



# Диагностика и мониторинг PostgreSQL

## Инструмент PGARM



Денис Леонтьев  
Ведущий инженер  
Центр технической поддержки ФОРС

Андрей Пауков  
Ведущий инженер  
Центр технической поддержки ФОРС

PGConf.Russia 2024



# План презентации

1. Цели проактивного мониторинга и диагностики
2. Обзор доступных инструментов в ванильном PostgreSQL
3. Пример «медленный запрос» - диагностика штатными средствами
4. Выводы
5. Обзор инструмента PGARM
  - 5.1 Пример «медленный запрос» - решение ФОРС
  - 5.2 Пример «ресурсоемкий запрос» - решение ФОРС
  - 5.3 Демонстрация - трассировка сессии

# Цели проактивного мониторинга и диагностики



Проактивно увидеть проблему и предотвратить ее

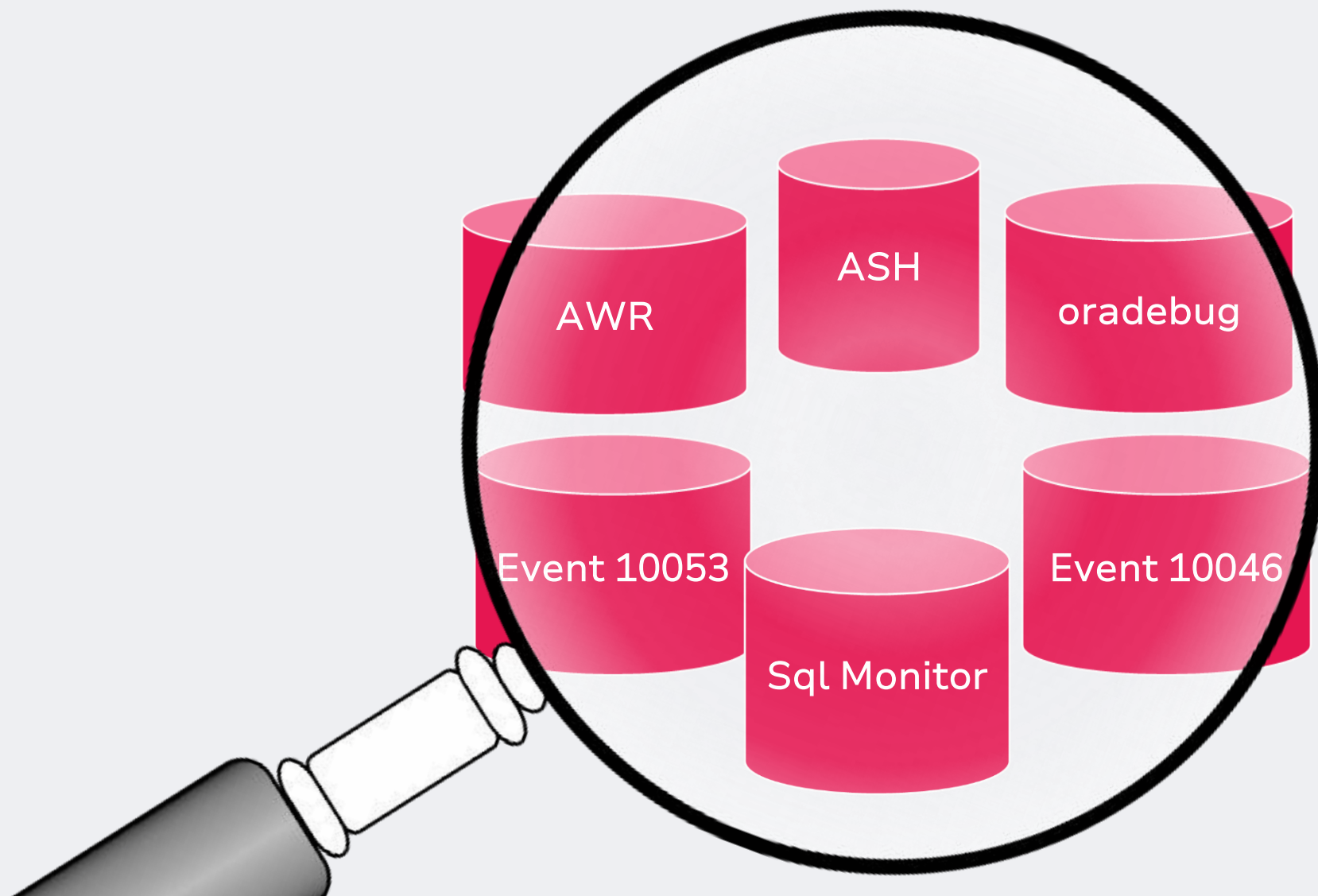


Если проблема уже случилась:

Как можно быстрее восстановить работоспособность системы

Определить причину и принять меры, чтобы не было повторения

# Где мои привычные инструменты?



# Что предлагает ванильный PostgreSQL



Представление  
«pg\_stat\_activity»  
(«оперативный  
мониторинг» )

Расширение  
«pg\_stat\_statements»



Накопление истории  
ожиданий

Накопление истории  
запросов

Графические  
инструменты

# Популярные расширения

## Собираем аналоги Oracle ASH и AWR:

- «pg\_stat\_statements»
- «pg\_wait\_sampling»
- «pg\_stat\_kcache»
- «pg\_profile»

## Иногда что-то полезное можно найти в логах:

- «pgBadger»
- «auto\_explain»

# Популярные графические инструменты

## Графические инструменты / панели /дашбоарды:

- pgAdmin
- ASH Viewer / PASH-Viewer
- PGWatch



- Позволяют наглядно увидеть общую картину работы экземпляра
- Увидеть проблемные запросы



- Большая часть графических продуктов мониторинга ориентирована на слабо нагруженные системы
- «Подвисают» при интенсивной нагрузке на СУБД
- Ограниченный набор показателей
- Не подходят для углубленной диагностики конкретной проблемы

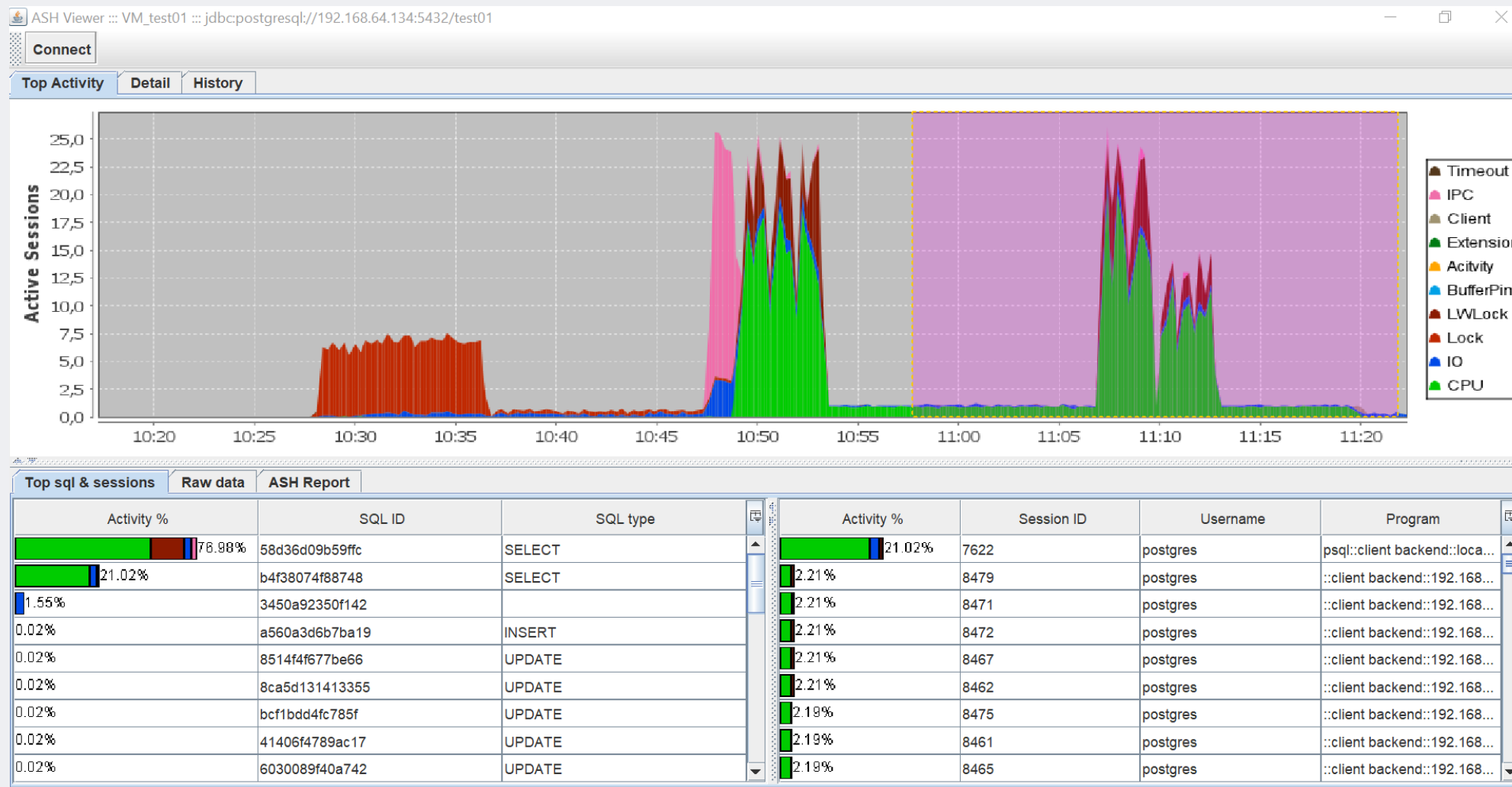
# Пример «медленный запрос» - описание

- Запрос из двух таблиц (master-detail) с фильтром по нескольким полям

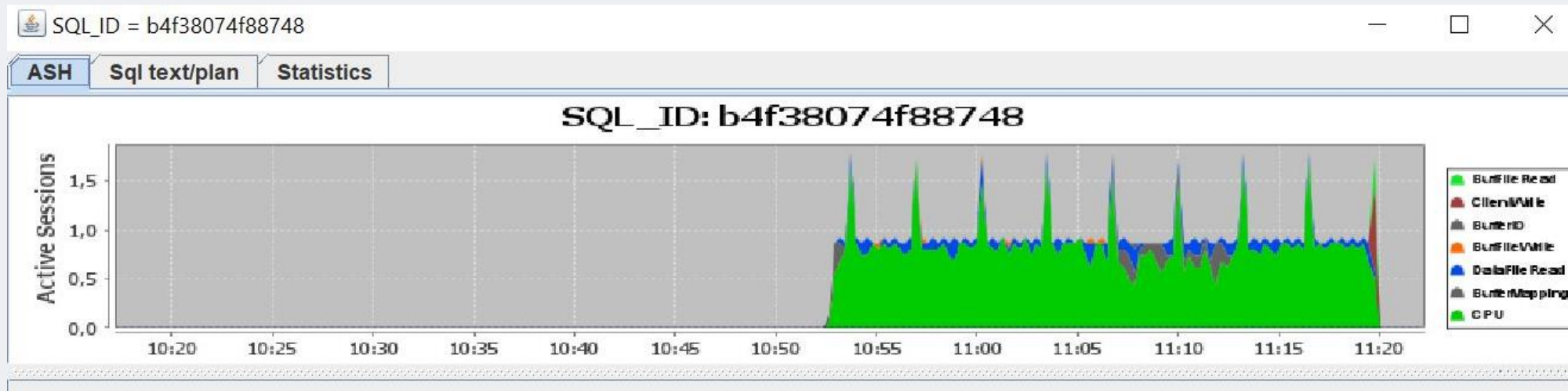
```
SELECT /* Long running query */ h.name, h.short_description, l.content, l.sort_field
FROM t_document_headers h, t_document_lines l
WHERE l.header_id=h.header_id AND
      h.security_level = 0 AND
      EXISTS (SELECT '1' FROM t_document_lines ll WHERE ll.header_id=h.header_id AND
              ll.security_level>0)
ORDER BY h.header_id, l.sort_field;
```

- После установки новой версии приложения и/или синхронизации данных с внешней системой запрос стал выполняться медленно
- В нашей БД заранее установлен и настроен «джентельменский набор» расширений, которые должны нам помочь понять в чем проблема

# Пример «медленный запрос» - ASH Viewer



# Пример «медленный запрос» - ASH Viewer



The screenshot shows the SQL Developer interface. At the top, the connection is identified as 'SQL\_ID = b4f38074f88748'. Below this, there are three tabs: 'ASH', 'Sql text/plan', and 'Statistics'. The 'Sql text/plan' tab is selected, displaying the following SQL query:

```
SELECT /* Long running query */ h.name, h.short_description, l.content, l.sort_field
FROM t_document_headers h,
     t_document_lines l
WHERE l.header_id=h.header_id AND
      h.security_level = 0 AND
      EXISTS (SELECT '1' FROM t_document_lines ll WHERE ll.header_id=h.header_id AND ll.security_level>0)
ORDER BY h.header_id l.sort_field;
```

Below the query, the 'PHV: 0' (Plan Hash Value) is shown. The execution plan is displayed in a table with the following columns: Description, Object name, Object type, Cost, Cardinality, and Bytes.

| Description | Object name | Object type | Cost | Cardinality | Bytes |
|-------------|-------------|-------------|------|-------------|-------|
|             |             |             |      |             |       |

# Пример «медленный запрос» - pg\_stat\_activity

Tue 05 Mar 2024 11:03:15 AM MSK (every 10s)

| current_time       | pid  | leader_pid | wait_event       | backend_type    | state  | query_start                   | state_change                  |
|--------------------|------|------------|------------------|-----------------|--------|-------------------------------|-------------------------------|
| 11:03:15.365358+03 | 5154 |            | DataFileRead     | client backend  | active | 2024-03-05 11:03:12.529928+03 | 2024-03-05 11:03:12.529931+03 |
| 11:03:15.365358+03 | 7571 | 5154       | MessageQueueSend | parallel worker | active | 2024-03-05 11:03:12.529928+03 | 2024-03-05 11:03:12.559396+03 |
| 11:03:15.365358+03 | 7572 | 5154       | MessageQueueSend | parallel worker | active | 2024-03-05 11:03:12.529928+03 | 2024-03-05 11:03:12.570997+03 |

(3 rows)

Tue 05 Mar 2024 11:03:25 AM MSK (every 10s)

| current_time      | pid  | leader_pid | wait_event | backend_type   | state  | query_start                   | state_change                  |
|-------------------|------|------------|------------|----------------|--------|-------------------------------|-------------------------------|
| 11:03:25.43785+03 | 5154 |            |            | client backend | active | 2024-03-05 11:03:12.529928+03 | 2024-03-05 11:03:12.529931+03 |

(1 row)

■ ■ ■

Tue 05 Mar 2024 11:17:50 AM MSK (every 10s)

| current_time       | pid  | leader_pid | wait_event  | backend_type   | state  | query_start                   | state_change                  |
|--------------------|------|------------|-------------|----------------|--------|-------------------------------|-------------------------------|
| 11:17:50.412338+03 | 5154 |            | ClientWrite | client backend | active | 2024-03-05 11:03:12.529928+03 | 2024-03-05 11:03:12.529931+03 |

(1 row)

Видим что работа идет, но почему запрос выполняется **медленно** ?

# Пример «медленный запрос» - pg\_wait\_sampling\_profile

```
select * from pg_wait_sampling_profile
```

| Tue 05 Mar 2024 11:10:54 AM MSK (every 10s) |            |                         |                      |        | Tue 05 Mar 2024 11:17:37 AM MSK (every 10s) |            |                         |                      |        |
|---------------------------------------------|------------|-------------------------|----------------------|--------|---------------------------------------------|------------|-------------------------|----------------------|--------|
| pid                                         | event_type | event                   | queryid              | count  | pid                                         | event_type | event                   | queryid              | count  |
| 5154                                        | Client     | ClientWrite             | -5201878321441831011 | 1232   | 5154                                        | Client     | ClientWrite             | -5201878321441831011 | 1232   |
| 5154                                        | IPC        | HashGrowBucketsAllocate | -5201878321441831011 | 1      | 5154                                        | IPC        | HashGrowBucketsAllocate | -5201878321441831011 | 1      |
| 5154                                        | IO         | BufFileRead             | -5201878321441831011 | 8      | 5154                                        | IO         | BufFileRead             | -5201878321441831011 | 8      |
| 5154                                        | IO         | BufFileWrite            | -5201878321441831011 | 825    | 5154                                        | IO         | BufFileWrite            | -5201878321441831011 | 865 V  |
| 5154                                        | IPC        | BufferIO                | -5201878321441831011 | 1      | 5154                                        | IPC        | BufferIO                | -5201878321441831011 | 1      |
| 5154                                        | Client     | ClientRead              | 0                    | 243312 | 5154                                        | Client     | ClientRead              | 0                    | 243312 |
| 5154                                        | IPC        | MessageQueueReceive     | -5201878321441831011 | 479    | 5154                                        | IPC        | MessageQueueReceive     | -5201878321441831011 | 479    |
| 5154                                        | LWLock     | BufferMapping           | -5201878321441831011 | 7      | 5154                                        | LWLock     | BufferMapping           | -5201878321441831011 | 7      |
| 5154                                        | IPC        | HashGrowBucketsElect    | -5201878321441831011 | 2      | 5154                                        | IPC        | HashGrowBucketsElect    | -5201878321441831011 | 2      |
| 5154                                        | IO         | DataFileRead            | -5201878321441831011 | 4909   | 5154                                        | IO         | DataFileRead            | -5201878321441831011 | 8188 V |
| (10 rows)                                   |            |                         |                      |        | (10 rows)                                   |            |                         |                      |        |

Счетчики обновляются в online режиме  
Запрос не завис, работа идет

Но почему так  
долго ?

# Пример «медленный запрос» - pg\_stat\_statements и pg\_profile

```
select *from pg_stat_statements where query like '%Long running query%';
```

Статистика обновляется в конце соответствующей фазы **и только при успешном завершении** этой фазы

## Top SQL by execution time

| Query ID                                           | Database | User     | Exec (s) | %Total | I/O time (s) |       | Rows    | Execution times (ms) |           |           |        | Executions |
|----------------------------------------------------|----------|----------|----------|--------|--------------|-------|---------|----------------------|-----------|-----------|--------|------------|
|                                                    |          |          |          |        | Read         | Write |         | Mean                 | Min       | Max       | StdErr |            |
| <a href="#">2a0aadb1bd70c413</a> ⓘ<br>[b4a64459d1] | test01   | postgres | 878.55   | 97.29  | 177.61       |       | 4490400 | 897975.22            | 897975.22 | 897975.22 |        | 1          |

Статистика по запросу будет доступна в pg\_profile для последующего анализа **после завершения** запроса и **после создания** «снимка»

# Пример «медленный запрос» - auto\_explain

```
2024-03-05 11:17:51.089 MSK [5154] LOG: duration: 878558.684 ms plan:
Query Text: SELECT /* Long running query */ h.name, h.short_description, l.content, l.sort_field
FROM t_document_headers h,
     t_document_lines l
WHERE l.header_id=h.header_id AND
      h.security_level = 0 AND
      EXISTS (SELECT '1' FROM t_document_lines ll WHERE ll.header_id=h.header_id AND ll.security_level>0)
ORDER BY h.header_id, l.sort_field;
Sort (cost=639956.71..642288.12 rows=932564 width=80)
Sort Key: h.header_id, l.sort_field
-> Nested Loop (cost=1000.27..464609.05 rows=932564 width=80)
    Join Filter: (h.header_id = l.header_id)
    -> Nested Loop Semi Join (cost=1000.27..152200.24 rows=1 width=47)
        Join Filter: (h.header_id = ll.header_id)
        -> Index Scan using t_document_headers_pkey on t_document_headers h (cost=0.27..44.42 rows=180 width=43)
            Filter: (security_level = 0)
        -> Materialize (cost=1000.00..152153.13 rows=1 width=4)
            -> Gather (cost=1000.00..152153.12 rows=1 width=4)
                Workers Planned: 2
                -> Parallel Seq Scan on t_document_lines ll (cost=0.00..151153.02 rows=1 width=4)
                    Filter: (security_level > 0)
            -> Seq Scan on t_document_lines l (cost=0.00..195838.36 rows=9325636 width=41)
```

- Информация о запросе отображается **только после** его выполнения
- Обычно, это расширение на высоконагруженных промышленных системах не используют или используют в целях диагностики

# Пример «медленный запрос» - pg\_show\_plans

- [https://github.com/cybertec-postgresql/pg\\_show\\_plans](https://github.com/cybertec-postgresql/pg_show_plans)
- Установка из исходников
- Влияние на производительность: "In overall approximately 15% performance penalty."

```
test01=# SELECT * FROM pg_show_plans where pid=5154 \gx
-[ RECORD 1 ]-----
pid      | 5154
level    | 0
userid   | 10
dbid     | 59450
plan     | Sort (cost=639956.71..642288.12 rows=932564 width=80)
          |   Sort Key: h.header_id, l.sort_field
          |   -> Nested Loop (cost=1000.27..464609.05 rows=932564 width=80)
          |       Join Filter: (h.header_id = l.header_id)
          |       -> Nested Loop Semi Join (cost=1000.27..152200.24 rows=1 width=47)
          |           Join Filter: (h.header_id = ll.header_id)
          |           -> Index Scan using t_document_headers_pkey on t_document_headers h (cost=0.27..44.42 rows=180 width=43)
          |               Filter: (security_level = 0)
          |           -> Materialize (cost=1000.00..152153.13 rows=1 width=4)
          |               -> Gather (cost=1000.00..152153.12 rows=1 width=4)
          |                   Workers Planned: 2
          |                   -> Parallel Seq Scan on t_document_lines ll (cost=0.00..151153.02 rows=1 width=4)
          |                       Filter: (security_level > 0)
          |   -> Seq Scan on t_document_lines l (cost=0.00..195838.36 rows=9325636 width=41)
```

# Пример «медленный запрос» - pg\_query\_state

```
-> Nested Loop (cost=1000.70..150096.22 rows=515460 width=80) (Current loop: actual time=8.231..29352.549 rows=4460010, loops=1)
  Buffers: shared hit=44952 read=153560 dirtied=1
  I/O Timings: read=50910.371
  -> Nested Loop Semi Join (cost=1000.27..130983.17 rows=1 width=47) (Current loop: actual time=4.227..21828.983 rows=4460010, loops=1)
    Join Filter: (h.header_id = ll.header_id)
    Buffers: shared hit=389 read=103111 dirtied=1
    I/O Timings: read=46355.604
    -> Index Scan using t_document_headers_pkey on t_document_headers h (cost=0.27..42.84 rows=90 width=43) (Current loop: actual time=0.005..20.398 rows=25355 loops=902) (Current loop: actual time=0.005..20.398 rows=25355 loops=902)
      Filter: (security_level = 0)
      Buffers: shared hit=185 read=19
      I/O Timings: read=2.890
    -> Materialize (cost=1000.00..130938.98 rows=1 width=4) (actual time=0.005..20.398 rows=25355 loops=902) (Current loop: actual time=0.005..20.398 rows=25355 loops=902)
```

# Пример «медленный запрос» - pg\_query\_state

- [https://github.com/postgrespro/pg\\_query\\_state](https://github.com/postgrespro/pg_query_state)
- Позволяет увидеть актуальный план запроса и статистику его выполнения (включая количество строк, затраченное время и статистику по буферам)
- Установка из исходников, готового пакета нет
- Требуется внесение изменений в ядро СУБД с последующей компиляцией и пересборкой модифицированной версии ядра
- Информация доступна только в момент выполнения запроса, история не сохраняется

# Пример «медленный запрос» - стек процесса (gdb)

```
(gdb)
(gdb) bt
#0 0x0000000000686e14 in ExecInterpExpr (state=0x24b9d48, econtext=0x248d3a8, isnull=<optimized out>) at execExprInterp.c:742
#1 0x00000000006b78e9 in ExecEvalExprSwitchContext (isNull=0x7ffcf9547bc7, econtext=0x248d3a8, state=0x24b9d48)
   at ../../src/include/executor/executor.h:342
#2 ExecQual (econtext=0x248d3a8, state=0x24b9d48) at ../../src/include/executor/executor.h:411
#3 ExecNestLoop (pstate=<optimized out>) at nodeNestloop.c:214
#4 0x00000000006b9ede in ExecProcNode (node=0x248d298) at ../../src/include/executor/executor.h:260
#5 ExecSort (pstate=0x248d088) at nodeSort.c:108
#6 0x000000000068b503 in ExecProcNode (node=0x248d088) at ../../src/include/executor/executor.h:260
#7 ExecutePlan (execute_once=<optimized out>, dest=0x24a6260, direction=<optimized out>, numberTuples=0, sendTuples=<optimized out>, operation=CMD_SELECT,
   use_parallel_mode=<optimized out>, planstate=0x248d088, estate=0x248ce18) at execMain.c:1551
#8 standard_ExecutorRun (queryDesc=0x23e62b8, direction=<optimized out>, count=0, execute_once=<optimized out>) at execMain.c:361
#9 0x00007f9d38628dd5 in pgss_ExecutorRun () from /usr/pgsql-14/lib/pg_stat_statements.so
#10 0x00007f9d382159bd in pgsk_ExecutorRun (queryDesc=0x23e62b8, direction=ForwardScanDirection, count=0, execute_once=<optimized out>)
   at pg_stat_kcache.c:1026
#11 0x00007f9d3800fdbd in pgsp_ExecutorRun (queryDesc=0x23e62b8, direction=ForwardScanDirection, count=0, execute_once=<optimized out>)
   at pg_show_plans.c:484
#12 0x00000000007fd28b in PortalRunSelect (portal=0x242bf98, forward=<optimized out>, count=0, dest=<optimized out>) at pquery.c:921
#13 0x00000000007fe59d in PortalRun (/
```

Стек backend процесса, выполняющего запрос (gdb bt).

- В стеке видны отдельные операции из плана запроса, но **полный план** здесь, к сожалению, **не посмотреть**
- По названиям функций можно догадаться, что они выполняют, но **для точной интерпретации нужны знания разработчика**

# Пример «медленный запрос» - семплирование стека (perf)

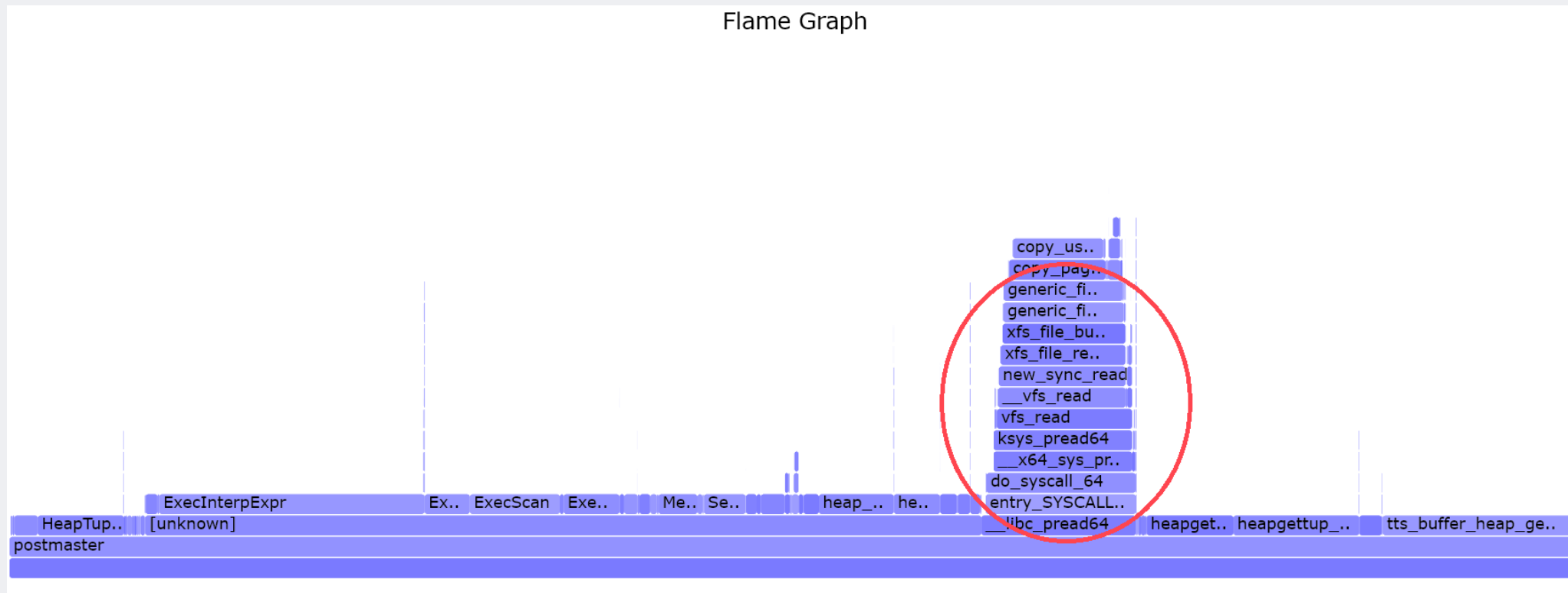
```
# Samples: 5K of event 'cpu-clock:pppH'
# Event count (approx.): 59050504460
#
# Children      Self  Command      Shared Object      Symbol
# .....
#
28.12%      0.00%  postmaster  [unknown]          [.] 0000000000000000
|
---0
|
| 16.85% --ExecInterpExpr
|
| 3.37% --ExecScan
|
| 2.87% --ExecNestLoop
|
| 2.48% --MemoryContextReset
|
| 1.08% --int4eq
|
| 0.63% --slot_getsomeattrs_int
|
16.88%      16.83% postmaster postgres          [.] ExecInterpExpr
|
| 16.80% --0
|
|      ExecInterpExpr
|
12.56%      12.45% postmaster postgres          [.] tts_buffer_heap_getsomeattrs
|
| ---tts_buffer_heap_getsomeattrs
|
10.95%      10.90% postmaster postgres          [.] heapgettup_pagemode
|
| 8.02% --heapgettup_pagemode
```

Семплирование стека backend процесса в течение 1 минуты («perf record»)

- Основной потребитель ресурсов CPU – «ExecInterpExpr»
- В плане запроса есть Nested Loops

Поможет ли эта информация диагностировать проблему?

# Пример «медленный запрос» - семплирование стека (Flame Graph)



Семплированные стеки, визуализированные в Flame Graph

Видны довольно интенсивные дисковые чтения (проверить план выполнения запроса, наличие индексов ?)

# Пример «медленный запрос» - СТАТИСТИКА ОПТИМИЗАТОРА

```
test01=# SELECT relname, reltuples FROM pg_class WHERE relname IN ('t_document_headers', 't_document_lines');
 relname      | reltuples
```

```
-----+-----
 t_document_headers |      5
 t_document_lines   |     100
(2 rows)
```

```
test01=#
```

```
test01=# SELECT count(*) FROM t_document_headers;
```

```
count
```

```
1000
(1 row)
```

```
test01=# SELECT count(*) FROM t_document_lines;
```

```
count
```

```
9950050
(1 row)
```

# Пример «медленный запрос» - план запроса

```
test01=# explain SELECT count(*) FROM
(
SELECT /* Long running query */ h.name, h.short_description, l.content, l.sort_field
FROM t_document_headers h,
     t_document_lines l
WHERE l.header_id=h.header_id AND
      h.security_level = 0 AND
      EXISTS (SELECT '1' FROM t_document_lines ll WHERE ll.header_id=h.header_id AND ll.security_level>0)
ORDER BY h.header_id, l.sort_field
) a;
```

## QUERY PLAN

```
-----
Aggregate  (cost=523570.53..523570.54 rows=1 width=8)
->  Sort   (cost=508164.55..510732.22 rows=1027065 width=104)
      Sort Key: h.header_id, l.sort_field
      => Nested Loop  (cost=1000.70..293275.62 rows=1027065 width=104)
            -> Nested Loop Semi Join  (cost=1000.27..259970.03 rows=1 width=8)
                  Join Filter: (h.header_id = ll.header_id)
                  -> Index Scan using t_document_headers_pkey on t_document_headers h  (cost=0.27..61.33 rows=175 width=4)
                        Filter: (security_level = 0)
                  -> Materialize  (cost=1000.00..259906.07 rows=1 width=4)
                        -> Gather  (cost=1000.00..259906.07 rows=1 width=4)
                              Workers Planned: 2
                              -> Parallel Seq Scan on t_document_lines ll  (cost=0.00..258905.97 rows=1 width=4)
                                    Filter: (security_level > 0)
            -> Index Scan using t_document_lines_header_id_idx on t_document_lines l  (cost=0.43..23034.94 rows=1027065 width=8)
                  Index Cond: (header_id = h.header_id)
```

(15 rows)

- 1 Популярные расширения показывают статистику выполнения запроса только после его завершения
- 2 Определить, что делает работающий запрос в текущий момент времени, средствами ванильного СУБД затруднительно
- 3 Средств трассировки процесса выполнения запроса на уровне СУБД нет
- 4 Имея установленный и настроенный набор из популярных расширений, можно собрать статистику выполнения SQL операторов для последующего анализа и диагностики проблем
- 5 Для интерпретации результатов трассировки уровня ОС требуются знания из смежных областей (системное администрирование, разработка)
- 6 В некоторых случаях трассировка backend процесса позволяет увидеть причины медленной работы запроса
- 7 Даже установка и настройка большого количества открытых инструментов не дает достаточно полной картины происходящего в СУБД

# Чего не хватает для оперативного анализа

- ⊗ Возможности online мониторинга запросов в процессе их выполнения (аналог Sql Monitor в Oracle)
- ⊗ Возможности полной трассировки проблемной сессий с минимальными накладными расходами и включением на лету (аналог трассировки 10046 в Oracle)
- ⊗ Единого полнофункционального модуля расширения для надежного мониторинга экземпляра PostgreSQL
- ⊗ Единого инструмента быстрого сбора диагностики по всем компонентам систем на основе PostgreSQL
- ⊗ Отсутствие быстрого и наглядного графического инструмента для интерпретации базовых статистик
- ⊗ Полноты исторических статистик



# Инструмент PGARM

## Диагностика и МОНИТОРИНГ PostgreSQL

Решение ФОРС

# PGARM - SQL Monitor

Средство мониторинга выполняющихся в СУБД запросов в режиме online

## Решаемые задачи

-  Хранение списка и статистик выполнения запросов, удовлетворяющих заданным критериям (например, «время выполнения» > 1 минуты)
-  Отображение плана выполнения запросов и статистик по отдельным узлам плана для уже завершенных запросов и для запросов в стадии выполнения

## Особенности

-  Собранные данные хранятся в памяти
-  Возможность настройки критериев для фильтрации запросов, попадающих в «зону внимания»

# Пример «медленный запрос» Sql Monitor

| sql_text                                           | plan_hash_value | exec_id | status    | elapsed_time |
|----------------------------------------------------|-----------------|---------|-----------|--------------|
| SELECT /* Long running query */ h.name, h.short_de | 3679691853      | 28      | done      | 38.00        |
| SELECT /* Long running query */ h.name, h.short_de | 3679691853      | 29      | done      | 38.00        |
| SELECT /* Long running query */ h.name, h.short_de | 3679691853      | 30      | done      | 36.00        |
| SELECT /* Long running query */ h.name, h.short_de | 3679691853      | 31      | done      | 38.00        |
| SELECT /* Long running query */ h.name, h.short_de | 3679691853      | 32      | done      | 39.00        |
| SELECT /* Long running query */ h.name, h.short_de | 141741635       | 38      | done      | 1.00         |
| SELECT /* Long running query */ h.name, h.short_de | 501967702       | 38      | executing | 84.00        |
| SELECT /* Long running query */ h.name, h.short_de | 141741635       | 38      | done      | 1.00         |

(8 rows)

- План запроса изменился
- Время выполнения запроса увеличилось
- Запрос с новым планом продолжает выполняться

# SQL Monitor

## План в реальном времени

Plan Hash Value: 501967702  
Parallel workers: 2  
Leader PID: 431556

| Operation                                       | Object name             | Line Id | Elapsed time | Timeline | Loops | Operation cost       | Est. rows | Act. rows | Buffer   | Buffer line | Temp     | Temp line | I/O bytes | I/O line |
|-------------------------------------------------|-------------------------|---------|--------------|----------|-------|----------------------|-----------|-----------|----------|-------------|----------|-----------|-----------|----------|
| Sort                                            |                         | 0       | 0ms          |          | 0     | 199001.78..200290.43 | 515K      |           |          |             | 0        |           | 0         |          |
| .. Sort Key: h.header_id,                       |                         |         |              |          |       |                      |           |           |          |             |          |           |           |          |
| .. Sort Method: still in                        |                         |         |              |          |       |                      |           |           |          |             |          |           |           |          |
| 0kB                                             |                         |         |              |          |       |                      |           |           |          |             |          |           |           |          |
| .. Nested Loop                                  |                         |         |              |          |       |                      |           |           |          |             |          |           |           |          |
| .... Nested Loop Semi Join                      |                         |         |              |          |       |                      |           |           |          |             |          |           |           |          |
| ..... Nested Loop                               |                         |         | 33s          |          | 0     | 1000.7..150096.22    | 515K      | 4M        | 505.20Mb |             | 976.77Mb |           | 0         |          |
| ..... Nested Loop Semi Join                     |                         | 4       | 26s          |          | 0     | 1000.27..130983.17   | 1         | 453       | 228.56Mb |             | 500.09Mb |           | 0         |          |
| ..... Join Filter: (h.header_id = l1.header_id) |                         | 5       |              |          |       |                      |           |           |          |             |          |           |           |          |
| ..... Index Scan                                | t_document_headers_pkey | 6       | 7ms          |          | 0     | 0.27..42.04          | 90        | 907       | 1.47Mb   |             | 16.00Kb  |           | 0         |          |
| ..... Filter: (h.security_level = 0)            |                         | 7       |              |          |       |                      |           |           |          |             |          |           |           |          |
| ..... Materialize                               |                         | 8       | 5ms          |          | 906   | 1000..130938.98      | 1         | 49K       | 227.09Mb |             | 500.00Mb |           | 0         |          |
| ..... Gather                                    |                         |         | 635ms        |          | 0     | 1000..130938.98      | 1         |           |          |             | 500.00Mb |           | 0         |          |
| ..... Workers Planned: 2                        |                         |         |              |          |       |                      |           |           |          |             |          |           |           |          |
| ..... Parallel Seq Scan                         | t_document_lines        | 11      | 622ms        |          | 0     | 0..129938.08         | 1         | 4K        | 77.84Mb  |             | 183.65Mb |           | 0         |          |
| ..... Filter: (l1.security_level > 0)           |                         | 12      |              |          |       |                      |           |           |          |             |          |           |           |          |

# Explain

## План после пересбора статистик

```
test01=# explain SELECT /* Long running query */ h.name, h.short_description, l.content, l.sort_field
FROM t_document_headers h,
     t_document_lines l
WHERE l.header_id=h.header_id AND
      h.security_level = 0 AND
      EXISTS (SELECT '1' FROM t_document_lines ll WHERE ll.header_id=h.header_id AND ll.security_level>0)
ORDER BY h.header_id, l.sort_field;
```

### QUERY PLAN

```
-----
Incremental Sort (cost=1460.16..1477253.96 rows=9024205 width=81)
  Sort Key: h.header_id, l.sort_field
  Presorted Key: h.header_id
  -> Merge Join (cost=100.31..423763.38 rows=9024205 width=81)
    Merge Cond: (h.header_id = l.header_id)
    -> Nested Loop Semi Join (cost=0.71..17888.53 rows=902 width=48)
      -> Index Scan using t_document_headers_pkey on t_document_headers h (cost=0.28..72.35 rows=907 width=44)
        Filter: (security_level = 0)
      -> Index Scan using t_document_lines_header_id_idx on t_document_lines ll (cost=0.43..513.48 rows=52 width=4)
        Index Cond: (header_id = h.header_id)
        Filter: (security_level > 0)
    -> Index Scan using t_document_lines_header_id_idx on t_document_lines l (cost=0.43..291254.25 rows=9949509 width=41)
(12 rows)
```

# Пример «ресурсоемкий запрос» ASH

```
select db_name, username, application, client_addr, exec_id, queryid sql_id, plan_hash_value,  
round(max(rss_memory)/1024/1024,2) memory_used_mb from pg_active_session_history where  
sample_time > current_timestamp - interval '60 minutes' group by dbname , username ,  
application, client_addr, exec_id, queryid , plan_hash_value order by 8 desc ;
```

| db_name<br>text | username<br>text | application<br>text     | client_addr<br>text | exec_id<br>integer | sql_id<br>bigint   | plan_hash_value<br>bigint | memory_used_mb<br>numeric |
|-----------------|------------------|-------------------------|---------------------|--------------------|--------------------|---------------------------|---------------------------|
| test01          | postgres         | pgAdmin 4 - CONN:968711 | XXX.XXX             | 1                  | 950642715280185996 | 3380900887                | 6888.00                   |

Получаем информацию о проблемной сессии:

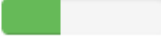
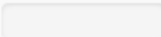
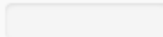
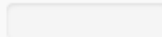
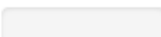
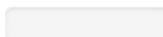
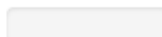
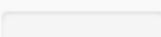
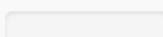
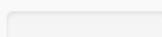
- Откуда была запущена сессия
- Идентификатор выполнения запроса `exec_id`
- Сколько памяти было использовано

# Пример «ресурсоемкий запрос» SQL Monitor

Plan Hash Value: 3380900887

Parallel workers: 0

Leader PID: 514772

| Operation                                     | Object name | Line Id | Elapsed time | Timeline                                                                            | Loops | Operation cost               | Est. rows | Act. rows | Buffer  | Buffer line                                                                         | Temp   | Temp line                                                                           |
|-----------------------------------------------|-------------|---------|--------------|-------------------------------------------------------------------------------------|-------|------------------------------|-----------|-----------|---------|-------------------------------------------------------------------------------------|--------|-------------------------------------------------------------------------------------|
| Aggregate                                     |             | 0       | 3m 37s       |  | 0     | 16429369.38..<br>16429369.39 | 1         | 1         | 32.00Kb |  | 4.90Gb |  |
| .. Sort                                       |             | 1       | 57s          |  | 0     | 14929369.38..<br>15179369.38 | 100M      | 100M      | 24.00Kb |  | 4.90Gb |  |
| .... Sort Key: o.id                           |             | 2       |              |  |       |                              |           |           |         |  |        |  |
| .... Sort Method: quicksort Memory: 7053218kB |             | 3       |              |  |       |                              |           |           |         |  |        |  |
| .... Seq Scan                                 | otdel       | 4       | 0ms          |  | 0     | 0..1641657                   | 100M      | 0         | 0       |  | 0      |  |

# Демонстрация - трассировка сессии



# Трассировка сессии средствами СУБД



```
log_statement_stats / log_parser_stats,log_planner_stats,log_executor_stats= on;  
log_min_messages = DEBUG5; log_error_verbosity = verbose;
```

## Текст SQL операторов

```
2024-03-15 17:26:34.788 MSK [35897] STATEMENT:  INSERT INTO t_document_lines (line_id, header_id, content, sort_field, last_updated)  
VALUES (nextval('t_document_line_id'), 1000, 'Описание эксперимента: ...', 0, NOW());
```

## Служебные («рекурсивные») вызовы ядра СУБД (например, при работе с FK)

```
2024-03-15 17:26:34.789 MSK [35897] CONTEXT:  
SQL statement "SELECT 1 FROM ONLY "public"."t_document_headers" x WHERE "header_id" OPERATOR(pg_catalog.=) $1 FOR KEY SHARE OF x"
```



# Трассировка сессии средствами СУБД

Фазы работы оптимизатора и затраченные ресурсы

[log\\_parser\\_stats](#)

- PARSER STATISTICS, PARSE ANALYSIS STATISTICS, REWRITER STATISTICS

[log\\_planner\\_stats](#)

- PLANNER STATISTICS

[log\\_executor\\_stats](#)

- EXECUTOR STATISTICS

[log\\_statement\\_stats](#)

- QUERY STATISTICS

[log\\_min\\_messages + log\\_error\\_verbosity](#)

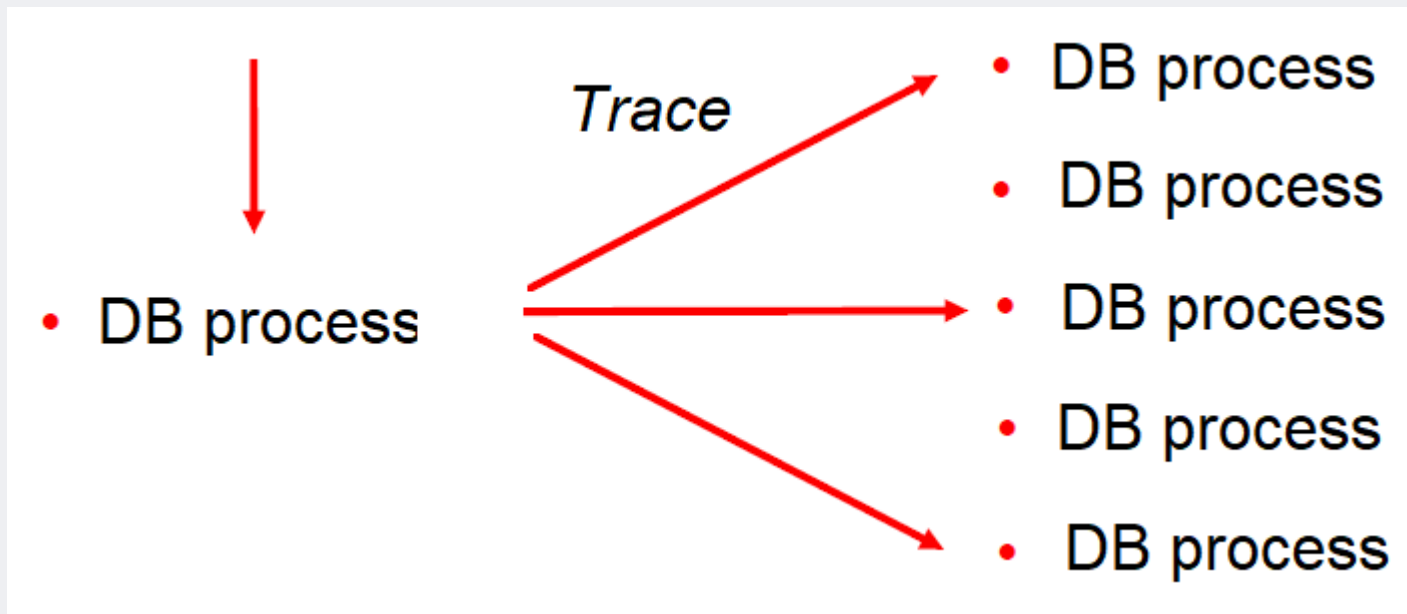
- Вызовы функций ядра СУБД

```
DETAIL:  ! system usage stats:
         !      0.000060 s user, 0.000014 s system, 0.000073 s elapsed
         !      [61.094308 s user, 14.148411 s system total]
         !      153400 kB max resident size
         !      0/0 [29368/5449320] filesystem blocks in/out
         !      0/0 [0/39584] page faults/reclaims, 0 [0] swaps
         !      0 [0] signals rcvd, 0/0 [0/0] messages rcvd/sent
         !      0/0 [6739/5967] voluntary/involuntary context switches
```

# Чего не хватает в стандартной трассировке

- ⊗ Возможность включения/выключения для любой сессии в процессе ее работы
- ⊗ История ожиданий, возникающих при выполнении каждого из запросов трассируемой сессии с их атрибутами
- ⊗ Нет иерархии рекурсивных запросов
- ⊗ Логирование запросов, выполняемых внутри функций/процедур

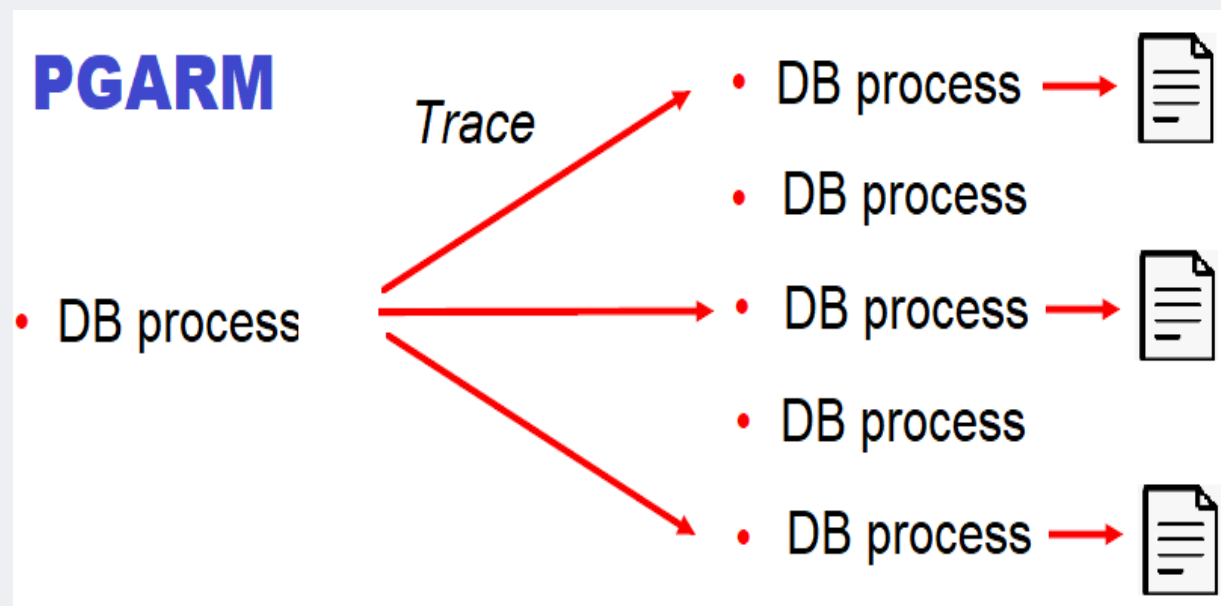
# Демонстрация (трассировка сессии расширением PGARM)



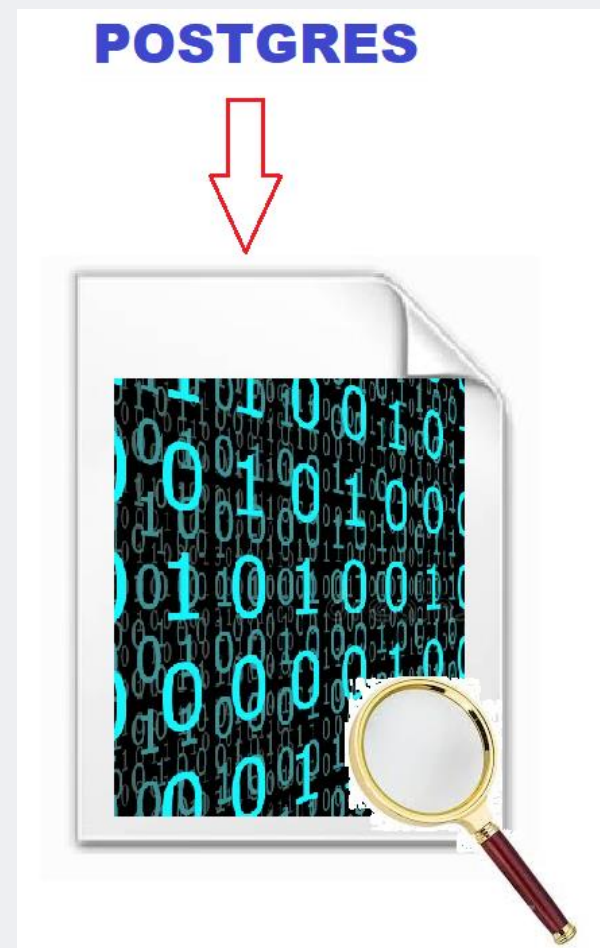
Пользователь может инициировать запуск трассировки backend-процесса либо своей, либо другой сессии

- SET pgarm\_sql\_trace = <N>; - своя сессия
- SELECT pgarm\_trace (<PID>,<N>); - соседний процесс

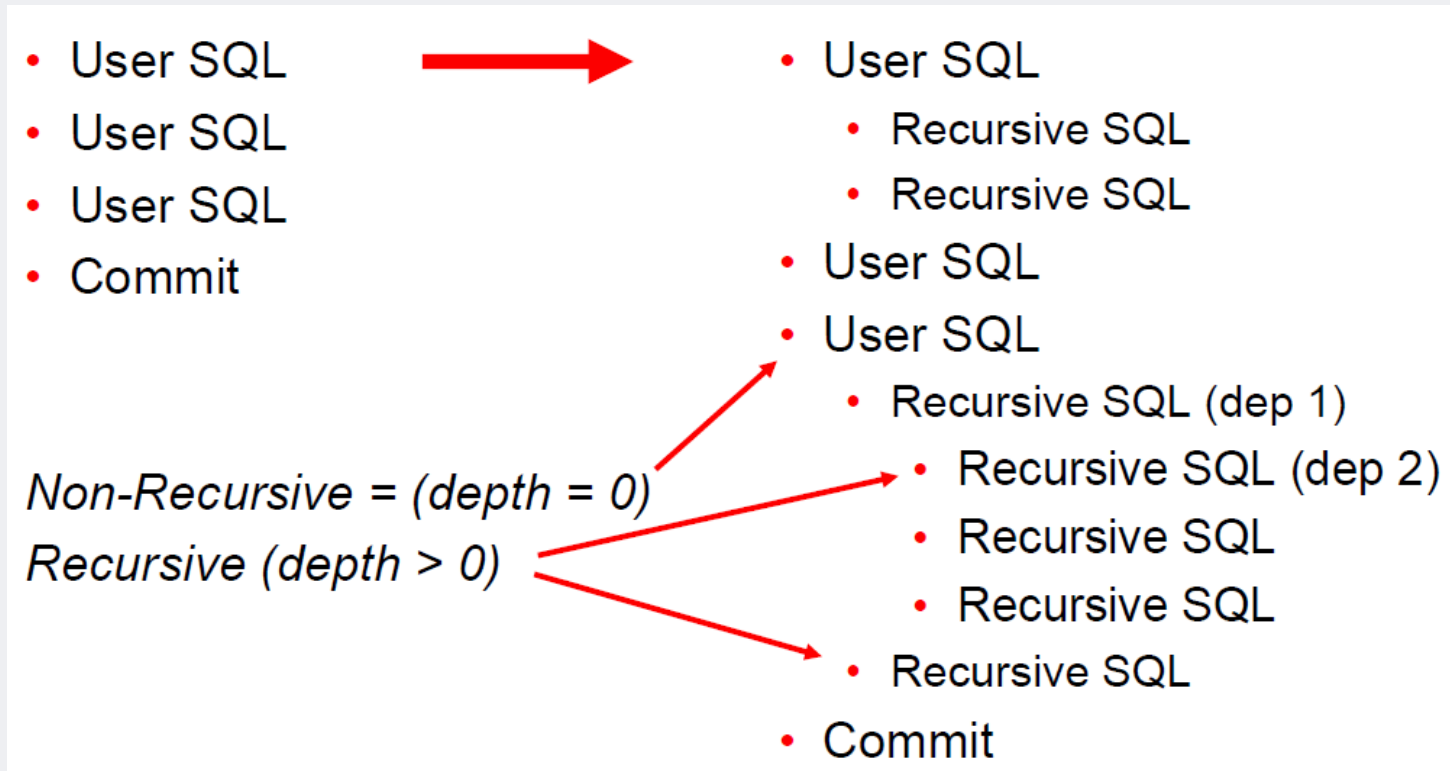
# Демонстрация (трассировка сессии расширением PGARM)



Для каждого трассируемого процесса создается свой лог файл

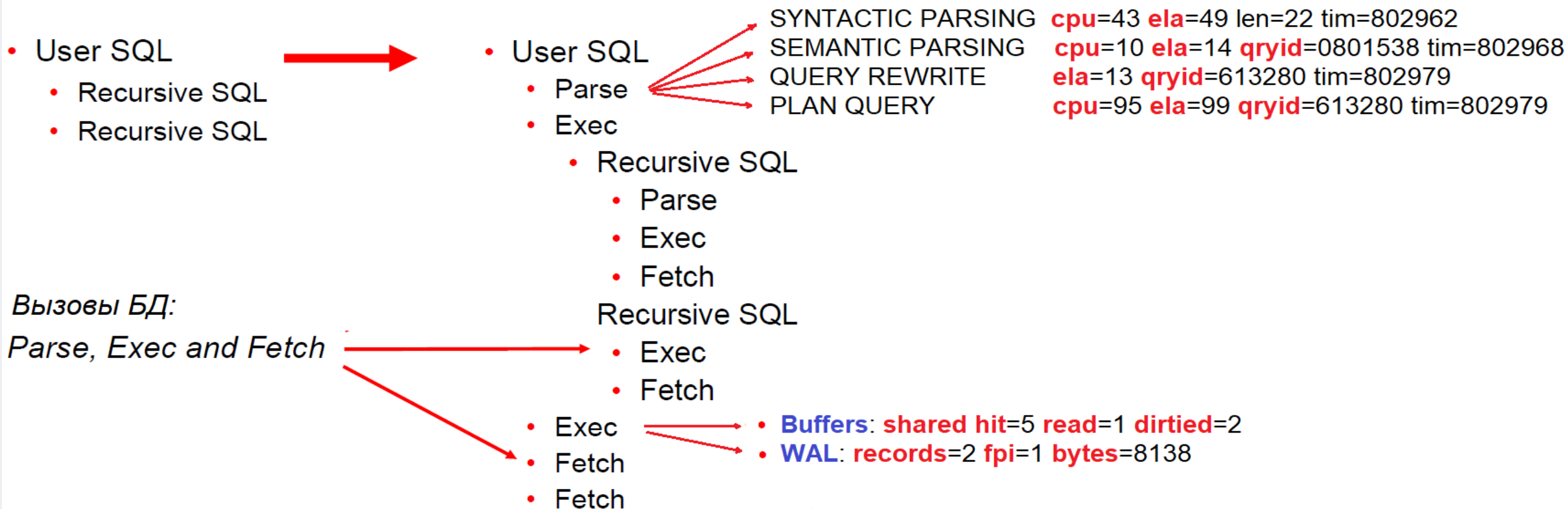


# Демонстрация (трассировка сессии расширением PGARM)



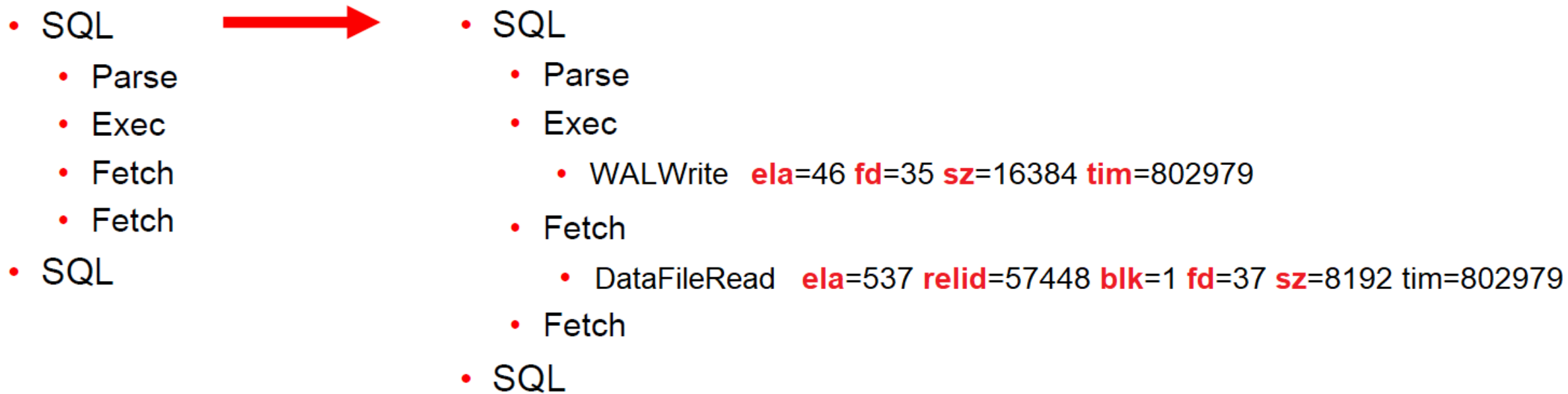
- В трассировку попадают выполнение всех рекурсивных запросов с их уровнем вложенности
- Фиксации или откат транзакций

# Демонстрация (трассировка сессии расширением PGARM)



Трассировка включает в себя все стадии выполнения запросов с основными атрибутами

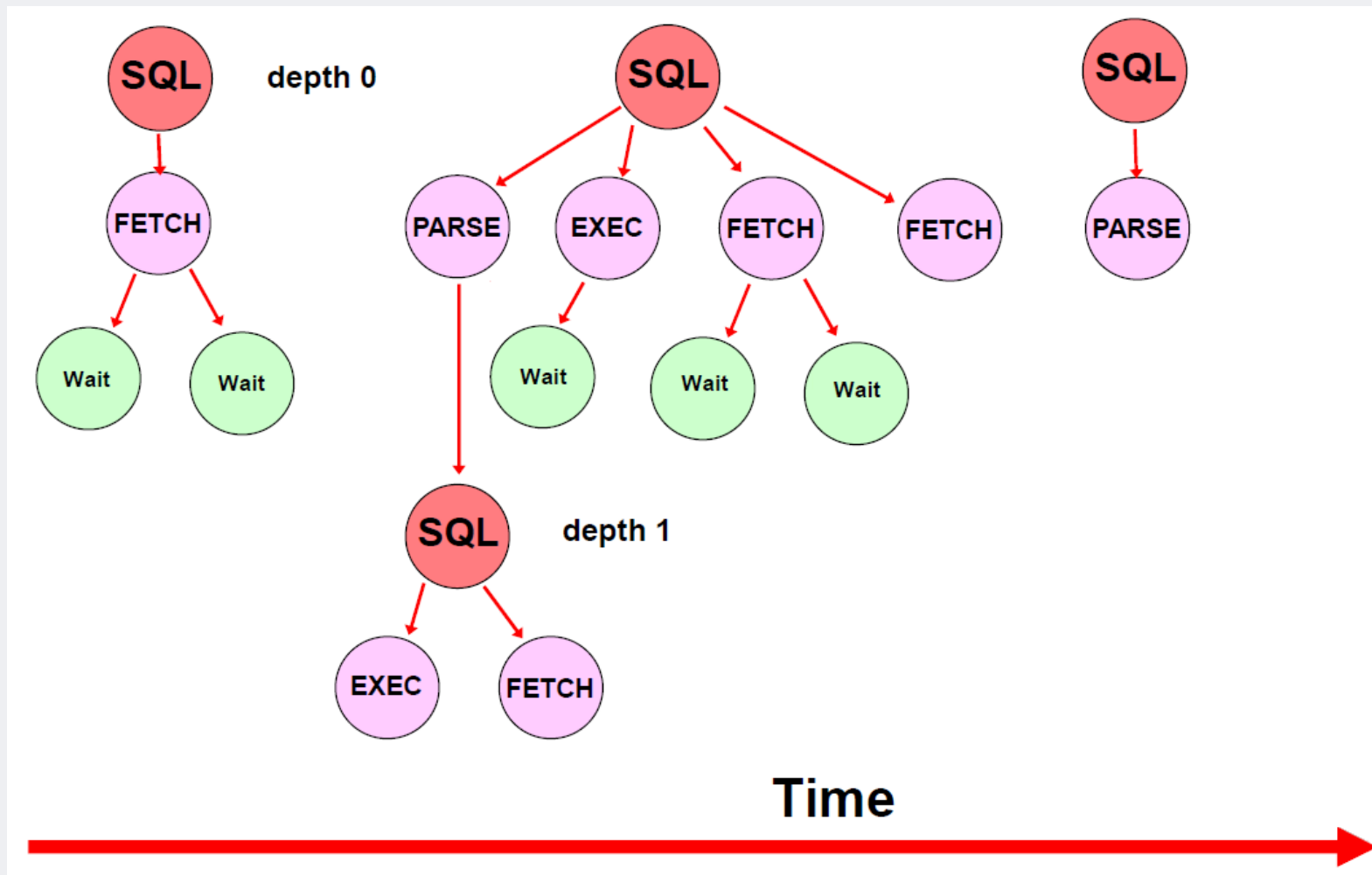
# Демонстрация (трассировка сессии расширением PGARM)



**В трассировку попадают все ожидания с их основными атрибутами:**

- Затраченное время
- Номер файлового дескриптора
- Номер объекта БД
- Номер блока
- Размер

# Демонстрация (трассировка сессии расширением PGARM)



# Демонстрация (трассировка сессии расширением PGARM)

```
SEMANTIC PARSING #3389489787819415 cpu=10 ela=14 qryid=10801538042594098731 tim=8029688648189
QUERY REWRITE #3389489787819415 ela=0 qryid=10801538042594098731 tim=8029688648201
SYNTACTIC PARSING #3708664253432936 cpu=105 ela=111 len=191 tim=8029792355595
INSERT INTO t_document_lines (line_id, header_id, content, sort_field, last_updated)
VALUES (nextval('t_document_line_id'), 1000, 'Описание эксперимента: ...', 99, NOW());
END OF STMT
SEMANTIC PARSING #3708664253432936 cpu=159 ela=164 qryid=6132808734000177446 tim=8029792356395
QUERY REWRITE #3708664253432936 ela=13 qryid=6132808734000177446 tim=8029792356425
PLAN QUERY #3708664253432936 cpu=95 ela=99 qryid=6132808734000177446 tim=8029792356535
```

Query Text: SELECT 1 FROM ONLY "public"."t\_document\_headers" x WHERE "header\_id" OPERATOR(pg\_catalog.=) \$1 FOR KEY SHARE OF x

LockRows (cost=0.28..8.30 rows=1 width=10) (actual time=0.057..0.058 rows=1 loops=1)

Buffers: shared hit=5

WAL: records=1 bytes=54

-> Index Scan using t\_document\_headers\_pkey on t\_document\_headers x (cost=0.28..8.29 rows=1 width=10) (actual time=0.037..0.038 rows=1 loops=1)

Index Cond: (header\_id = 1000)

Buffers: shared hit=3

PLAN QUERY #4383609713628882 cpu=1691 ela=3143 qryid=16631607933342280911 tim=8032018409777

EXEC #4383609713628882

WAIT #4383609713628882: nam='DataFileRead' ela=839 relid=57441 blk=0 fd=18 sz=8192 tim=8032018411459

WAIT #4383609713628882: nam='DataFileRead' ela=283 relid=57441 blk=1 fd=18 sz=8192 tim=8032018411874

WAIT #4383609713628882: nam='DataFileRead' ela=58 relid=57441 blk=2 fd=18 sz=8192 tim=8032018412030

WAIT #4383609713628882: nam='DataFileRead' ela=32 relid=57441 blk=3 fd=18 sz=8192 tim=8032018412157

WAIT #4383609713628882: nam='DataFileRead' ela=892 relid=57441 blk=4 fd=18 sz=8192 tim=8032018413134

WAIT #4383609713628882: nam='DataFileRead' ela=32 relid=57441 blk=5 fd=18 sz=8192 tim=8032018413275

# Пример «рекурсивный запрос» SQL trace

## CURSOR #6 dep=0

```
TRACE ON #5 level=1 tim=10267129192394
PARSING IN CURSOR #6 len=389 dep=0 tim=10267136008262
INSERT INTO document_headers(header_id, name, short_description, author, description, sec
VALUES(10001, 'Программа PG Conf', 'Программа выступлений на PG Conf 2024', 'Организаторы',
END OF STMT
PARSED #6 c=355 e=361 qryid=1597700373026691200 dep=0 tim=10267136008624
PLAN #6 c=85 e=88 qryid=1597700373026691200 dep=0 tim=10267136008758
PLAN #6 c=389 e=405 qryid=873468798256325797 dep=1 tim=10267136009398
WAIT #6: nam='DataFileRead' ela=49 relid=207505 blk=0 fd=6 sz=8192 tim=10267136010485
WAIT #6: nam='DataFileRead' ela=32 relid=207505 blk=1 fd=6 sz=8192 tim=10267136010613
WAIT #6: nam='DataFileRead' ela=30 relid=207505 blk=2 fd=6 sz=8192 tim=10267136010695
```

## CURSOR #6 dep=1

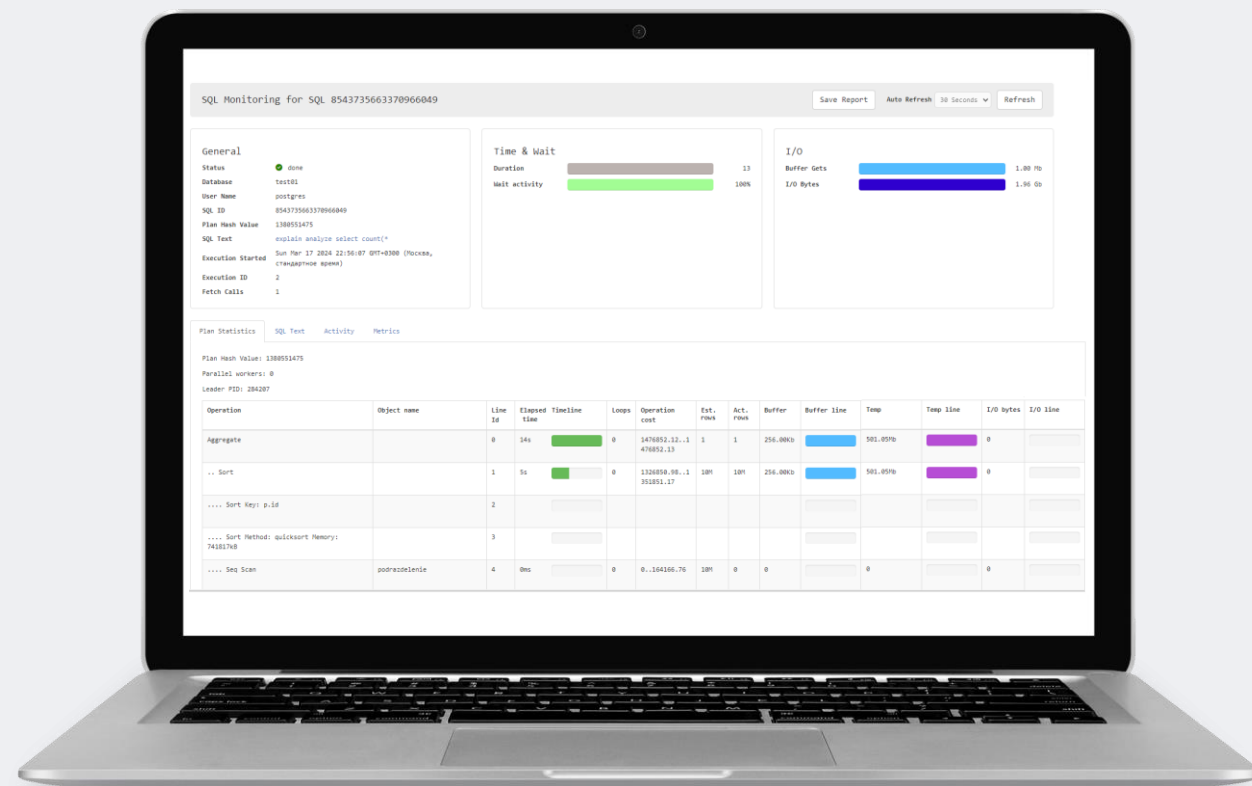
```
PARSING IN CURSOR #6 len=22 dep=1 tim=10267142648509
Query Text: SELECT :SCE(max(access_level), 0) FROM t_access_level WHERE (NEW.created BETWEEN
END OF STMT
PARSED #6 c=50 e=55 qryid=10801538042594098731 dep=0 tim=10267142648565
WAIT #6: nam='DataFileRead' dep=1 ela=9 relid=207505 blk=35 fd=6 sz=8192 tim=10267136013461
WAIT #6: nam='DataFileRead' dep=1 ela=9 relid=207505 blk=36 fd=6 sz=8192 tim=10267136013538
```

# Трассировка расширением PGARM

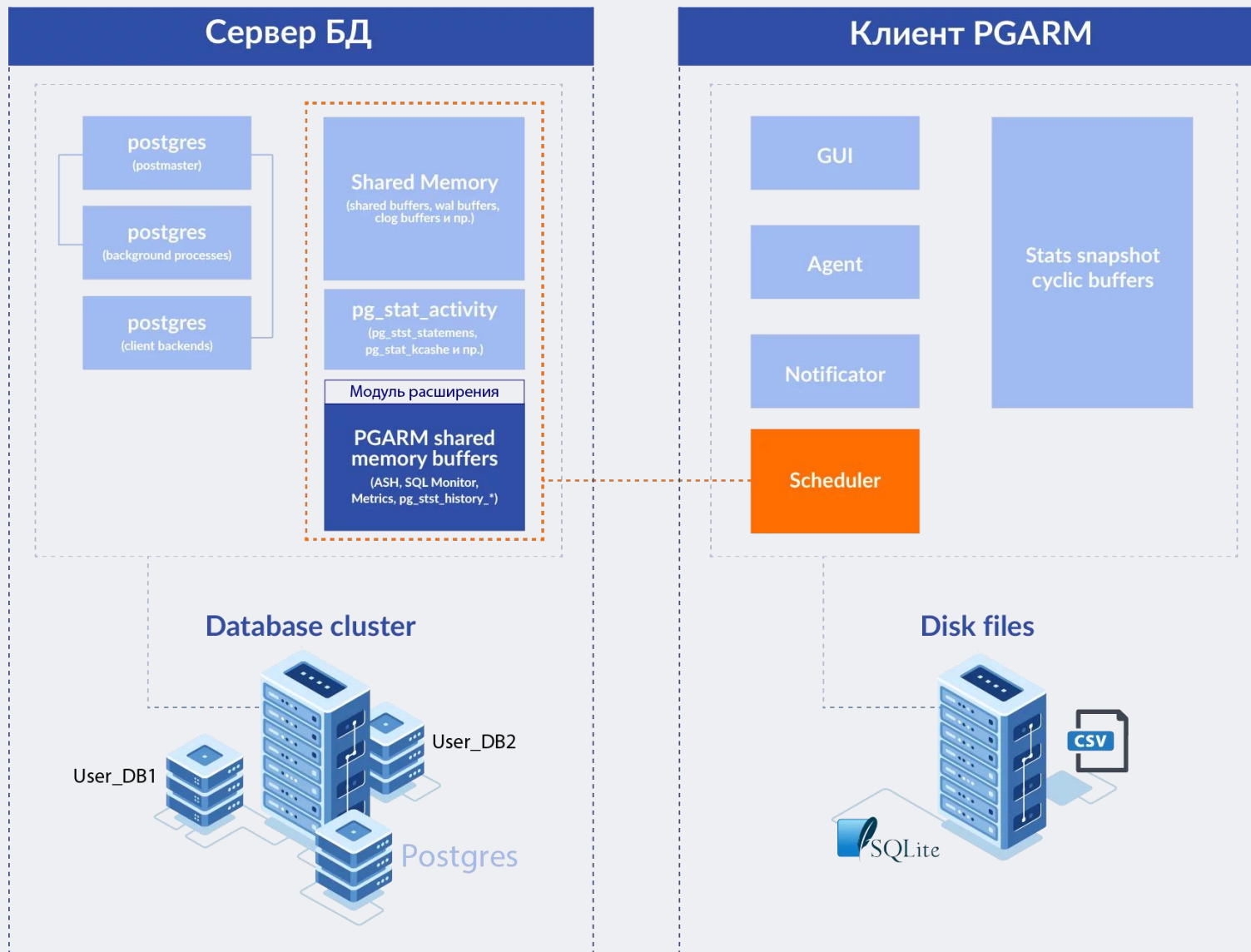
- ☒ Полная история SQL операторов сессии, выполненных за период трассировки – включая операторы из вызванных в сессии процедур и функций и служебные («рекурсивные») вызовы СУБД
- ☒ Возможность включить/отключить трассировку для любой сессии в любой момент времени
- ☒ Отдельный файл для каждой трассируемой сессии (нет спама в журнал БД)
- ☒ История ожиданий для каждой операции
- ☒ Планы запросов со статистикой выполнения

# Возможности PGARM

- SQL монитор
- Трассировка сессии
- История активных сессий.  
Аналог ASH в Oracle
- Таблица дерева блокировок
- Расширенные статистики, Метрики
- Сборщик снимков статистик работы  
целевых экземпляров
- Графические панели
- Агент заданий, процесс оповещений
- Кнопка для сбора пакета полной  
диагностики



# Архитектура PGARM



# Модуль расширения

## Текущая реализация модуля расширения:

Реализован механизм динамического менеджера разделяемой памяти

Также собираются основные статистики ОС. Данные считываются из псевдо файловой системы /proc

Для реализации исторического буфера истории активных сессий используется выделенный фоновый процесс экземпляра

Механизм трассировок позволяет реализовать любую трассировку, на текущий момент реализован аналог трассировки Oracle 10046

Для реализации исторических буферов всех остальных статистик и метрик используется другой фоновый процесс

Загрузка разделяемой библиотеки при запуске экземпляра

## Текущая реализация клиента («Light-версия»):

Совмещает функции сборщика снимков, агента, инструмента оповещений и графического клиента мониторинга

Предусматривает установку как на наблюдаемый сервер СУБД, так и наблюдение со сторонней клиентской машины администратора

Использует процессную архитектуру:

- Процесс scheduler выполняет сбор снимков статистик с наблюдаемых экземпляров, экспорт и упаковку статистик, позволяет отправить собранную диагностику в репозиторий, либо подготовить пакет для отправки в техническую поддержку
- Процесс GUI формирует графические панели по требованию
- Агент выполняет команды
- Процесс мониторинга, анализа и оповещений

Используется общее адресное пространство памяти для процесса сбора снимков статистик и процесса отрисовки графических панелей

# Сборщик снимков

## Реализация процесса scheduler:

Выполняет сбор снимков статистик с наблюдаемого экземпляра, позволяет отрисовывать их прямо из буфера памяти, сохраняет снимки в выбранное хранилище

Использует циклические буфера памяти для безостановочного сбора снимков статистик экземпляра

Умеет работать с любой редакцией PostgreSQL, больше статистик собирает с установленным модулем расширения

Позволяет собирать снимки статистик с нескольких локальных или удаленных экземпляров PostgreSQL

Утилита управления командной строки имеет богатый набор ключей, и текстовый конфигурационный файл для тонких настроек

# Графические панели PGARM



Портативны, информативны и интерактивны



Легкие, не требуют дополнительной установки и настройки



Отображают информацию экземпляра с минимальными задержками прямо из памяти процесса сбора статистик PGARM



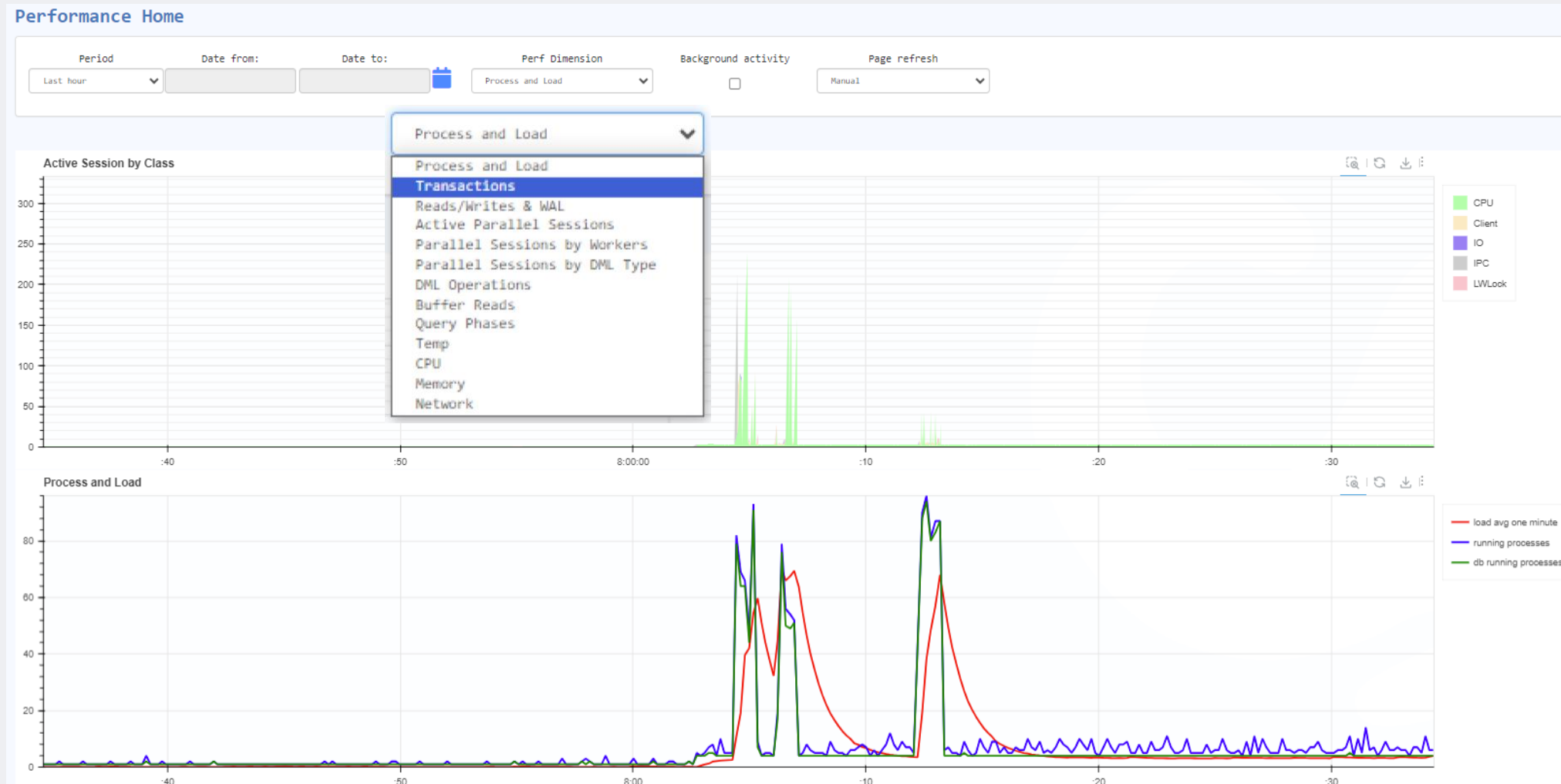
Отображают большинство статистик производительности экземпляра



Позволяют увидеть тренд развития события

# Графические панели PGARM

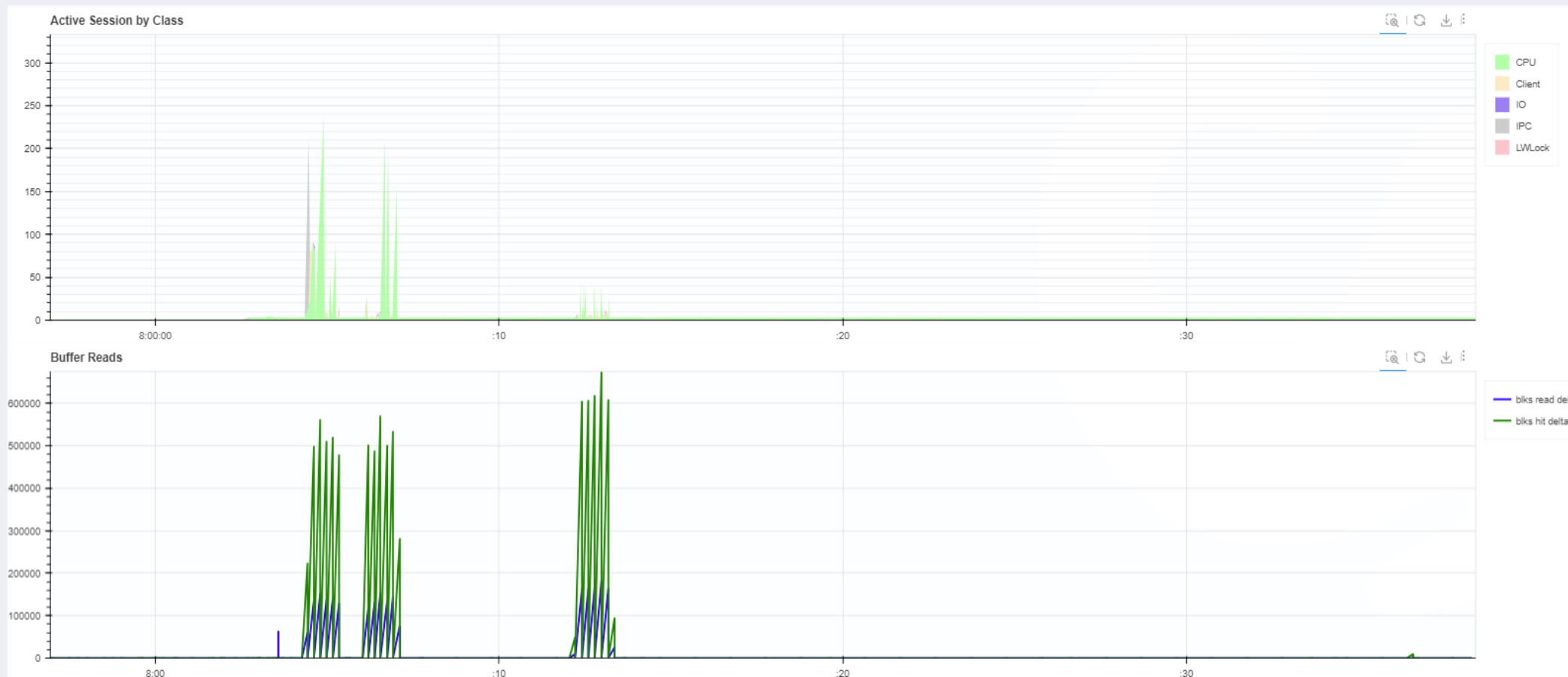
## Панель производительности БД



# Графические панели PGARM

## Панель производительности БД

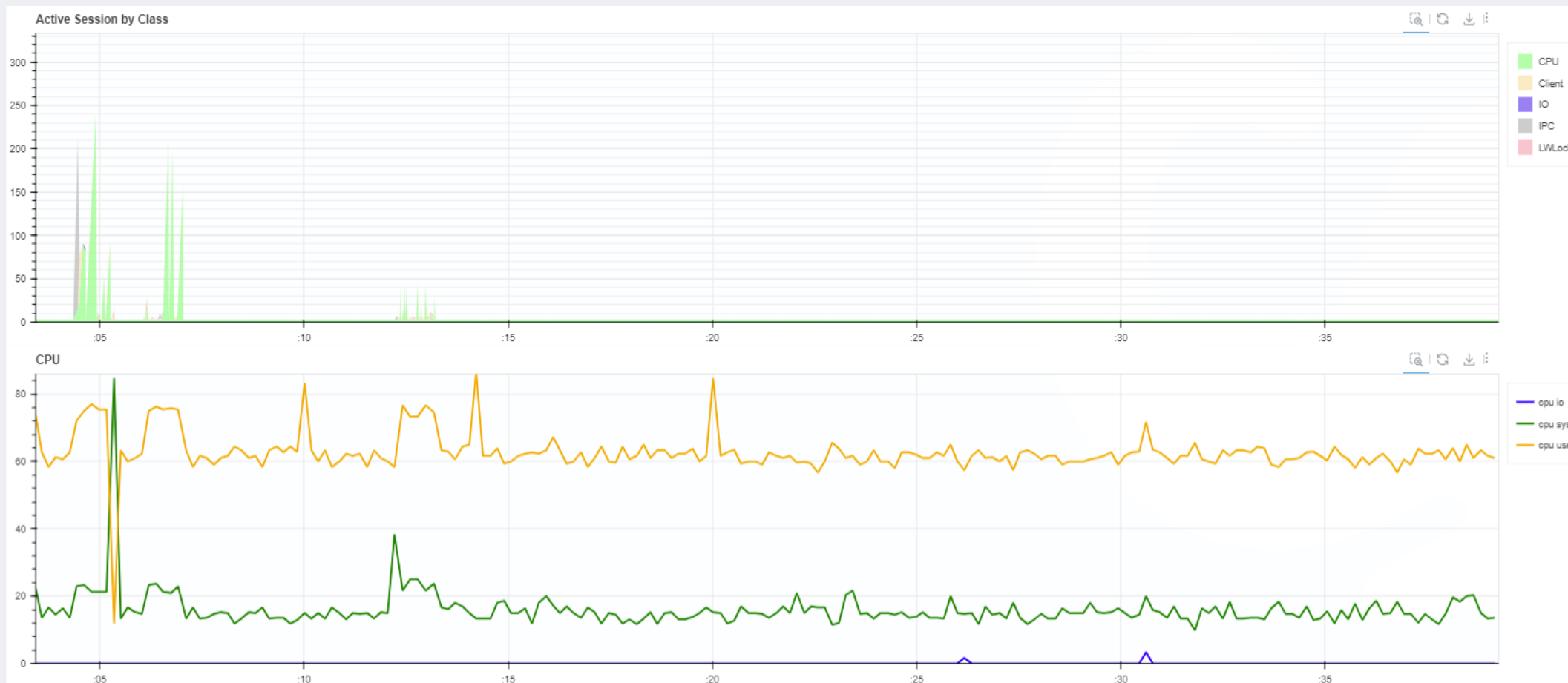
### Buffer Reads



# Графические панели PGARM

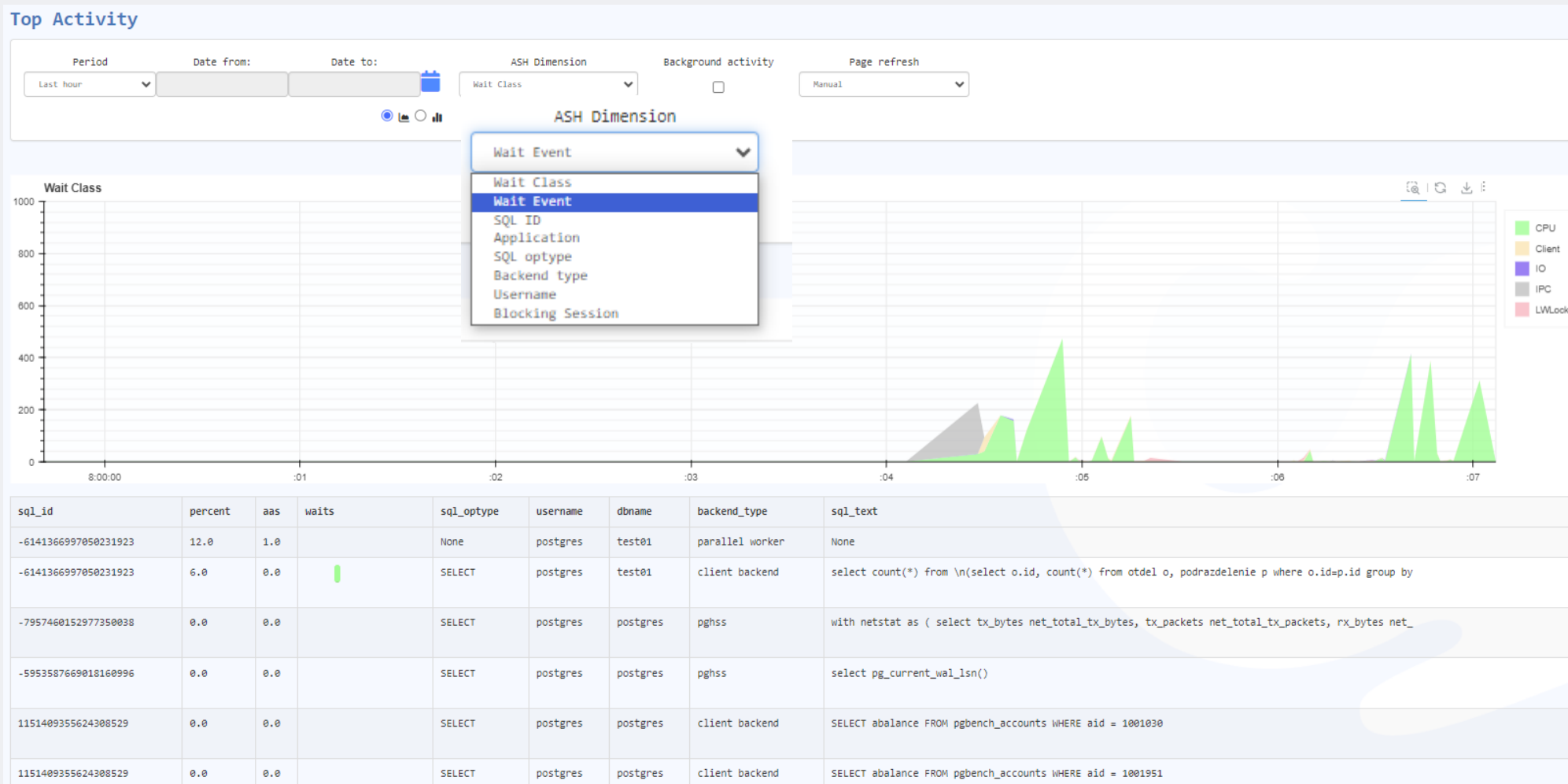
## Панель производительности БД

### CPU



# Графические панели PGARM

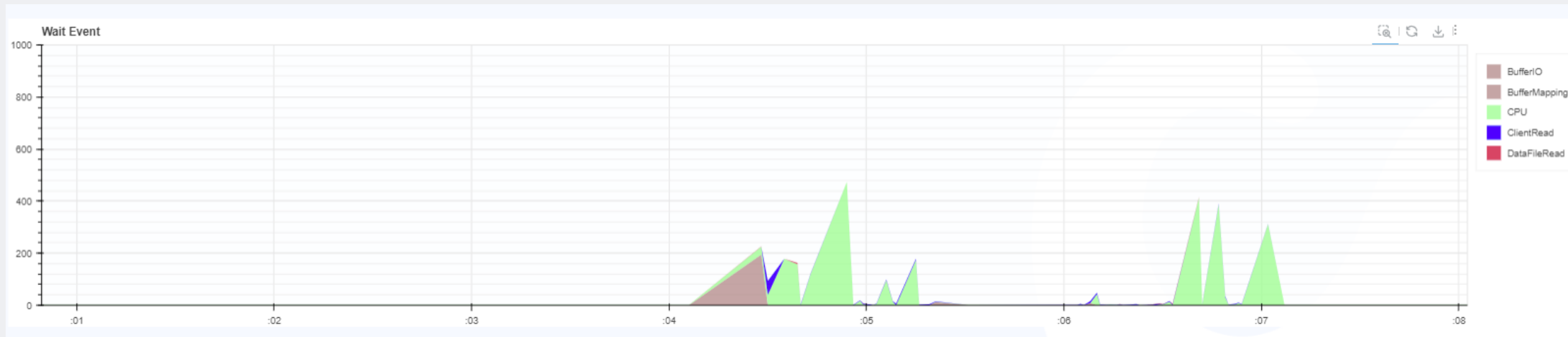
## Профиль нагрузки БД



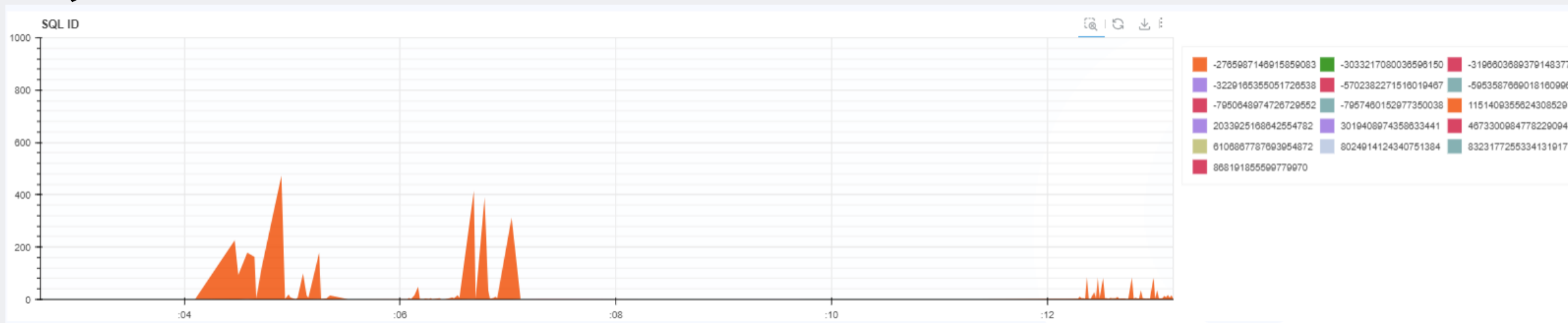
# Графические панели PGARM

## Профиль нагрузки БД

### Wait Event

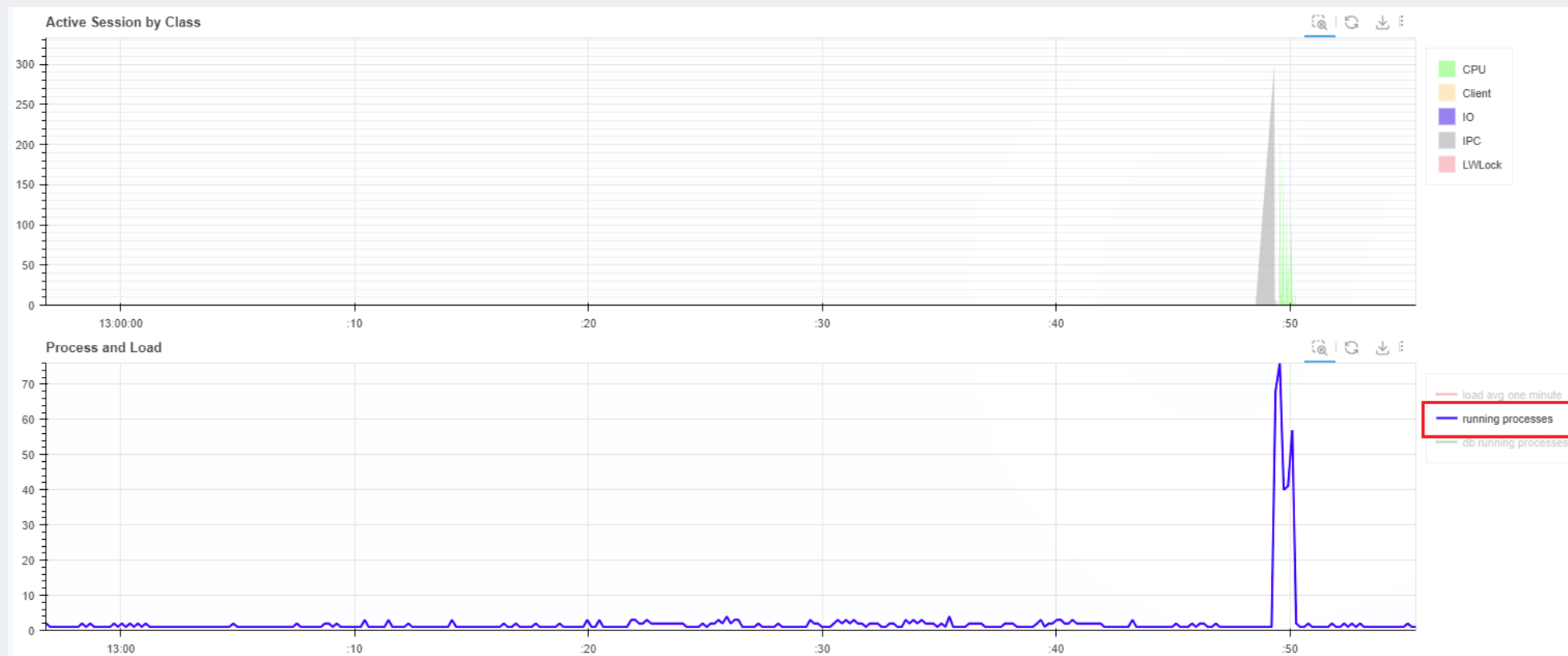


### SQL ID



# Графические панели PGARM

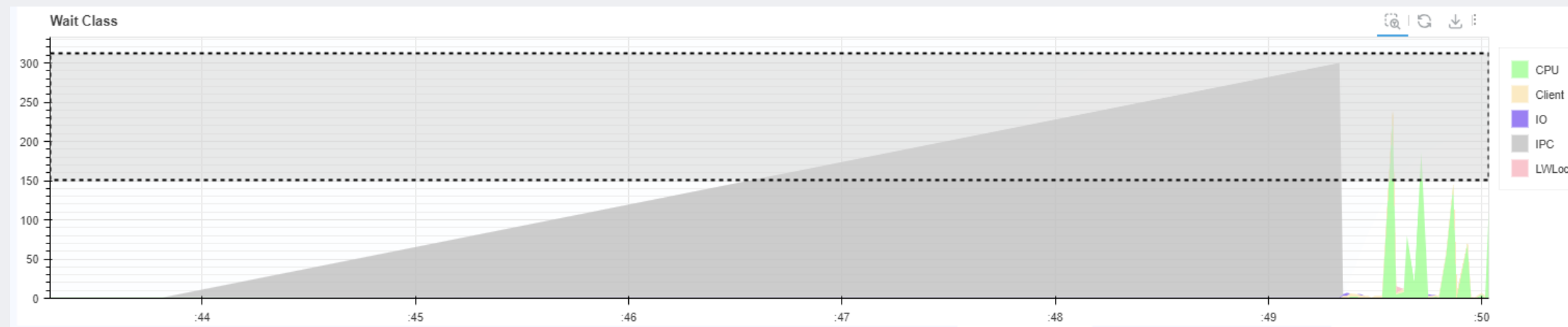
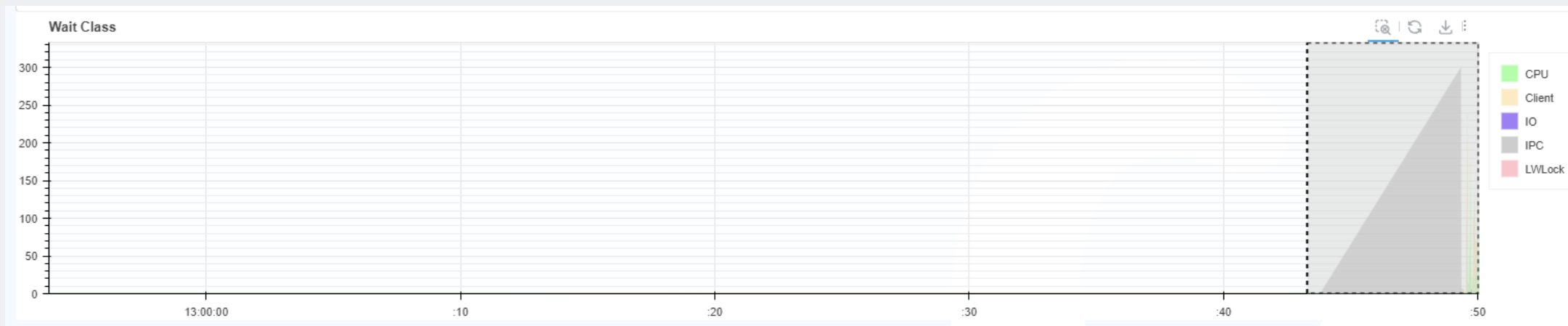
## Панель производительности БД



На панели можно включать и отключать отдельные графики в интерактиве

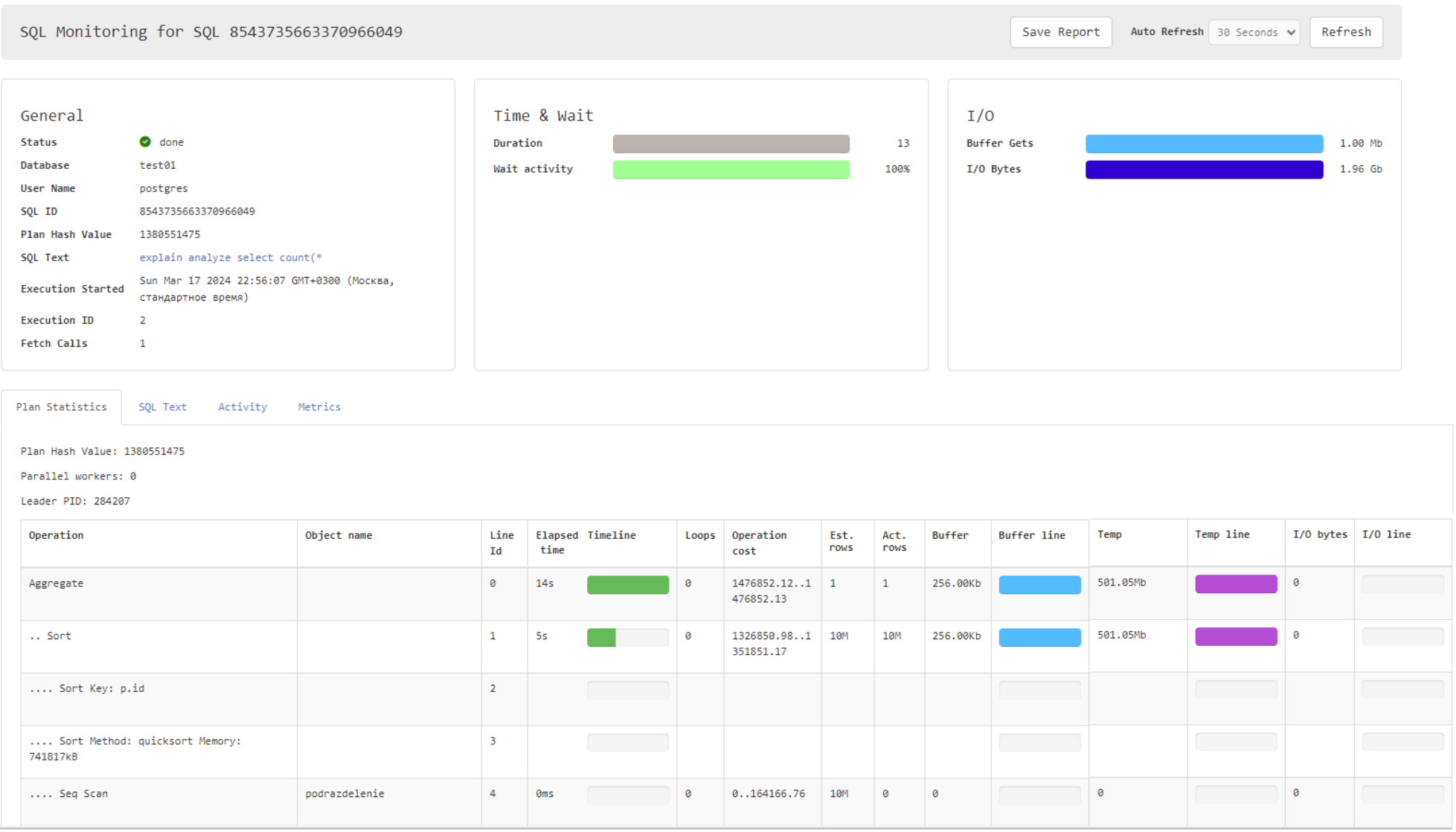
# Графические панели PGARM

## Профиль нагрузки БД



Графики можно зуммировать по оси X и Y

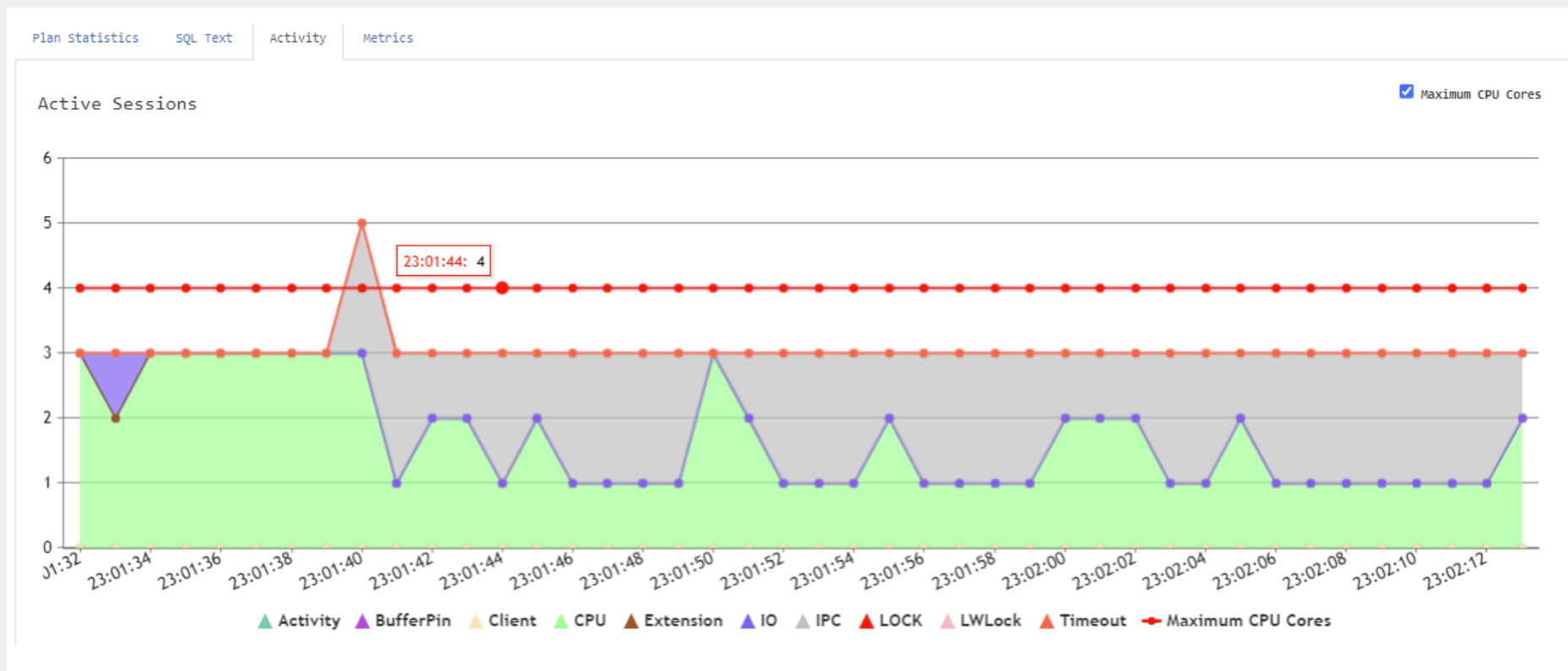
# Графические панели PGARM SQL Monitor



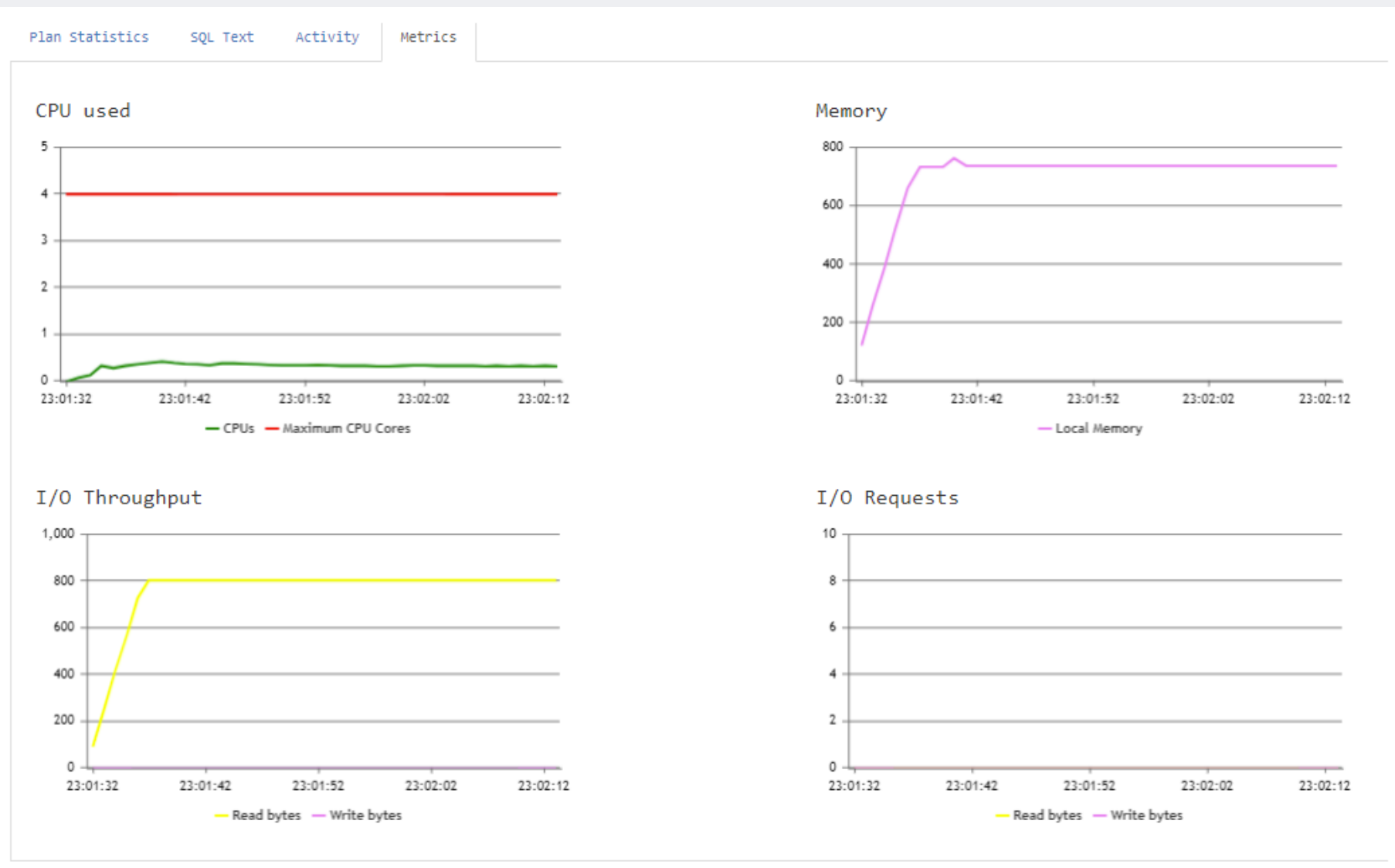
# Графические панели PGARM SQL Monitor

Plan Statistics SQL Text Activity Metrics

```
explain analyze select count(*) from (select * from podrazdelenie p order by p.id) dd;
```



# Графические панели PGARM SQL Monitor



# Преимущества PGARM



Мониторинг запросов в процессе их выполнения



Обширные базовые статистики PostgreSQL



Полная трассировка проблемной сессий с минимальными накладными расходами и включением на лету



Быстрый и наглядный графический инструмент для интерпретации текущих статистик



Единый инструмент быстрого сбора диагностики по всем компонентам систем на основе PostgreSQL



Единый полнофункциональный модуль расширения для надежного мониторинга экземпляра PostgreSQL



Свой алгоритм LRU пула статистик. Эффективнее в 20 раз, чем родной LRU ядра Postgres



Возможность инструмента PGARM взаимодействовать друг с другом на разных узлах



Легко устанавливается и масштабируется

# Спасибо за внимание!



## Контакты



129 272, Москва,  
Трифоновский тупик, 3



+7 (495) 747-7040



pgarm@fors.ru  
support@fors.ru



www.pgarm.ru

