

PGCONF.RUSSIA 2024

pgpro_gp – приоритизация
ресурсов

Александр Попов

a.popov@postgrespro.ru



План доклада

Проблемы высоконагруженных систем

Основные возможности pgpro_gp

Как использовать

Демо

Результаты тестов

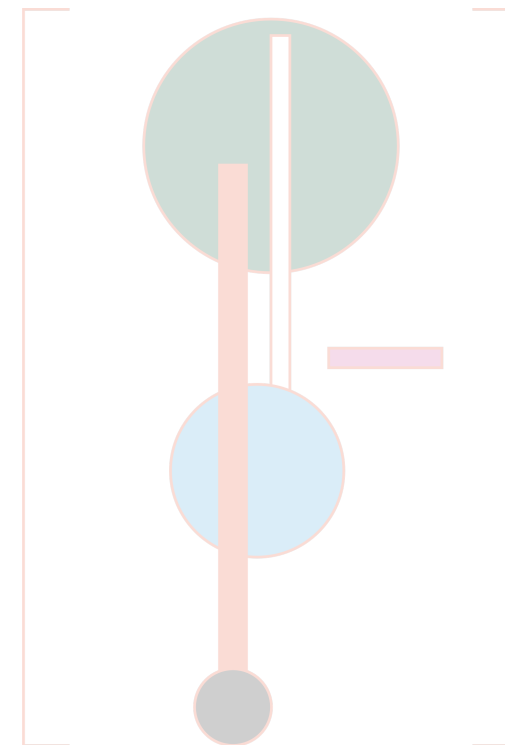


Проблемы высоконагруженных систем

Мало ресурсов

Кривые запросы

Одновременно идут транзакционные запросы и аналитические запросы (oltp vs olap)



Как решать

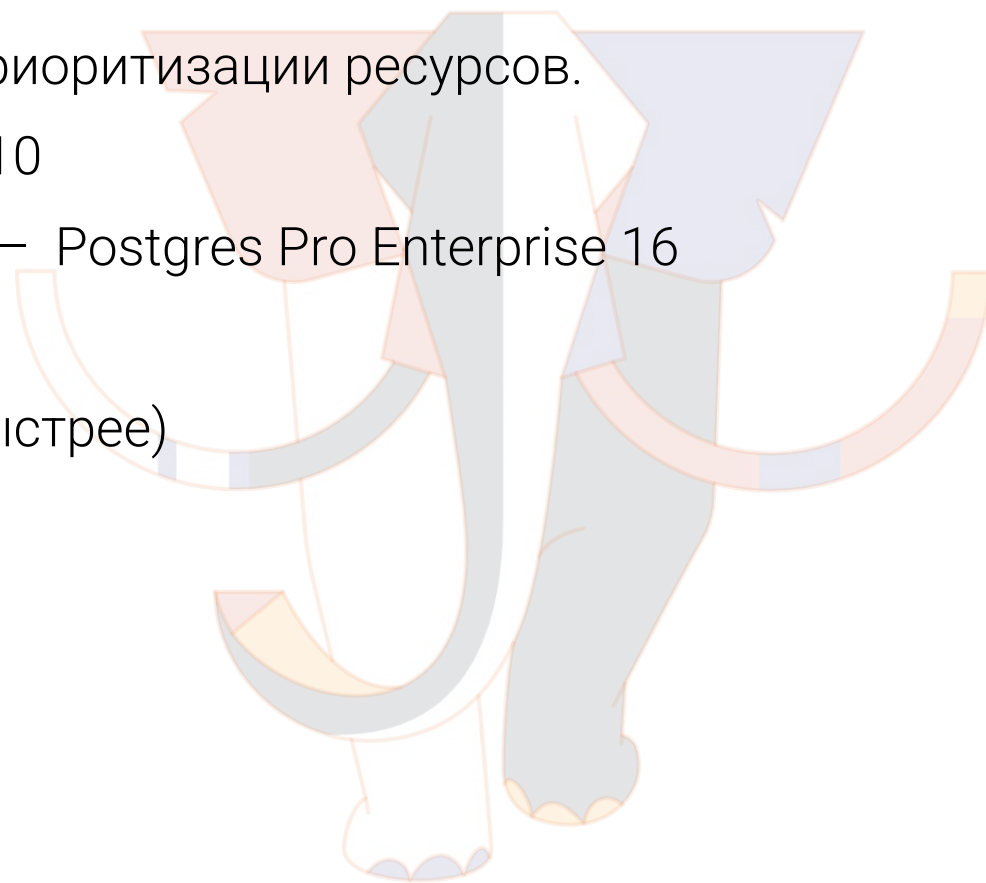
- Добавление железа (не всегда возможно)
- Переписывание запросов (часто дорого, запрос выполняется раз в месяц)
- Добавление реплик
- Приоритизация важных запросов!



Описание

pgpro_gp — это расширение Postgres Pro Enterprise для приоритизации ресурсов.

- Экспериментальная версия — Postgres Pro Enterprise 10
- Версия, доступная для промышленной эксплуатации — Postgres Pro Enterprise 16
- Работа на реплике
- Варианты приоритетов — 1, 2, 4, 8 (чем больше, тем быстрее)
- По умолчанию приоритет 4
- Участвуют только backend_type=client backend



Виды приоритетов

Приоритет выставляется по использованию:

- процессора - `session_cpu_weight`
- объём читаемых данных - `session_ioread_weigh`
- объём записываемых данных - `session_iowrite_weight`



Возможности – уровни приоритетов

Приоритет выставляется на уровне:

- Сессии
- Пользователя (группы)
- БД

Выбирает план с максимальной суммой приоритетов среди планов, назначенных ролям, членом которых является пользователь.



Возможности – смена приоритета работающей сессии

- `pg_backend_set_config`
- `pgpro_rp_backend_set_plan`



Возможности – работа с планами приоритизации ресурсов

- Создание плана с заданным именем и параметрами:
pgpro_rp_create_plan(
 a_plan_name text,
 a_plan_options jsonb
) returns bigint

Назначение плана роли:

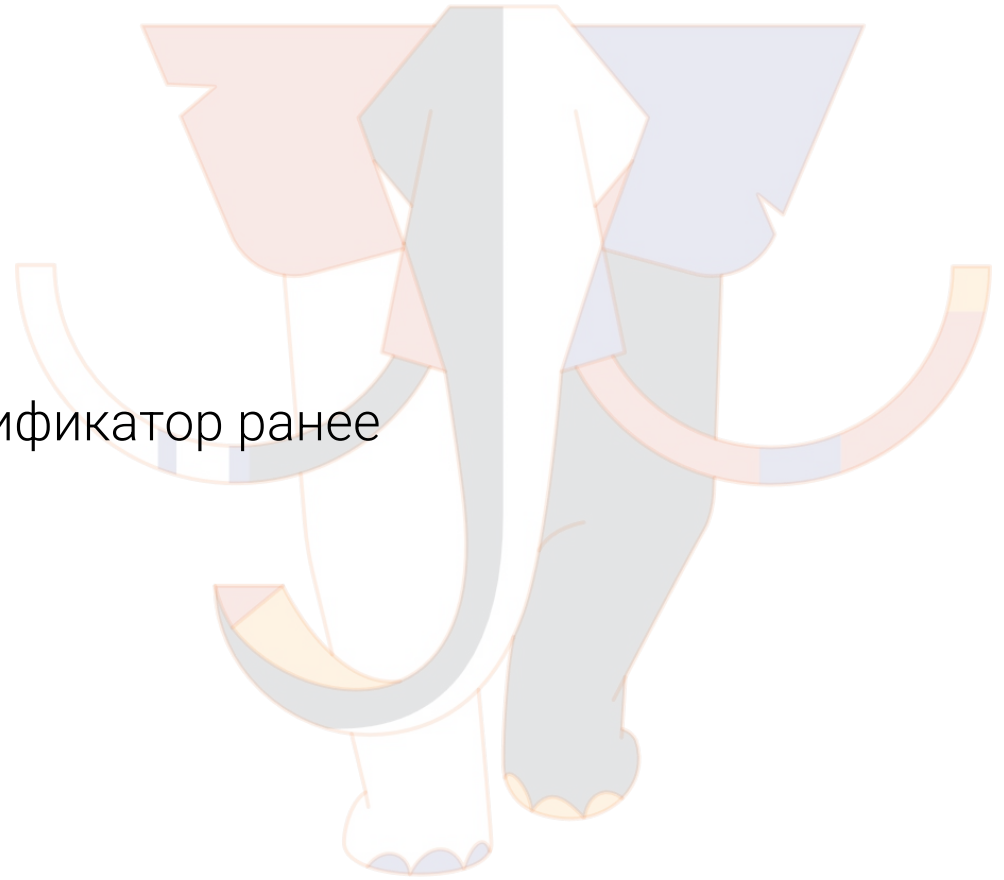
- pgpro_rp_create_role_plan(
 a_role_name text,
 a_plan_name text
) returns void



Возможности – выбор плана при входе пользователя в систему

```
pgpro_rp_set_plan_selection_function(  
    a_func_name text  
) returns void #
```

a_func_name – имя функции, которая возвращает идентификатор ранее созданного плана



Выставление через сессионные переменные

```
postgres=# create extension pgpro_rp;
```

```
CREATE EXTENSION
```

```
postgres=# alter system set  
usage_tracking_interval = 1;
```

```
ALTER SYSTEM
```

```
postgres=# select pg_reload_conf();
```

```
pg_reload_conf
```

```
-----
```

```
t
```

```
(1 row)
```

```
postgres=# set session_cpu_weight =8;
```

```
SET
```

```
postgres=# set session_ioread_weight =8;
```

```
SET
```

```
postgres=# set session_iowrite_weight =8;
```

```
SET
```



Работа с планами приоритетов – создание плана

```
postgres=# SELECT pgpro_rp_create_plan(  
    a_plan_name => 'plan_max'  
    , a_plan_options => '{'  
        ' "session_cpu_weight": 8'  
        , "session_ioread_weight": 8'  
        , "session_iowrite_weight": 8 '  
    }'  
    );  
pgpro_rp_create_plan
```



Работа с планами приоритетов – назначение плана пользователю

```
postgres=# SELECT pgpro_rp_create_role_plan(  
    a_role_name => 'r_oltp'  
    , a_plan_name => 'plan_max'  
    );  
pgpro_rp_create_role_plan
```

(1 row)



Назначение плана работающему процессу

— сессия 1

```
postgres=# select pg_backend_pid();
```

```
pg_backend_pid
```

```
-----
```

```
80947
```

```
(1 row)
```

```
postgres=# select count(*) from
```

```
pgbench_accounts;
```

```
count
```

```
-----
```

```
1000000
```

— Сессия 2

```
postgres=# select pgpro_rp_backend_set_plan(
```

```
    a_pid => 80947
```

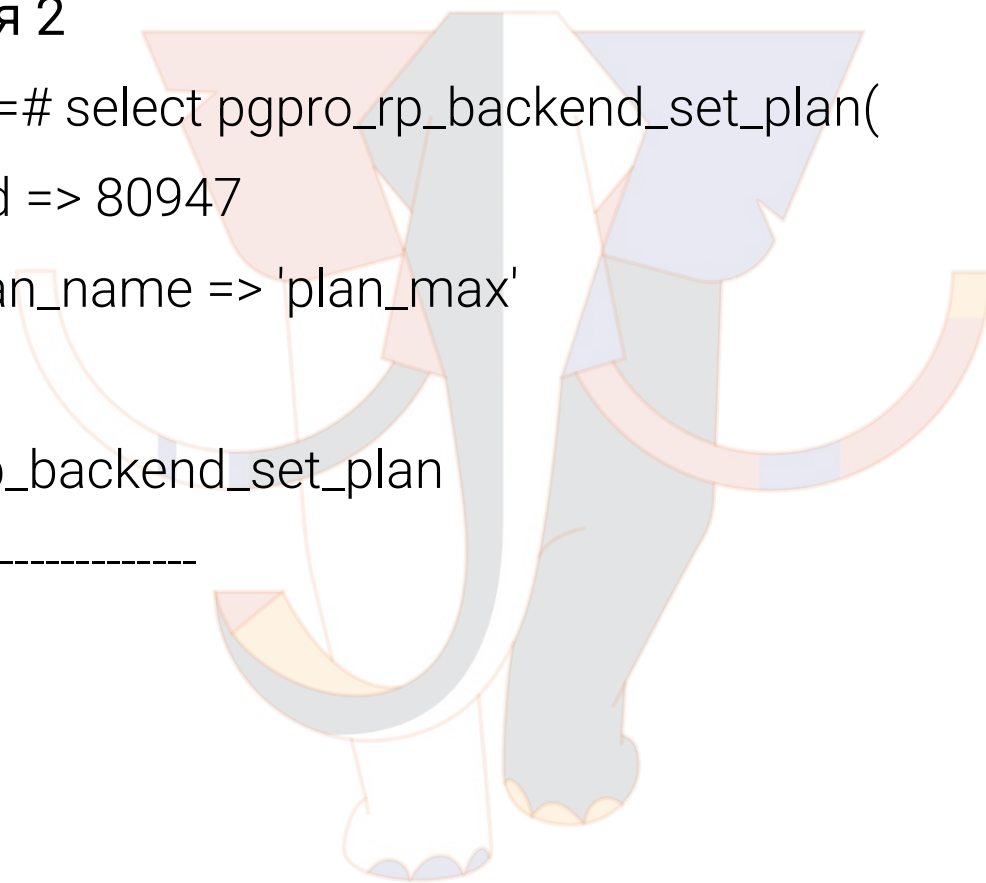
```
    , a_plan_name => 'plan_max'
```

```
);
```

```
pgpro_rp_backend_set_plan
```

```
-----
```

```
(1 row)
```



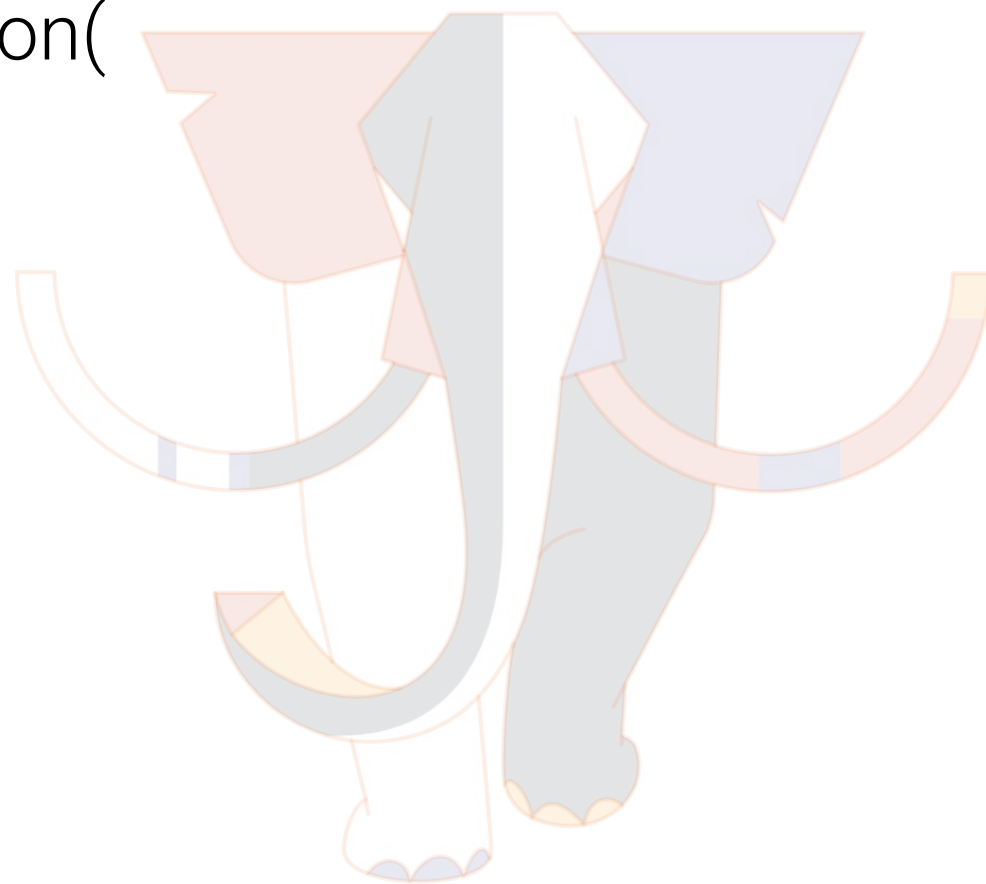
Функция выбора плана при входе – создание логики

```
create or replace function get_plan(  
    ) returns bigint as  
    $body$  
begin -- логика  
    If current_user = 'postgres' then  
        return 1;  
    else  
        return 2;  
    end if;  
end;  
$body$ language plpgsql;
```



Функция выбора плана при входе

```
select pgpro_rp_set_plan_selection_function(  
    a_func_name := 'get_plan'  
);
```



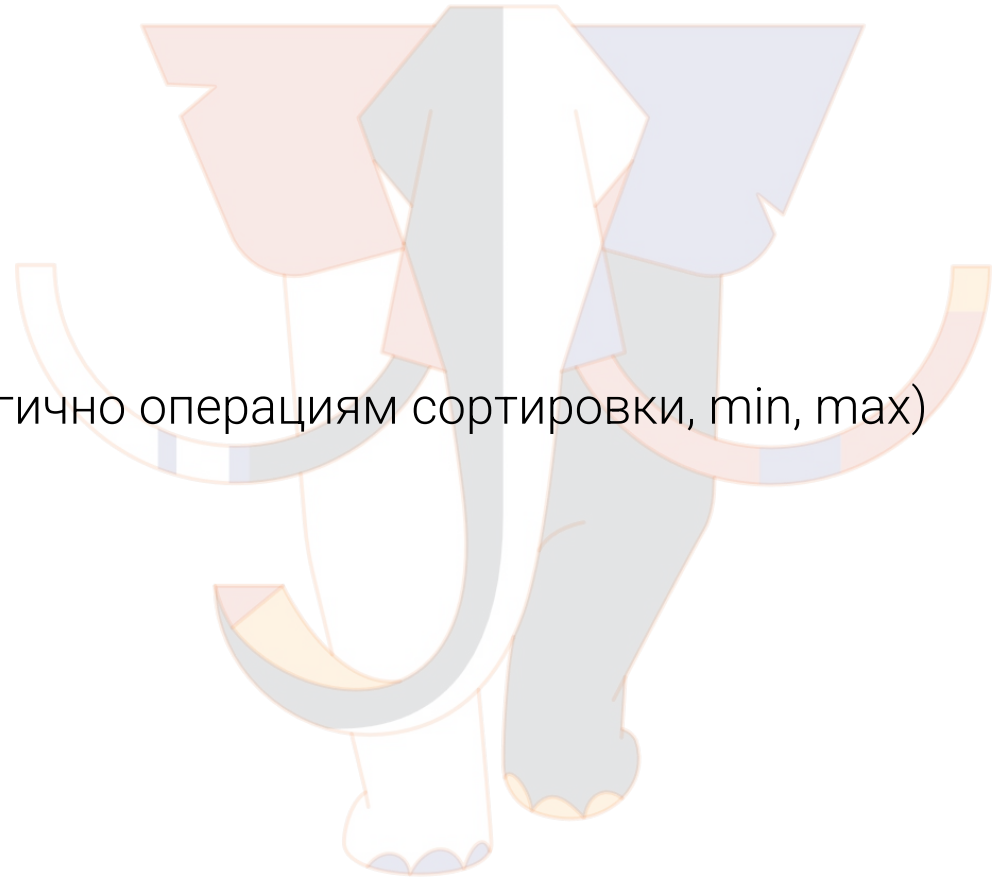
Функция выбора плана при входе

```
postgres=# \c
postgres=# select name, setting
  from pg_settings
 where name like 'session_%_weight';
   name      | setting
-----+-----
session_cpu_weight    | 8
session_ioread_weight | 8
session_iowrite_weight | 8
(3 rows)
```



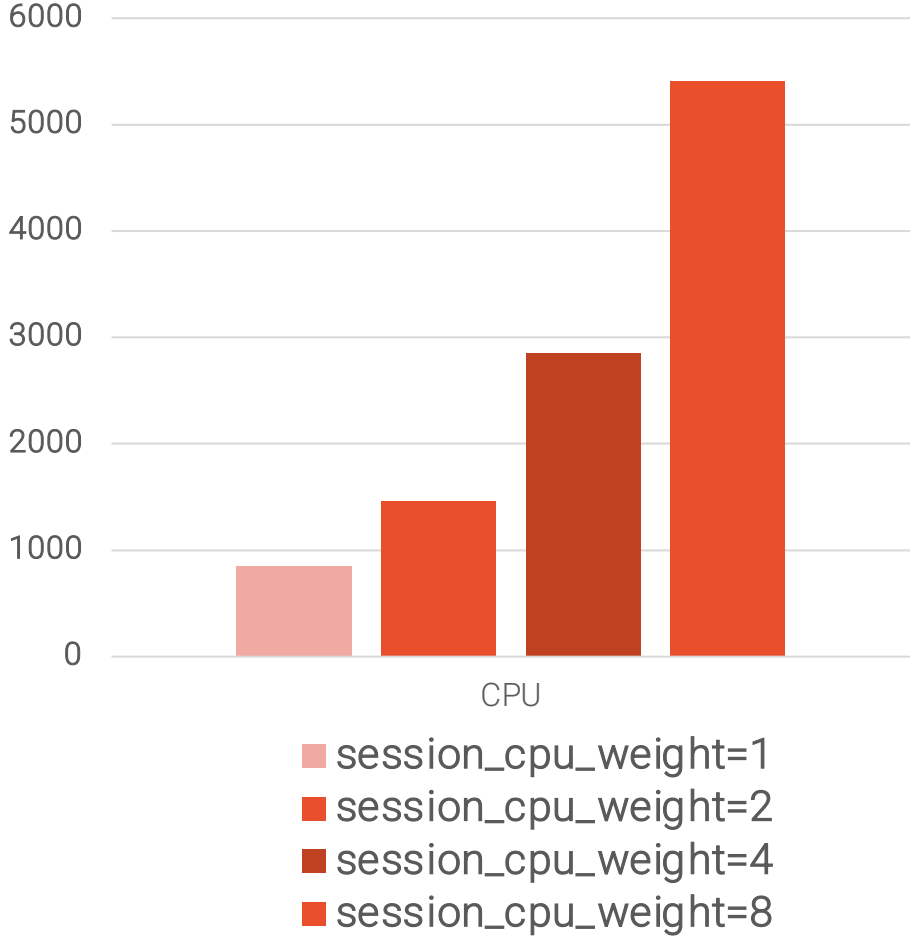
Тесты

- Ubuntu 22.04
- 4 ядра
- 16GB оперативной памяти.
- session_cpu_weight
 - вычисления числа "пи" методом Монте-Карло (аналогично операциям сортировки, min, max)
- session_iowrite_weight -
 - UNLOGGED таблица
 - 10.000 записей = 2МБ
- session_iowrite_weight
 - UNLOGGED таблица
 - 10.000 записей = 2МБ

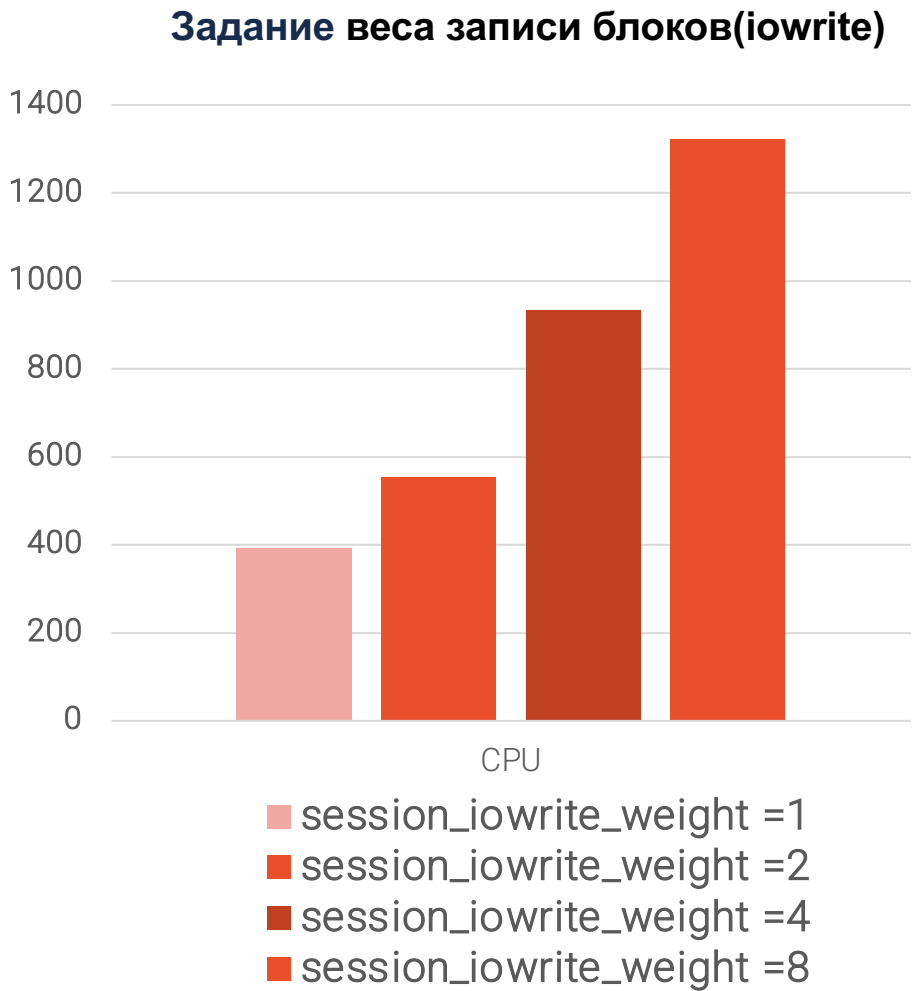




Задание веса использования процессора (cpu)

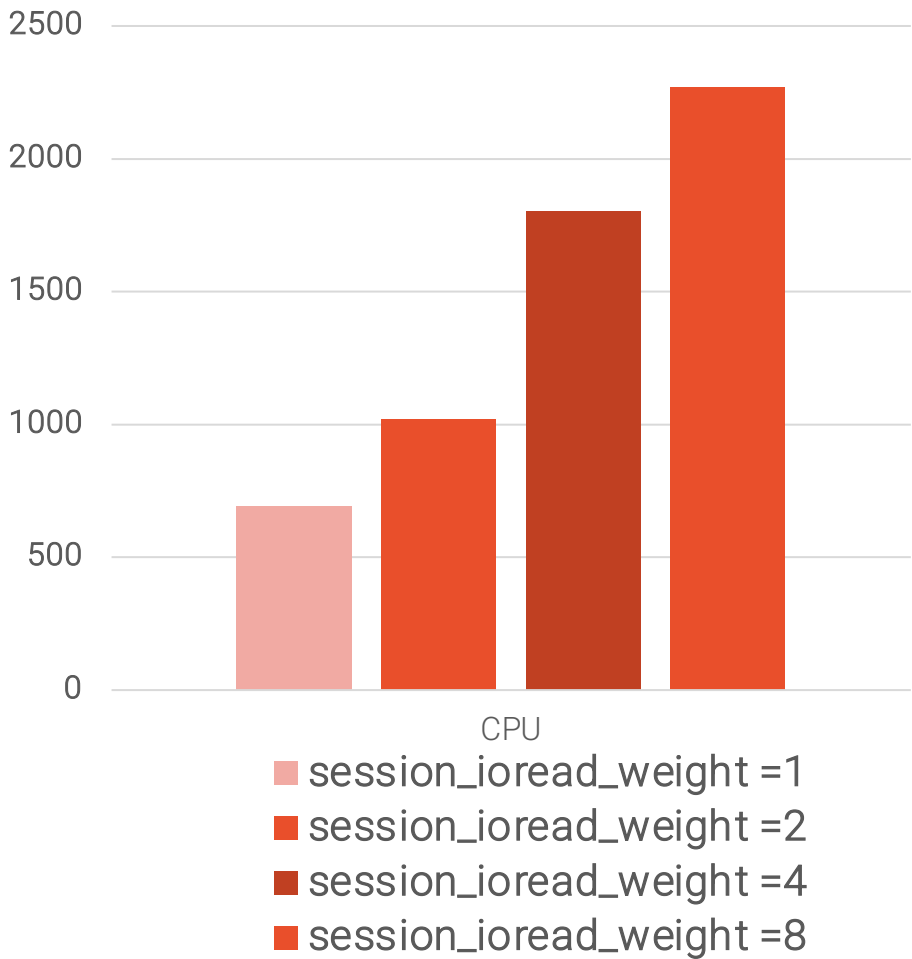


	1	2	4	8
Запуск 1	932	1464	2893	6253
Запуск 2	757	1531	2859	5406
Запуск 3	853	1393	2794	5186
Среднее количество транзакций	847	1463	2849	5615
Соотношение транзакций	1	1,73	3,36	6,63



	1	2	4	8
Запуск 1	426	604	1042	1356
Запуск 2	376	560	889	1275
Запуск 3	370	495	871	1338
Среднее количество транзакций	391	553	934	1323
Соотношение транзакций	1	1,42	2,39	3,39

Задание веса чтения блоков(ioread)



	1	2	4	8
Запуск 1	664	1049	1801	2309
Запуск 2	706	1034	1812	2172
Запуск 3	711	975	1789	2327
Среднее количество транзакций	694	1019	1801	2269
Соотношение транзакций	1	1,47	2,60	3,27

Как это все работает?

$$1 + 2 + 4 + 8 = 15$$

$1 - 1/15 = 0.066666$ – на все задачи

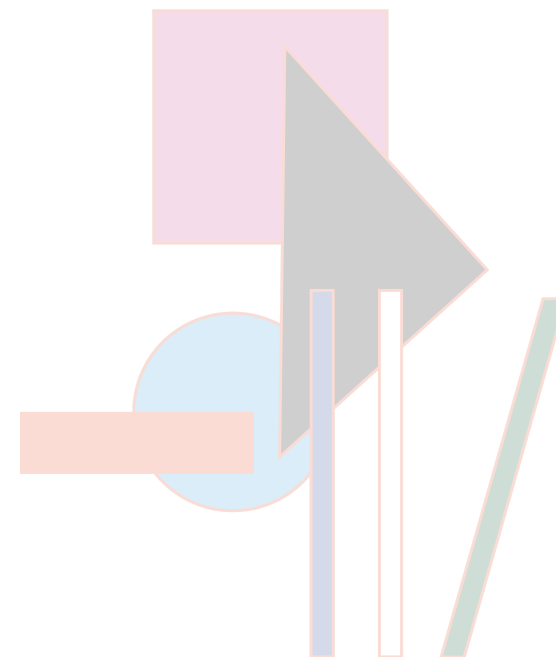
$2 - 2/15 = 0.133333$ – на все задачи

$4 - 4/15 = 0.266666$ – на все задачи

$8 - 8/15 = 0.533333$ – на все задачи

Регулируется задержками sleep-ами.

Запросы, отличные от client backend, не участвуют в приоритизации.



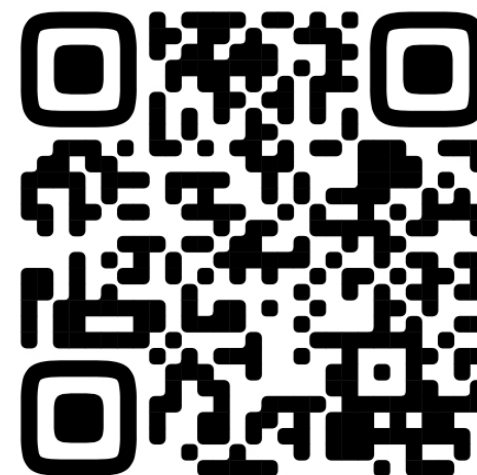
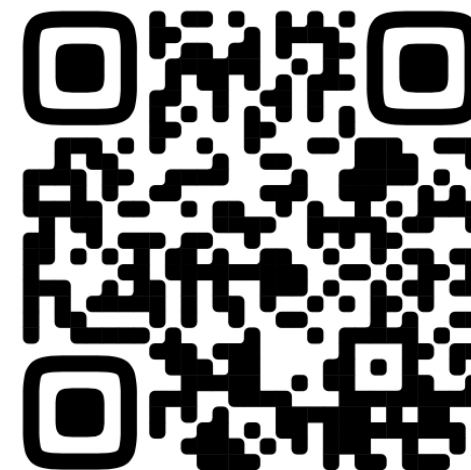
Накладные расходы

- не связаны с замедлением отдельных сессий
- 3-7 %



Ссылки

- <https://postgrespro.ru/docs/enterprise/16/pgpro-rp>
- <https://postgrespro.ru/docs/enterprise/16/runtime-config-resource#RUNTIME-CONFIG-RESOURCE-PRIORITIZATION>



Вопросы?



Спасибо за внимание!

Александр Попов
a.popov@postgrespro.ru

