
- мы видим запросы которые читают с диска shared_blks_read
- можем видеть динамику(!) времени выполнения запроса

Из плохого:

- очень много уникальных запросов, поэтому удаляем старое
- нужно какое-то дополнительное хранилище (redis)
- проблемы с отображением больших запросов в графане

PG Metricus

Надо смотреть за динамикой времени выполнения куска большой plpgsql процедуры.

- добавить *raise notice* и парсить логи
- складывать в таблицу и анализировать её
- а что если….. кидать в канал через механизм NOTIFY, а оттуда отправлять в графит….

PG Metricus

```
x1 = clock_timestamp();
…. your long running query
x2 = clock_timestamp();

perform pg_notify('test', format(E'%s.%s:%s|%s\n',
   'db.sql.metric',
   'get_val_hstore_duration',
   extract(millisecond from (x2 - x1))::bigint::text,
   'ms' ));
```

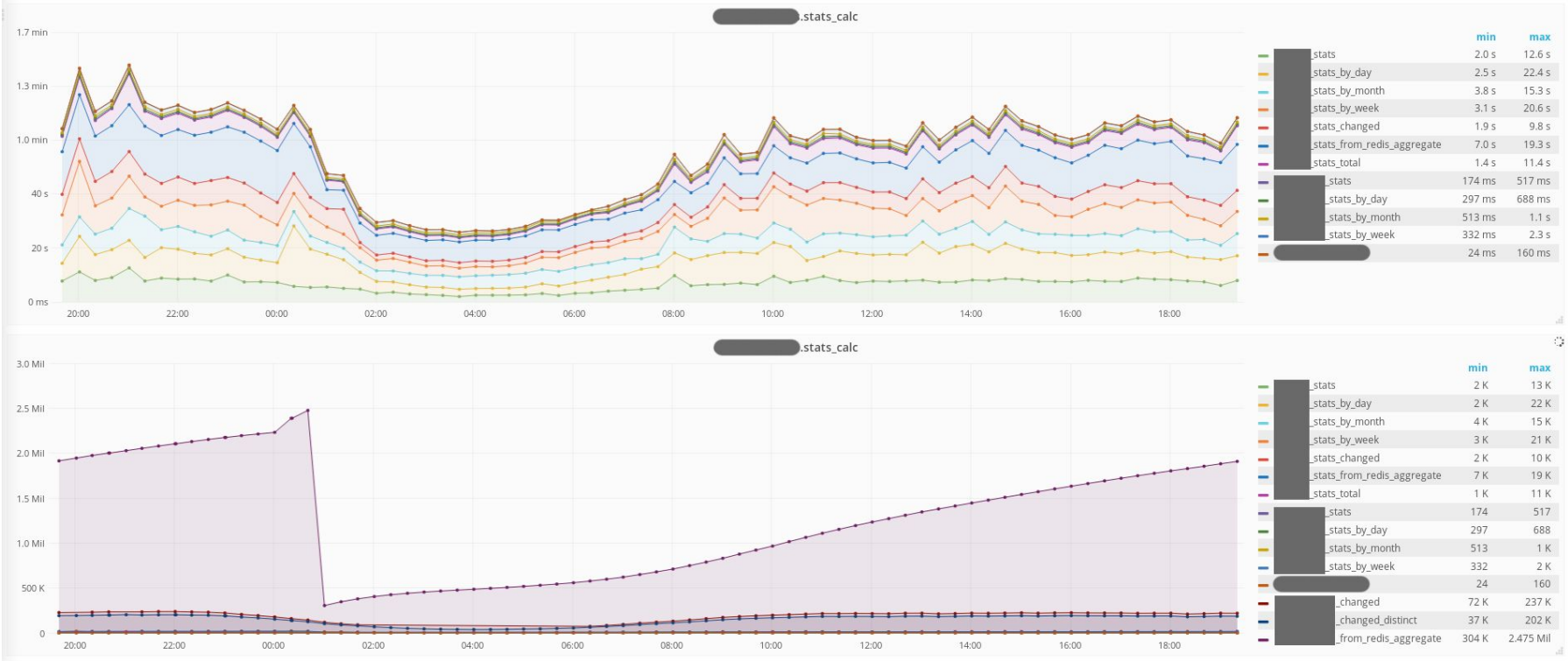
PG Metricus

```
curs = conn.cursor()
curs.execute("LISTEN {0};".format(args.channel))

addr = (args.socket_host, args.socket_port)
sock = socket(AF_INET, SOCK_DGRAM)

while not checkStop():
    if select.select([conn],[],[],5) != ([],[],[]):
        conn.poll()
        while conn.notifies:
            notification = conn.notifies.pop(0)
            sock.sendto(notification.payload.encode('utf-8'), addr)
```

PG Metricus


[redacted].stats_calc
1.7 min
1.3 min
1.0 min
40 s
20 s
0 ms
20:00
22:00
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
min
max
[redacted]_stats 2.0 s 12.6 s
[redacted]_stats_by_day 2.5 s 22.4 s
[redacted]_stats_by_month 3.8 s 15.3 s
[redacted]_stats_by_week 3.1 s 20.6 s
[redacted]_stats_changed 1.9 s 9.8 s
[redacted]_stats_from_redis_aggregate 7.0 s 19.3 s
[redacted]_stats_total 1.4 s 11.4 s
[redacted]_stats 174 ms 517 ms
[redacted]_stats_by_day 297 ms 688 ms
[redacted]_stats_by_month 513 ms 1.1 s
[redacted]_stats_by_week 332 ms 2.3 s
[redacted] 24 ms 160 ms
[redacted].stats_calc
3.0 Mil
2.5 Mil
2.0 Mil
1.5 Mil
1.0 Mil
500 K
0
20:00
22:00
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
min
max
[redacted]_stats 2 K 13 K
[redacted]_stats_by_day 2 K 22 K
[redacted]_stats_by_month 4 K 15 K
[redacted]_stats_by_week 3 K 21 K
[redacted]_stats_changed 2 K 10 K
[redacted]_stats_from_redis_aggregate 7 K 19 K
[redacted]_stats_total 1 K 11 K
[redacted]_stats 174 517
[redacted]_stats_by_day 297 688
[redacted]_stats_by_month 513 1 K
[redacted]_stats_by_week 332 2 K
[redacted] 24 160
[redacted]_changed 72 K 237 K
[redacted]_changed_distinct 37 K 202 K
[redacted]_from_redis_aggregate 304 K 2.475 Mil

PG Metricus

Подробнее можно ознакомиться в статье на хабре

https://habrahabr.ru/company/avito/blog/323900/

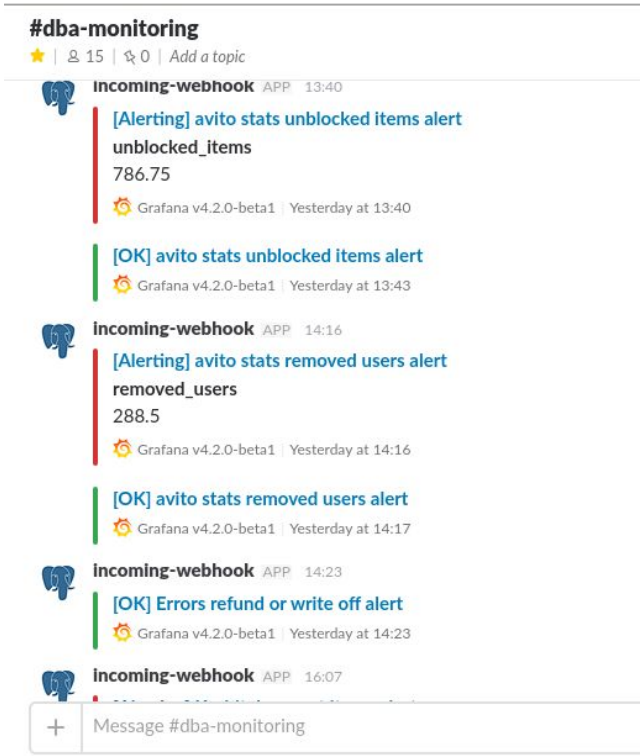
https://github.com/avito-tech/pg_metricus_python

Алерты

Мы используем

- grafana 4+ имеет встроенный алертинг
- moira

Нотификации в Slack.



Ну вроде всё...

Вопросы?

Предложения?

Комментарии?